An Automated Directive Fall Detection System Using Single 3D Accelerometer and Learning Classifier

By

Shaikh Farhad Hossain

MASTER OF SCIENCE IN

INFORMATION AND COMMUNICATION TECHNOLOGY

Institute of Information and Communication Technology

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

February, 2017

This thesis titled, **“AN AUTOMATED DIRECTIVE FALL DETECTION SYSTEM USING SINGLE 3D ACCELEROMETER AND LEARNING CLASSIFIER”** submitted by Shaikh Farhad Hossain, Roll No: 0412312030, Session: April 2012 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of Master of Science in Information and Communication Technology on the 8 February, 2017.

BOARD OF EXAMINERS

CANDIDATE’S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Farhad

Shaikh Farhad Hossain

DEDICATED TO MY PARENTS

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADL	Activities of Daily Living
BPNN	Back Propagation Neural Network
BSN	Body Sensor Networks
DFS	Depth-First Search
DFT	Discrete Fourier Transform
DSP	Digital Signal Processor
DT	Decision Tree
FFT	Fast Fourier Transform
FN	False Negatives
FP	False Positives
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
HMM	Hidden Markov Model
HOGs	Histogram of Oriented Gradients
KNN	K-Nearest Neighbors
LASA	Longitudinal Aging Study Amsterdam
MEMS	Micro Electro Mechanical Systems
NHS	National Health Service
PCA	Principal Component Analysis
PIR	Pyroelectric Infrared
RN	Rovering Network
SVM	Support Vector Machines
TN	True Negatives
TP	True Positives
VM	Vector Magnitude
WHO	World Health Organization
WSN	Wireless Sensor Networks

ACKNOWLEDGEMENTS

First of all, I would like to thank Almighty Allah for his mercy and charity. This thesis is the most significant accomplishment in my life and would have been impossible without the will and wish of the almighty and I am grateful to him.

I would like to thank my supervisor Dr. Md. Zahurul Islam, Associate Professor, Dept. of EEE, BUET. His patience, sincere guidance and encouragement had made this work possible. My sincere gratitude and thanks to him.

I have had the help of a lot of people without whose contribution this work may ever have come to light. The second person to acknowledge should be Prof. Dr. Md. Liakot Ali, Institute of Information and Communication Technology (IICT), Bangladesh University of Engineering and Technology (BUET), for introducing me in the arena of Bio-medical Engineering.

My thanks to all my teachers here at IICT for helping me prepare for this task. I also have had the pleasure of support from the cordial and always helpful staff of IICT over the years that have made my task at IICT that much easier. I would also like to thank the volunteers who came to the lab for giving me their time so I could perform the experiments.

I would like to thank *Dr. Khondaker* Abdullah Al *Mamun,* Associate Professor, Dept. of CSE, United International University (UIU) for his wonderful suggestions to improve the thesis during and after the oral examination.

Finally, my gratitude must go to my family whose unconditional support had made life challenges easier to meet and reach for success.

Abstract

Technology advances to accelerate the quality and type of services provided for health care and monitor. Wearable sensor systems, composed of small and light sensing nodes, have the potential to revolutionize the health care system. An important application of wearable sensors can be the detection of fall with its direction, particularly for elderly or otherwise vulnerable people. In this thesis work, we implemented a direction-sensitive fall detection system prototype using a single three-dimensional accelerometer and machine learning algorithm which includes feature extraction and classification methods, e.g. PCA, SVM and KNN. Four types of fall, forward, backward, left and right falls are detected. In addition to the detection of a fall, it is also important to determine its direction, which could help locate joint weakness or post-fall fracture and help decrease reaction time. Most wearable fall detection algorithms are based on thresholds set by observational analysis for various fall types. However, such algorithms do not generalize well for unseen data sets and their applications in finding the directions of falls are not well recognized. A more appropriate approach, as presented in this thesis, is a machine learning based algorithm SVM and KNN were implemented for fall detection. Among the two methods, SVM provides better performances which leads to 96.45% of accuracy using PCA, mean and standard deviation features, exceeding the performances reported in the literature. The performances of the developed system in real time were also evaluated and they were found satisfactory.

This work not only shows a machine learning algorithm that provides accuracy beyond the currently available algorithms but also shows direction-sensitive and cost-effective fall detection system using single 3D accelerometer.

Chapter 1
Introduction

1.1. Introduction

Fall, particularly among the elderly people, may result in serious injury or even death if the person becomes incapacitated or fails to seek medical aid in a short time. It is at the sixth position in the list of causes of death for the people aged between 60 and 65; the second between 65 and 75 and the first over 75 [1]. The number of elderly people is ever increasing. There are more than 600 million people over the age of 60 around the world and that number is expected to approach two billion in 2050 [2]. Statistics from 2001 showed that 17 per cent of the population in Europe was 65 or older, and by 2035, an estimated 33 per cent will be 65 or older [3]. In Canada, the number of elderly people aged 65 years and older was 10.6 per cent of the population in 1991. This number is expected to increase rapidly, reaching 14.5 per cent by 2011 and 21.8 per cent by 2031 [4]. This future large population of elderly people pose a potential high risk of fall related injury and fatality. Also, among the people affected by Alzheimer's disease, the probability of a fall increases by a factor of three [1]. According to the American Heart Association, Treatment of a patient, experiencing complications due to a fall, within the first 12 minutes after the fall brings a survival rate of 48% -75% [5]. This sensitive post-fall time is one of the key factors that determine the future of an elderly fall patient. Many older fallers are unable to get up again without assistance and any subsequent long inactivity can lead to hypothermia, dehydration, bronchopneumonia and pressure sores [6]. Because of these life-threatening consequences of falls, automated and accurate fall detection is emerging as a big necessity for many countries where the society adopts the culture of independent living for elderly people [7].

Treatment of fall injuries can be very costly. In 2013, the payments of patients and insurance companies totaled to $34 billion for direct medical cost for falls [8]. Over 800,000 patients a year are hospitalized because of a fall injury, most often because of a broken hip or head injury [9]. The costs of treatment of fall injuries go up with age. The condition of a fall patient may deteriorate and can be critical as time passes after the fall. This can further increase the cost of treatment. So, automatic fall detection with high accuracy is a crying need for the societies that adopt independent living of elderly people.

A long-term, continuous monitoring system becomes a necessity for detecting fall of elderly people and for the patients, particularly during their time spent at home. Traditionally, people at risk of falls are provided with pendants containing a button that can be pressed for summon help. But, in case a fall results in a faint, the patient will not be able to press the button and the alarm will not be activated.

Different technologies have been adapted and integrated to support the monitoring of elderly people at independent living or patient in a home environment. Small and light wearable sensor systems have the potential to save lives. Mobile wearable sensor systems and wireless sensor networks (WSNs) in particular, are worn on the body for the purpose of acquiring physiological data. Over the years, there has been an increase in the use of such systems to healthcare for long-term monitoring of patients in their homes. Micro electro-mechanical systems (MEMS) technology for sensors is allowing smaller and lower-power sensors. Wearable sensors have advantages that they are generally smaller and cheaper and able to track the wearer at any location [10]. This facilitates independent living and safety as it allows the person to live normally in their own home ensuring that a caregiver or medical practitioner is alerted if a fall occurs. As no wearer-accessible activation button is needed, the system can be concealed on the person's body more easily. Wearable fall detection system may be used to alert appropriate personnel in the event of a fall. The need for wearable sensors in the healthcare sector continues to increase such as long-term monitoring of patients in their homes. However, the commercialization of this technology is relatively slow due to the following reasons:

- lack of existing communication infrastructure within medical facilities and homes
- low reliability of the detection system, particularly with regard to issues, such as, the accuracy of detection of events, radio interference and battery life.

1.2. Fall Detection System: Present State of the Problem and Motivation for this Thesis

Recently, many studies have been made on fall detection system in terms of hardware (sensors) and algorithms, and both. Different fall detection schemes are reported in the literature that used different types of sensors, e.g. accelerometer, acoustic sensors, gyroscopes, cardio tachometer, magnetometer and barometric pressure sensors [11]. Some schemes used multiple sensors in single system for increased accuracy but they have the disadvantages of being expensive [12]. Some researchers report the use of Doppler Radar [13] which has high cost and complex method to implement. Some used video based fall detection which has video limitation [14]. In terms of algorithm, most fall detection algorithms are based on setting thresholds determined via observational analysis for various fall types [15, 16]. However, such algorithms resulted in high rates of 'false positive' (FP) and 'false negative' (FN) when evaluated on unseen data sets [17, 18, 19]. Existing solutions do not provide the detection accuracy required for the technology to gain the trust of medical professionals. Some schemes used complex data processing algorithms for increased accuracy. Hidden Markov model [20], neural network [21, 22] and fuzzy logic [23, 24], which have high space and time complexity, are some examples. Some researchers used image processing which needed more processing time [25, 26, 27].

1.3. Objectives

The objective of this work is to develop an automated and directive fall detection system with increased accuracy using single accelerometer. To fulfill this objective, the studies have set the following aims:

1. To implement a wireless fall detector using commercial 3D accelerometer module SHIMMER^{TM}
2. To collect and analyze fall data to extract important and sensitive statistical features related to a fall
3. To analyze the accelerometer data for a detected fall to determine the direction of the fall
4. To utilize machine learning algorithms (e.g. SVM, KNN) to configure the system with the selected classifiers for fall detection and compare the results.
5. To compare the performance of the proposed system with existing works

1.4. Overview of the Thesis

This thesis aims to develop a fall detection system based on single sensor and machine learning algorithm with a view to improving the detection accuracy at a reduced cost. For this purpose, a wireless fall detector using a commercial 3D accelerator module has been implemented; fall data of human subjects have been collected and analyzed to extract important and sensitive statistical features related to the fall and its direction; machine learning algorithms, SVM and KNN have been used to configure the system with the selected classifiers; and, the performance of the resulting automated fall detection system are compared with the existing works. The feature of determining the direction of the falls is also included in the study, as it can help determine the locations of joint weakness or post-fall fracture and help decrease reaction time [11].

1.5. Thesis Organization

The rest of this thesis is organized as follows:

In Chapter 2, a brief overview of fall detection systems and different aspects of fall events are described. Surveys on existing works or literature review are also carried out in this chapter. Chapter 3 describes the methodology adopted for the research work and data collection. Chapter 4 discusses the algorithms and features used for fall detection along with the details of the classification method. Chapter 5 presents the experimental results including their discussions. Chapter 6 concludes the thesis with future vision.

Chapter 2
Background and Related Work

2.1. Definitions of falls

According to Kellogg International Working Group, falls can be defined as unintentional coming to ground or a lower level as a result of a sustained blow, loss of consciousness or health related problems. Moylan and Binder define falls as unintentional position changes that result in patients coming to rest on the ground, floor or other lower surface [28]. A fall is an event in which a body's center of gravity quickly declines, according to Liu and Cheng [29].

2.2. Falls in the elderly

The above definitions show a general agreement that falls are unintentional and result in a faller coming to rest on the ground, and may involve causal agents. The following sections further discuss the circumstances of falls.

2.2.1. Circumstances of falls

As age increases, degeneration of body muscles occurs. This degeneration may result in weakness of bones and skeletal system thus being unable to adequately support the body. Trips and slips are also events that can result in falls. Interventions such as clearing obstacles from paths around the home and administering medical treatments to increase muscle strength can reduce fall incidence. Continuous monitoring will allow fallers to be identified in advance before serious falls occur. Incorrect shifting of body weight (which causes the center of gravity of the body to move from the base of support during walking or standing) accounted for around 40% of falls recorded, followed by tripping or stumbling. Slipping was considered to cause the least number of falls.

Overstall et al. considered tripping as the most common cause of falls, but argues that the proportion of falls due to tripping decline with increasing age [30]. The ADL during which falls occur most is walking.

2.3. Typical fall scenarios

The most important scenarios of falls are described [1] below in detail:

2.3.1. Fall from standing

1. It lasts for 1 to 2 seconds.
2. In the beginning, the person is standing. At the end, the head is stuck on the floor for a certain amount of time.
3. A person falls along one direction and the head and the center of mass move along a plane.
4. The height of the head varies from the height while standing to the height of the floor.
5. During the fall the head is in free-fall.
6. During the fall the head traces a virtual circle that is centered in the position of the feet before the fall and has a radius equal to the height of the person.

2.3.2. Fall from chair

1. It lasts for 1 to 3 seconds.
2. The height of the head varies from the height of the chair to the height of the floor.
3. During the fall the head is in free-fall.

2.3.3. Fall from bed

1. It lasts for 1 to 3 seconds.

2. In the beginning the person is lying.
3. The height of the body varies from the height of the bed to the height of the floor.
4. During the fall the head is in free-fall.

2.4. Fall risk factors

A person can be more or less prone to fall, depending on a number of risk factors and hence a classification based on only age as a parameter is not enough. In fact, medical studies have determined a set of so called risk factors: (i) Intrinsic (ii) Extrinsic

2.4.1. Intrinsic risk factors

History of falls: Associated with an increased risk in recurrent falls [31].

Age: Falls increase with age because of a reduced ability to respond rapidly and effectively compared to younger adults. Moreover, studies of reaction time in old people observed a decrease in stepping, step initiation, and execution timing and coordination time – which has also been linked to lower extremity fracture risk – when breaking a fall by outstretching the hand is also delayed [32].

Gender: For the younger elderly, the rate of falls is similar for both men and women; however, among the most elderly people, women fall more often than men and are more likely to suffer from fractures when they fall [33].

Medical conditions: Vascular diseases, chronic obstructive pulmonary disease, depression, and arthritis are each associated with a 32% increased risk. The frequency of falling increases with increasing deterioration due to chronic disease. Moreover, the risk increases with thyroid dysfunction, which leads to an excessive secretion of thyroid hormones, and also with diabetes and arthritis that leads to loss of peripheral sensation. The incidence of falls relevant to cardiovascular causes is unknown in the general population, but vertigo is common in people with falls. Depression and incontinence also occur frequently in populations with falls [34].

Impaired mobility and gait: The reduction of strength and endurance after the age of 30 (10% loss per decade) as well as loss of muscle power (30% loss per decade) lead to a decrease in physical function below the limit. As a result, daily living activities become difficult and then impossible – this is the case in early aging in generally sedentary subjects. When strength, endurance, power, and especially functionality are reduced considerably, it is not impossible for a false trip or a slip to turn into fall. Muscle weakness is a significant risk factor in falls, as well as difficulty in gait, imbalance, and the use of walking aids. Any disability of lower limbs (lack of power, orthopedic disorders, or poor sensation) is associated with increased risk. Having difficulty in getting up from a chair is also associated with increased risk [35].

Drugs: The use of benzodiazepines in older people is associated with a 44% increase in risk of hip fracture and night falls. There is a significantly increased risk of falling when using drugs such as psychotropic, antiarrhythmic drugs, digoxin, diuretics, and sedatives. The degree of prescription of medicines has been increased in chronic disease management. According to almost all studies, the risk increases significantly if more than four medications are taken, regardless of the type of drug [36].

Solitary lifestyle: It may indicate greater functional capability, but injuries and their consequences could be even worse, especially if the person cannot get up from the floor. The fact that someone lives alone seems to be a risk factor in falls, although part of this effect appears to depend on the type of house in which they reside [34].

Race: Evidence from the United Kingdom and the United States suggests that Caucasian subjects fall more often than African tribes of the Caribbean, Hispanics, or South Asians, but there are no studies to report national differences in continental Europe [37].

Attenuated vision: Visual acuity, contrast sensitivity, field of vision, cataracts, glaucoma, and glaucoma plus bifocals or multifocal lenses lead to risk of falls. Multifocal lenses reduce the depth of perception and impair edge-contrast sensitivity when detecting obstacles in the environment. The elderly can benefit from wearing non-multifocal glasses when using stairs and in unfamiliar surroundings outside their home [34, 38].

Foot problems: Calluses on the big toe, long toe defects, ulcers, deformed nails, and general pain when walking increase the difficulty of balance and the risk of falls. Correctly fitting shoes are also important [34, 39].

Cognitive disorders: A lack of understanding is clearly associated with increased risk, even at relatively modest levels. For example, a result of less than 26 or less than 24 on the Mini Mental State Examination is related to increased risk. Poor memory has been proven to be an independent risk factor for falls in people over 75 years, according to LASA (Longitudinal Aging Study Amsterdam). Residents of institutions with dementia fall more than twice as often as people with normal cognition, but there is no difference in the severity of injury between the two groups [34].

Deconditioning/immobility: Those who fall tend to be less active and, through disuse, may cause further irreversible atrophy of the muscle around an unstable joint. Nonactive persons fall down more often than those who are moderately or very active, but fall down in a safe environment [34].

Psychological condition/fear of falling: Up to 70% of people who have recently fallen down and up to 40% of those who have not reported a recent fall confess fears of falling. Reduced physical and functional activity is associated with stress and fear of falling. Up to 50% of those who fear falling limit or exclude social or physical activities because of this fear. Strong links were found between fear and poor posture, low-speed walking and muscle weakness, and poor health self-esteem and reduced quality of life. Women with a history of stroke are at greater risk of falling and experiencing fear of falling. Having four or more medications is also implicated in a fall-related phobia. However, many older people do not appreciate sufficiently the level of danger [31, 34].

Nutritional deficiencies: A low body mass index, which indicates poor nutrition, is associated with increased risk. Vitamin D deficiency is quite common in elderly people living in institutions and may lead to wrong gait patterns, muscle weakness, osteomalacia and osteoporosis [34].

2.4.2. Extrinsic risk factors

The magnitude of the influence of environmental factors on the risk of falls in the elderly is uncertain. Some studies have indicated that in the elderly living in the community, 30%–50% of falls are due to environmental cause’s e.g.

- Slipping floors
- Stairs
- Need to reach high objects
- External Environment:
- Damaged roads
- Crowded places
- Dangerous steps
- Poor lighting
- uneven surfaces

and approximately 20% of falls are due to significant external factors (i.e., those that would lead to a fall in any healthy elderly person). A frequent problem that older people encounter is to slip, trip, or misstep, i.e., a loss of balance where righting mechanisms prevent a fall [34].

2.5. Causes and Consequences of fall

Among elderly people that live at home, almost half of the falls take place near or inside the house. Usually women fall in the kitchen whereas men fall in the garden [40].

The rate of falls increases significantly among elderly people living in nursing homes: at least 40% of the patients fall twice or more within 6 months [34, 41]. This rate is five times more with respect to the rate of fall when people live at home. The most important causes of falls are described in detail:

2.5.1. Physical causes

The factors that lead to most of the falls in people over 65 are to stumble on obstacles or steps and to slip on a smooth surface. The fall is usually caused by loss of balance due to dizziness. Approximately 14% of people do not know why they fall and a smaller number of people state that the fall is due to the fragility of the lower limbs [42].

2.5.2. Activities

Most of the falls happen during the activities of daily living (ADL) that involve a small loss of balance such as standing or walking. Fewer falls happen during daily activities that involve a more significant movement such as sitting on a chair or climbing the stairs. Conversely, activities usually defined “dangerous”, such as jogging or physical exercises are less likely to increase the probability of a fall. There are more falls during the day than during the night. [41]

Accidental falls are the main cause of admission in a hospital and the sixth cause of death for people over 65. For people aged between 65 and 75 accidental falls are the second cause of death and the first cause in those over 75 [1].

2.5.3. Physical damage

Scratches and bruises are the soft injures due to a fall. In the worst cases the injuries are concentrated on the lower part of the body, mainly on the hip. On the upper part of the body the head and the trunk injuries are the most frequent. About 66% of admissions to a hospital are due to at least one fracture. The fracture of elbow and forearm are more frequent but hip fracture is the most difficult to recover from. Such a fracture in fact requires a long recovery period and involves the loss of independence and mobility. Sometimes, when a person falls and is not able to stand up by himself, he lies down on the floor for long time. This leads to additional health problems such as hypothermia, confusion, complications and in extreme cases can cause death [42].

2.5.4. Psychological damage

A fall also involves hidden damages that affect the self-confidence of a person [42]. Common consequences are fear, loss of independence, limited capabilities, low self-esteem and generally, a lower quality of life.

2.5.5. Economic damage

The direct costs associated with falls are due to the medical examinations, hospital recoveries, rehabilitation treatments, tools of aid (such as wheelchairs etc.) and caregivers service cost. Indirect costs concern the death of patients and their consequences. Recent studies have determined that in the year 2000 alone fall-related expenses was above 19 billion dollars and it is estimated to reach 54.9 billion in 2020. This shows that year by year, health costs due to the falls are increasing dramatically [43].

2.6. Related Work

2.6.1. Camera based fall detection

Camera based detection systems make decisions on whether an event is a fall or not (Fig 2.1) by extracting fall patterns from the images captured [44].

Ozcan et al [45] created an autonomous system that is able to provide quick and accurate real-time responses to critical events like a fall. The system is not only able to detect falls but also to classify non-critical events such as sitting and lying down. Their solution is based on a modified version of the histogram of oriented gradients (HOGs) algorithm. When a fall occur edge orientations in a frame vary drastically and extremely fast, as a result of this subsequent frames get blurred. One of the drawbacks of the system is its lack of auto exposure adjustment in the camera. False alarms may be raised if the scenery changes.

Crispim-Junior et al. [46] used a video camera in addition to an accelerometer device (strapped to subject's chest) for fall detection. They considered that, by combining the subject's acceleration with visual information, the detection sensitivity and precision could be improved compared to using only visual data. In their proposed system, the vision component

Fig: 2.1: Camera based fall detection [87]

detected postures such as standing, sitting, lying and changes in postures. The multisensor approach (vision and acceleration) resulted in a system with a sensitivity of 93.5% and precision of 63.6%, while the approach based only on vision produced a sensitivity of 77.3% and a precision of 57.7%.

A major advantage driving the use of camera based systems is that they are non-intrusive because they do not have to be worn on the body. Nonetheless, they have disadvantages that make them less attractive to users, including:

1. The addition of cameras around a home may be considered an invasion of privacy by the occupants due to the fear that images captured on the cameras can be viewed by a third party. Many falls occur in wash-rooms and patients will generally not accept cameras to be installed in such a place.
2. Algorithms developed based on camera data are computationally demanding and require multiple cameras to be installed in and around the house. High specification microprocessors are necessary to deliver fall decisions in real-time. Also, in situations in which there are multiple occupants in a room, it becomes difficult to know whom to track. This increases the computation complexities.

Despite the limitations described, cameras are still in wide use as platforms for fall detection.

2.6.2. Ambient sensor based fall detection

Ambient solutions use sensors installed in the surroundings (Fig 2.2) of users (for example, pressure sensors, and acoustic sensors) [47].

Litvak et al. [48] proposed a system based on floor vibration and acoustic sensing for fall detection. Their system acquired sound and vibration data using a microphone and accelerometer, and the algorithm used pattern recognition techniques to differentiate between ADL, human's falls, and an object being dropped /falling.

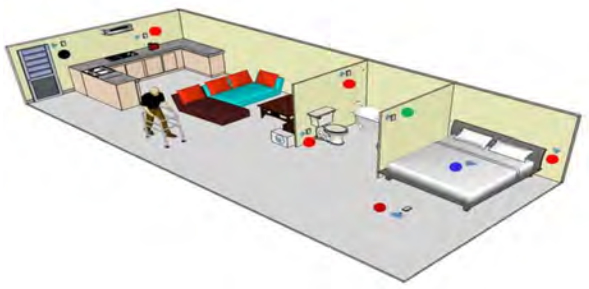


Fig 2.2: Ambient sensor based fall detection [88]

A human-like doll was used in fall simulation and objects such as a bag, plastic box and metal box were used to simulate objects being dropped. The doll was used to simulate 48 forward falls, while the objects were dropped 78 times. An evaluation of the algorithm showed a sensitivity and specificity of 95%. As pointed out by the authors, the fall detection system is not sensitive to low impact falls and was only tested for distances between 2 meters and 5 meters.

Luo et al. [49] developed a fall detection system using 7 pyro-electric infrared (PIR) sensors to detect the heat energy emitted by individuals within a room. Each PIR sensor was sampled at 25 Hz and detected the variance of the thermal heat flux within each section of a room. Then, a 2-layer Hidden Markov Model (HMM) classifier was used to model the time varying PIR signal. PIR sensors were used in order to avoid infringing individual's privacy as can happen with cameras. Eighty falls were simulated, but only 87% were classified correctly.

Khawandi et al. [50] used multiple webcams to perform fall detection. Their algorithm detects faces and measures the speed with which detected faces move toward the ground. Based on a set threshold, it determines if a fall has occurred or not.

2.6.3. Fall detection by image processing

Liu and Zuo [51] proposed an algorithm that compares the ratio of the width and height of a person while standing and lying, the ratio of the area of a person's figure to the area of the room and the rate of variation of an image (Fig 2.3) during a fall.

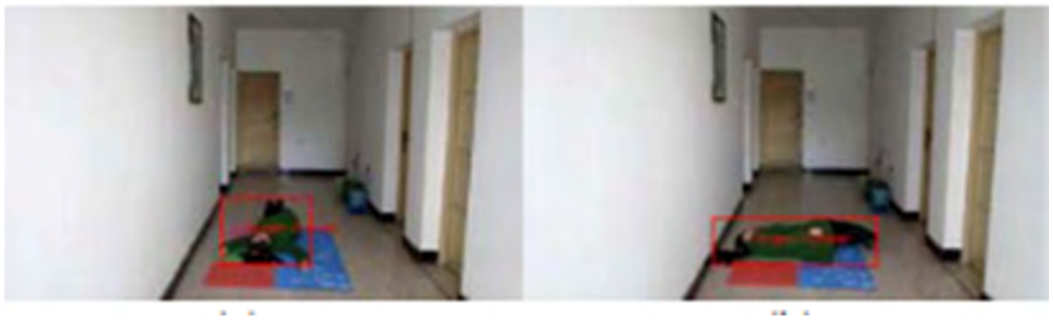


Fig 2.3: Fall detection by images processing [51]

They concluded that by computing the three features on each image frame, their system will prevent FPs, and thus increase accuracy. However, evaluation results were not presented.

Olivieri et al. [52] extracted velocity information across video frames and trained a machine learning algorithm to detect falls. Their system was able to detect 99% of falls, but the number of FPs recorded was not reported.

2.6.4. Thresholds based fall detection

Most wearable fall detection algorithms are based on thresholds set to discriminate between falls and ADL. However, majority of the thresholds are set based on observational analysis of acceleration and angular velocity signals.

Bashir et al [53] proposed a system based on a wireless body area network. It uses a tri-axial accelerometer, and a tri-axial gyroscope sensor. Three stages are used to determine human status namely, fall, ADL, and sleep. The algorithms used are threshold-based and very simple. It employs the posture angle, angular velocity, and acceleration to determine if a fall has occurred. The accuracy for ADLs was 100%, while the sensitivity was 81.6%.

Jantaraprim et al. [54] computed vector magnitude (VM) for a 3D accelerometer mounted on subjects' trunk region. A threshold was set by observing the VM signal to discriminate between falls and ADL. Similarly, Ivo et al. (2011) set thresholds manually for VM in their fall detection algorithm.

Ojetola et al. [17] showed that VM alone is sufficient to accurately detect falls. Sudden movements, transitions from one posture to another and walking do generate high VM similar to falls. Hence, algorithms based on thresholds set for VM alone will trigger false alarms.

Li et al. [55] proposed an algorithm that used a 3D accelerometer and 3D gyroscope for

fall detection. Their algorithm use thresholds set for acceleration and angular velocity to determine if transition to a lying position is intentional or not, and it is based on thresholds set for VM and postures. An unintentional transition to a lying posture was considered a fall. Basically, their algorithm only detects falls in which fallers end-up in a lying posture. Conversely, activities which are not falls, but result in lying position will trigger False Positives (FP).

Wang et al. [56] proposed a fall detection system based on a 3D accelerometer placed behind subjects' ears. Their algorithm was based on simple rules and thresholds set by observational analysis of acceleration data. VM of acceleration, magnitude of horizontal acceleration, time from start to end of a fall and velocity were computed as part of their algorithm. During falls, acceleration signals vary considerably from subject to subject and for different fall types. Hence, thresholds set for features based on observation of acceleration signal only will result in high level of FPs and FNs.

Anania et al. [57] implemented a fall detection algorithm based on 3D acceleration data sampled at 100 Hz. A Kalman filter was used to separate the signal component due to gravity from acceleration data and then the trunk inclination angle was computed. Anania et al.

defined two thresholds; one for the subject’s tilt angle and the second for the rate of change of tilt angle. A fall is detected if the subject’s tilt angle is greater than the first threshold and when the change in the tilt angle over a short period is greater than the second threshold. The main drawbacks with this algorithm are i) thresholds set manually do not generalize well for unseen subjects, and ii) only 2 postures are considered as corresponding to falls, however fallers may end up in other unrecognized postures such as crouching and kneeling.

2.6.5. Smartphones based fall detection

A growing number of wearable fall detection systems are now based on smartphones; this is due in part to the inclusion of accelerometers in smartphones.

Sposaro and Tyson [58] used an android-based smartphone as a platform for fall detection. Their algorithm was based on threshold set for acceleration data. They noted that their algorithm triggers a false alarm when the phone is dropped to the floor.

Kaenampornpan et al. [59] in their study assumed that phones are placed in the left breast pocket and thresholds were set for the minimum and maximum acceleration reading of a subject’s body during falls and ADL. Before their algorithm is used, it is expected that subjects will simulate falls first so that appropriate unique thresholds are for each subject.

Two major draw backs exist with this approach.

1) A fall detector should be trained with more than one fall instance from one subject.

2) The elderly, many of whom are frail, will not be able to simulate falls before using the proposed solution.

Some of the main challenges with the use of smartphones as a platform for fall detection are:

1) The need to understand how individuals use their phones so that algorithms can adapt to differences between normal use of phones and falls,

2) The need to track orientation of a phone during normal use since fall algorithms often consider subjects orientations, and
3) The need to differentiate between when a phone is dropped and when a fall actually occurs.

From the literature reviewed above, it is evident that a major challenge in fall detection is identifying appropriate thresholds that can discriminate between falls and ADL. Human gait patterns are complex and vary considerably for different subject set. As a result, thresholds set manually will not allow for algorithms that generalize well for different subjects and different fall types to be developed.

2.6.6. Machine Learning Based fall detection

In the previous section, algorithms based on thresholds set by an observational analysis of fall data were discussed and their limitations are identified. This section reviews the literature with regard to machine learning algorithms for fall detection. It is to mention again here that the work in this thesis proposes a machine learning approach for fall detection.

Zhang et al. [60] proposed one-class support vector machine (SVM) for fall detection, Liu and Cheng [61] also proposed SVM to discriminate between falls and ADL.

Decision tree based algorithm was proposed by Zhao et al. [62]. They identified faller's locations by using wireless network infrastructure distributed within a building, with notifications being sent to carers whenever falls were detected. A tree based machine learning algorithm was implemented for fall detection and features such as mean, standard deviation, slope, energy and correlation were used as input. Ten subjects (5 for training and 5 for testing) were recruited for their experimentation and the phones were strapped to subject waist. No false alarm was recorded and the system had a recall of 75%.

Shi et al [63] developed a fall detection system based on an android-based smartphone. It integrates an SVM. The proposed technique uses the acceleration data from the phone's accelerometer to detect a fall. The fall detection process is divided into five phases namely, normal, unstable, free fall, adjustment, and motionless. An acceleration threshold is used to trigger the five-phase feature extraction method. A 16-elements vector is obtained as a result of the extraction method. This vector is fed into a SVM that is used to differentiate falls from ADLs. The acquired results were the following: recall 90% and precision 95.7%.

Liu and Cheng [61] proposed the use of an SVM for fall detection. Features were developed using 3D acceleration data sampled at 200 Hz. The features extracted include the VM, the difference between the maximum and minimum acceleration for each axis of acceleration, the vertical acceleration and the tilt angle.

Sengto et al [22] proposed a fall detection system algorithm based on a back propagation neural network (BPNN). The system utilizes a tri-axial accelerometer mounted on the user's waist in order to collect his/her acceleration data behavior. Human activities are divided in three groups: falling activities (forward, backward, right and left), slow motion activities (walking, getting up from bed, flopping), and sudden motion activities (running, jumping). An acceleration threshold is set to differentiate between slow motion activities and other activities. The overall recall of the detection algorithm was 96.25% while the specificity was 99.5%.

Humenberger et al [64] developed a bio-inspired stereo vision fall detection system. It utilizes two optical detector chips, a field-programmable gate array (FPGA), a digital signal processor (DSP) and a wireless communication module. The optical chips capture video frames. The FPGA creates the input data for the DSP by calculating 3D representations of the environment. The DSP is loaded with a neural network that is used for classification purposes. Falls are divided into 4 states or phases pre-fall, critical, post-fall, and recovery phase. To run the experiments the hardware was mounted on the top corners of a room in order to monitor the subjects of interest. The trial results are 90% of fall detection.

Takeda et al. [24] developed a foot age assessment system that estimates how likely a person is to fall based on his/her balance ability and gait condition. The system uses mat type distribution sensor to gather the SOI’s gait characteristics. Fuzzy logic the system is able to make educated guesses. The fuzzy membership functions were obtained through a learning process. The system was not reliable method

Zhang and Sawchuk [65] proposed a fall detection framework that combines decisions from a fall detection algorithm with context information using a Bayesian network. The context information includes physical activity level, personal health record, blood pressure level, heart rate and location (indoor or outdoor). However, gathering physiological data such as blood pressure level and heart rate requires additional sensors to be worn by subjects and thus affect the acceptability of such systems.

Lan et al. [66] embedded a 3D accelerometer, 3D gyroscope and two pressure sensors in a walking cane. The two pressure sensors were fixed to the handle and the tip of the cane and measure the grip and the downward-push force, respectively. The accelerometer and gyroscope measure the acceleration and angular velocity of the cane. A Decision Tree (DT) and subsequent matching (a technique in data mining for finding exact or closely matching segments of a much longer sequence) were used to discriminate between falls and ADL. Data were sampled at 26 Hz. The main challenge of the system is in differentiating between whether an individual has fallen or the cane was just dropped or left on the floor. Furthermore, authors noted that the system gives FNs in cases where the cane hits an obstacle midway during a fall before coming to rest.

Chen et al [67] developed A human fall detection system using a computer vision approach is introduced. The solution is capable of detecting fall-related events in real time using skeleton features and human shape variations. The system is able to extract the human posture and reduce the computational burden by using a 2D model instead of a complicated 3D one. The skeleton (a spanning tree) is acquired by running the well-known graph traversal algorithm Depth-first search (DFS) on the center of the triangular meshes. A distance map is used to calculate the distance between two skeletons. A fall is detected if the user's motion does not change within a certain period of time. The system is able to obtain a high detection accuracy (90.9%) while maintaining a low false alarm rate.

Gjoreski et al [68] proposed combines posture recognition with thresholds set by observation analysis to detect falls. Their algorithm uses 3D acceleration data sampled at 6 Hz. The extracted features were VM, tilt angle, mean of accelerometer x-axis, Root Mean Square (RMS) of VM, standard deviation of VM and change in VM. Postures (such as such as lying or sitting on the floor) are recognized via a Random Forest machine learning algorithm. A fall is detected by combining the recognized posture with a threshold set for the VM. If a subject's posture is lying or sitting and the VM goes above the threshold, then a fall is detected.

2.7. Categorization of fall detectors

Wearable fall detectors are generally categorized into three generations, namely, first generation, second generation and Third generation [69].

2.7.1. First generation

First generation systems rely only on the user interaction. Often known as a pendant around the neck or wrist bands, the user must push a button to contact the call center or emergency services. This type of fall detector do not possess any form of intelligence, they rely entirely on the user pushing a button in order to summon help. In circumstances where the user is unable to push the button (for instance, in case of unconsciousness), help will not be available and such a case could result in aggravated consequences.

2.7.2. Second generation

Second-generation systems that are based on the first-generation systems have an embedded level of intelligence. The second generation comprises fall detection devices and life-style monitoring systems, and includes worn automatic fall detectors that are triggered without the wearer having to press a button.

2.7.3. Third generation

Third-generation systems use data, often via ambient monitoring systems, to detect changes (e.g. changes in activity levels) that may increase the risk of falling (or risk factors for other negative events). The third-generation systems are more preemptive rather than reactive approach.

2.8. Existing wearable fall detectors

There are many existing fall detection products on the market to assist elderly (Fig 2.4).

a. Wrist-Worn: An integrated health monitoring instrument with a tele-reporting device for telemedicine and telecare (Fig 2.4: a).

b. MCT-241MD: It is a stylish wireless fall detector that functions both as a standard manual emergency alert button and as a fall detector, which automatically triggers a call to the monitoring center for immediate help. The fall detection is enabled by a built-in tilt sensor that can sense when the detector, which is worn by the user, tilts at more than 60^0; for more than a predefined period of time (usually about a minute). This activates an alert transmission to the control panel, which notifies the monitoring station, enabling help to be dispatched immediately (Fig 2.4: b).

c. Galaxy fall detector: Fall detection system that will immediately detect a fall and get the help you need on its way. (Fig 2.4: c)

d. iHelp Smart Fall Detector: The device simply clips onto your clothing and is small and un-obstructive to wear. The iHelp device can also remind a user to take any medicine they might be taking at specific intervals. GPS coordinates can be queried by relatives and sent in the event of a fall. These coordinates are also sent to a dedicated care support network. (Fig 2.4: d)

e. Oval Fall Alert: This is the world's smallest and sensitive fall detector. Designed to rest easily in the palm of your hand and sized so one can carry it anywhere. It is able to detect a fall and automatically send a signal to the monitoring station (Fig 2.4: e).

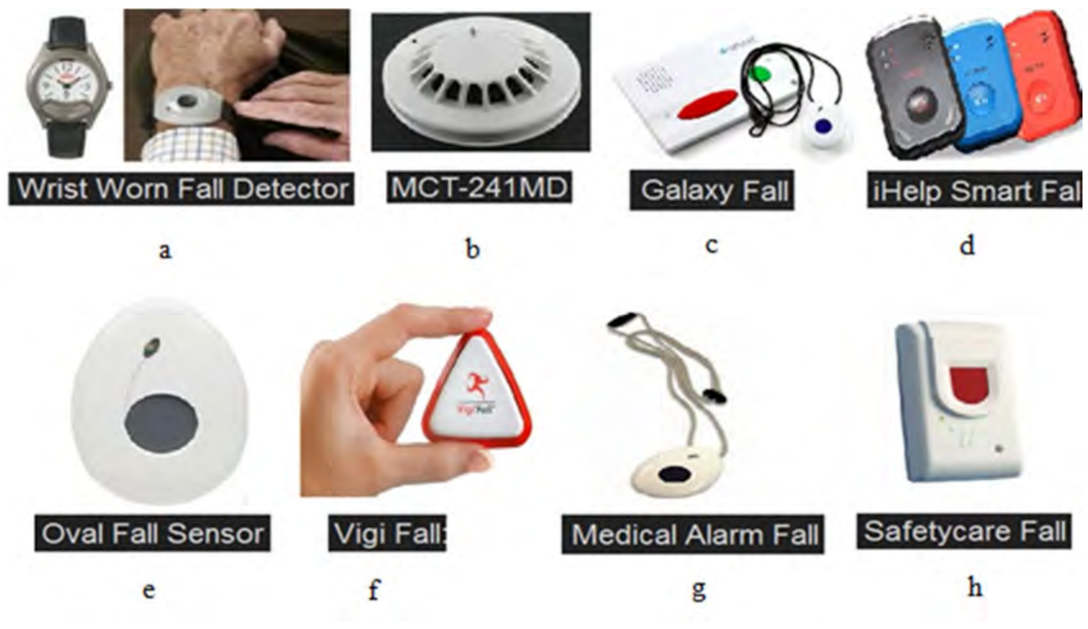


Fig 2.4: Commercial fall detectors

f. The Vigi' Fall: This solution generates an alert automatically in case of a fall followed by the inability to get up by oneself. It is dedicated to elderly persons living at home, to geriatric institutions (nursing homes, sheltered homes, etc.) and hospital premises (acute, rehabilitation, long term care settings) (Fig 2.4: f).

g. Medical Fall Alert: These systems feature sensors (multiple accelerometers and processors) that can detect between normal activity, and an actual fall. By continuously measuring the speed of movements in all directions, the fall detector can compare what it senses to what it considers an actual fall (Fig 2.4: g).

h. SafetyCare: It is an emergency alert system with monitoring service designed for use by seniors or individuals at risk for falls (Fig 2.4: h)

Table 2.1: Wearable sensors for falls and activity monitoring

Sensor	Measurement	Products
Goniometer	Angles (for example: angle of joint movement)	Motion Lab Systems Electro Goniometers
Gyroscope	Angular velocity	SHIMMER sensors, Xsens MVN BIOMECH
Accelerometer	Acceleration	Vigi Fall, Brickhouse Fall Detector
Actometer	Motion	Timex Model 108 Motion Recorder
Pedometer	Step counter (counts number of steps a person takes)	Omron Pedometers
Insole pressure plantar sensor	Pressure distribution across the sole of the foot	Pedar System

2.9. Traditional Direction Measurement System

In traditional methods to find out direction, gyroscope (Fig 2.5) or more accelerometers (Fig 2.6) are used.

Fig 2.5: Gyroscope [89]

Using the key principles of angular momentum, the gyroscope helps indicate orientation. Gyroscopes, or gyros, are devices that measure or maintain rotational motion.

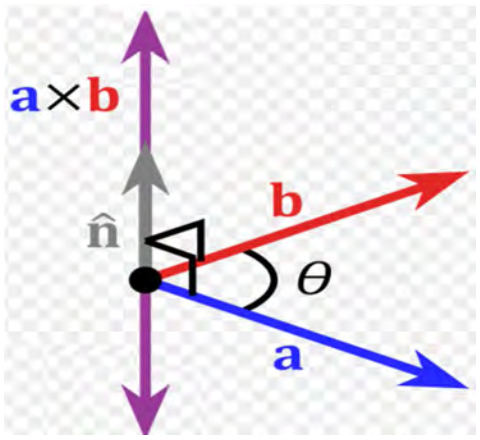


Fig 2.6: 3 Accelerometers with cross product [90]

In determining the direction with the help of more than one accelerometer, the cross product to find the angle between two sensors is used.

In our thesis, there is no gyroscopes sensor or more than one accelerometer. Based on, axis amplitude we successfully find out the directions.

2.10. *Issues with existing wearable fall detectors*

There are currently numerous commercial fall detection systems available in the market. Some of the brands are: Vivatec's wrist care , Tynetec, FALLWATCH (Vigi' Fall), activPAL , Philips Lifeline AutoAlert pendant, Brickhouse fall detector , SafeGuard, Task Community Care fall detectors and Tunstall fall detector (Fig 2.4). Despite numerous commercial and research based solutions, automatic fall detection has several outstanding challenges. A major reason for low acceptance of automatic fall detectors is the high level of FPs and FNs [70]. Both FPs and FNs result in lack of trust for the system. For instance, Ward et al. [71] reported that health and social care staff are not convinced about the benefits of automatic fall detectors.

Chapter 3
Methodology and Data Collection

3.1. System Design

The system consists of two main parts: Body sensor network (BSN) and the monitoring application (Fig 3.1). The BSN consists of wearable sensor that collects accelerometer data. Sensor has a wireless Bluetooth to communicate with the user's server side.

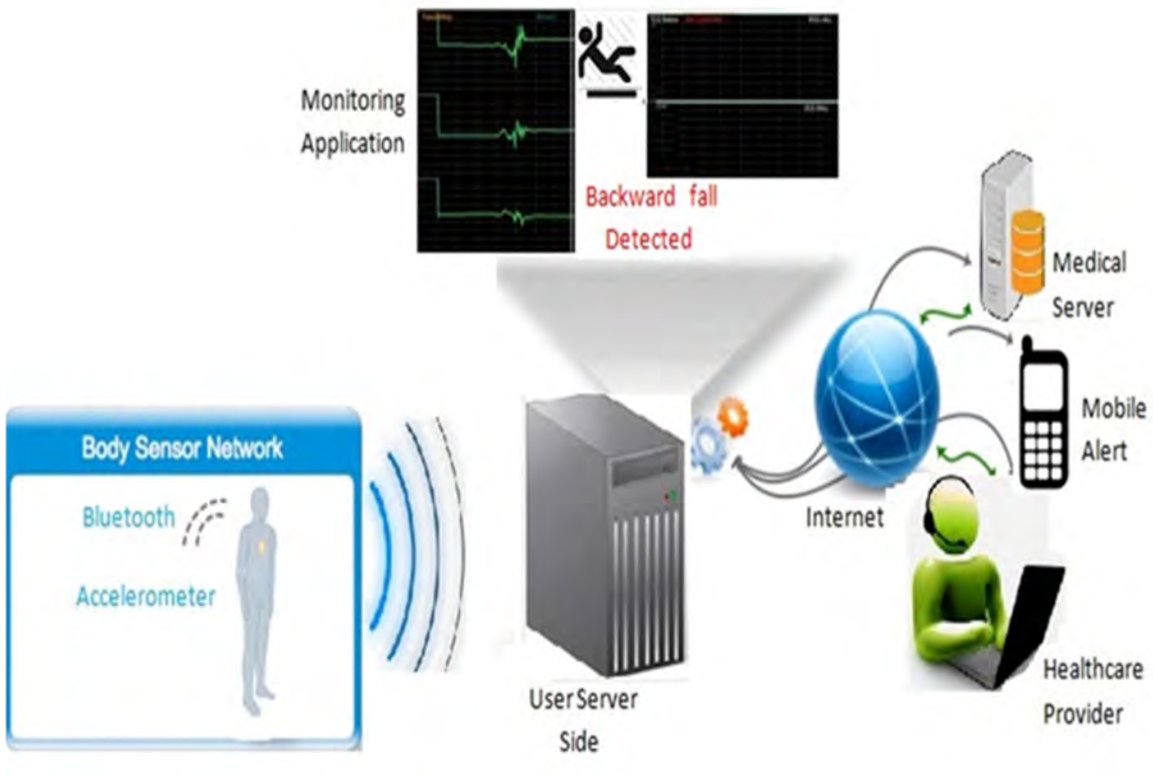


Fig 3.1: Data flow diagram of the direction-sensitive fall detection system

The monitoring application is installed in the user's server side (patient's home or health institute). The server should have the Internet connectivity to send alert to the healthcare provider or mobile message to relative and send/retrieve data from the medical server.

If a patient’s fall occurs, then user server generate the alert by triggering alarm locally as well as sending notification to healthcare provider via Internet and message to mobile phone. Patient fall related information will be recorded in the medical server for further requirement. The patient’s therapist has specific interface for viewing and manipulating the sensory data and an administration panel is implemented to handle the exceptional data.

3.2. Use of accelerometer

Motion acceleration is widely used to classify the type of executed activity by analyzing the change of accelerations with respect to time. Therefore, the accelerations are used to assess postures and motions by calculating a vector of features, such as mean and standard deviation of the acceleration signal. Attaching an accelerometer to different body segments helps identify the type of activity and distinguish one activity from another by analyzing the data and identifying appropriate vector of features.

3.3. Hardware platform

We used the SHIMMER [10] wearable wireless sensor, an acronym for Sensing Health with Intelligence, Modularity, Mobility and Experimental Reusability. We picked the SHIMMER (Figure 3.2) due to its low-power consumption, lightweight (27g) and small size (53 x 32 x 19 mm).

Fig 3.2: Shimmer Sensor [79]

The SHIMMER sensor holds a lithium-polymer battery, and a TI MSPP430 microprocessor with 10 KB of RAM and a flash memory of 48 KB of capacity. It supports wireless communications through its Bluetooth wireless module. The Bluetooth device (Rovering Network RN-42) has a range exceeding 10 m, a default transmission rate of 115 kbaud, and is a class 2 Bluetooth module. The battery life of the SHIMMER mainly depends on the type of application installed on the tiny node platform and the type of selected mode of power consumption. Figure 3.3 shows SHIMMER block diagram and its integrated devices.

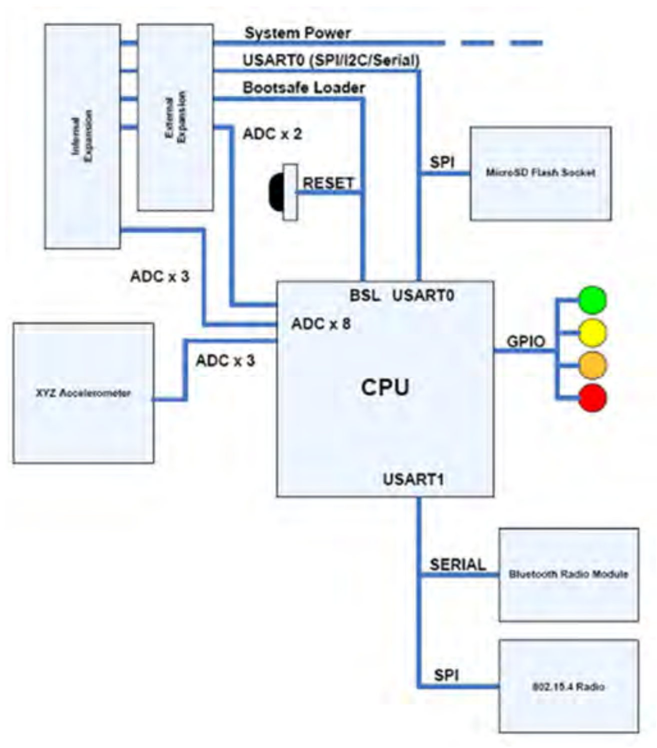


Fig 3.3: Block diagram of the SHIMMER base board interconnections and integrated devices [10].

The sensor node consists of a 3D accelerometer. The tri-axial accelerometer (MMA7260Q) from Free scale Semiconductor has a range up to ±6g. A Micro-Electro-Mechanical Systems (MEMS) accelerometer behaves as a mass on a spring which is displaced when it experiences acceleration. The displacement of the mass is measured to determine the acceleration of the sensor. An accelerometer measures an acceleration of g = 9.81 m.s-2 (1g).

IEEE 802.15.4 compliant wireless communications SHIMMER uses a Chipcon CC2420 radio transceiver and gigaAnt 2.4 GHz Rufa™ antenna. The CC2420 [72] is designed for low-power and low-current applications (current usage 17.4 mA for transmission and 18.8 mA for reception). The radio may be turned off by the MSP430 for low-power operation. The CC2420 is controlled by an SPI connection over the USART1. The CC2420 has support for applications such as packet handling, data transmissions, data encryption, received signal strength, link quality and packet timing, the work load on the MSP430 controller is reduced. Lowering the duty cycle of interaction between the radio or microSD card can be used to extend battery life, however this is not feasible with applications that require high frequency data capture. SHIMMER2 hardware provides features to simplify application programming and enhance event driven applications which can be used to improve power management.

A push-button power controller is used to control the board power-on sequence. From an “off” state, board reset is low and the board power regulator is disabled. When the reset button is pushed, the regulator is enabled and the processor is brought out of reset after a short delay. Short subsequent reset button pushes will generate a board reset. A long button push (preset to 6 s but HW customizable) will shut down the board regulator. When the battery voltage falls below a preset kill voltage, e.g., 2.5 V, a signal is also generated to power-off the board.

The accelerometer, microSD card, 802.15.4 radio and Blue- tooth radio module can be powered off by firmware when not in use. The digital serial number IC component can also go to auto sleep mode when not in use [73].

3.4. Network Setup

The communication between the BSN (Body Sensor Networks) and the user server side is carried out using a wireless Bluetooth connection (Fig 3.4). Bluetooth is a low-power consumer and low-cost option, but has short transmission range.

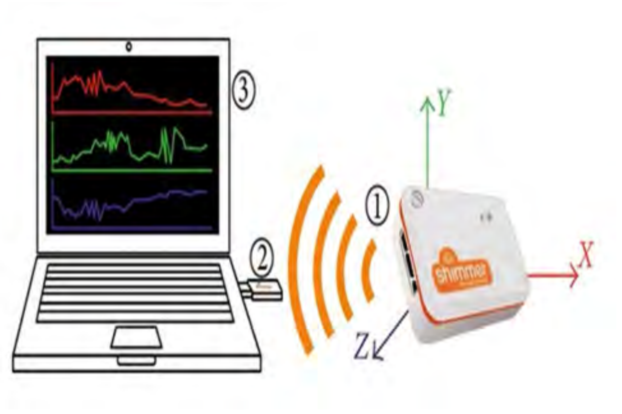
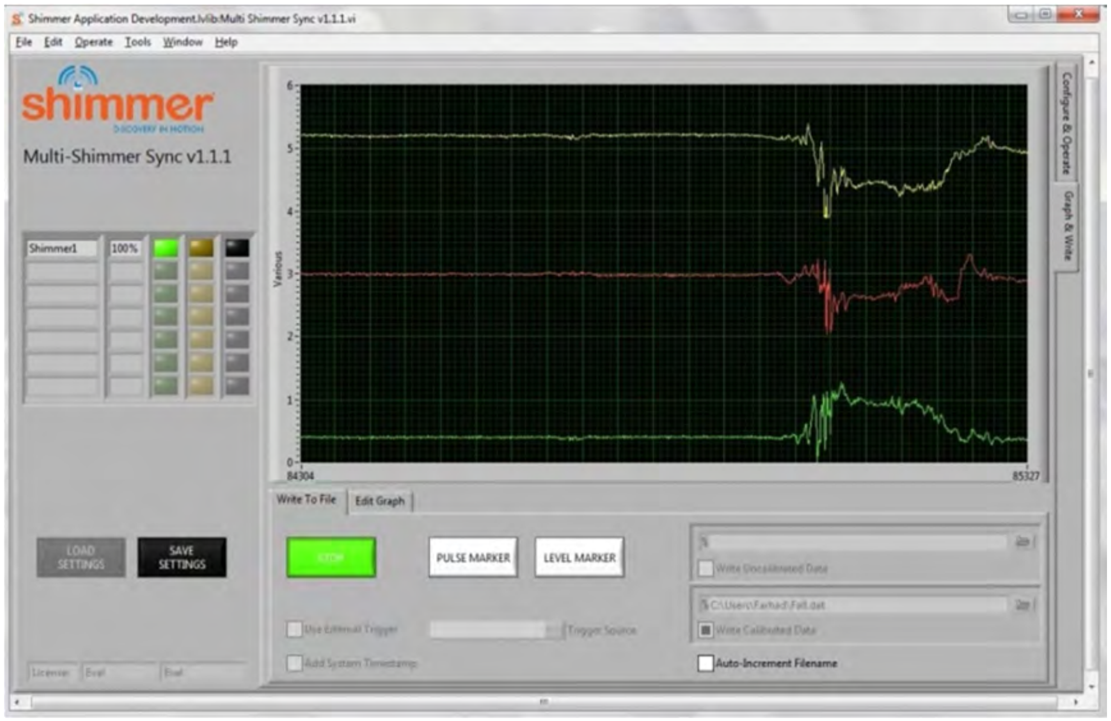


Fig 3.4: Connection and Data Streaming [79]

We used the 'Multi Shimmer Sync' software for capturing data. A screenshot of the user interface of the software is shown in Fig. 3.5. The accelerometer sensor captures the change



C:\Users\Farhad\Fall.dat
Write Uncalibrated Data
Write Calibrated Data
Use External Trigger
Trigger Source
Add System Timestamp
Auto-Increment Filename

Fig 3.5: Multi Shimmer sync

of accelerations over the x, y, and z-axis along with the timestamps and sends them back to the monitoring 'Multi Shimmer Sync' application. 'Multi Shimmer Sync' is ideal for users looking to develop applications where simultaneous data capture from a number of units is required and interpretation and analysis require the data to be synchronized with a high level of accuracy (synchronization error of the order of milliseconds).

3.5. Sampling frequency

Sampling at low frequencies implies a low processor resource requirement and low power requirements. We use sampling frequencies 50 Hz for fall classification. Other fall detection also used data sampled at 10-200 Hz. Considering battery life and detection accuracy; we set 50 Hz of sampling frequency. High frequency gives high accuracy but low battery life.

3.6. Placement of Sensors

There are a number of body locations used for sensors in the literature, including: waist, thigh, hip, trunk, chest, lower back, lower leg, wrist and behind the ear. Some investigation has been carried out regarding the best location for sensors to provide optimum algorithm performance. Doughty et al. [74] evaluated a Tunstall fall detector strapped to the chest, waist, knee, wrist and arm. They concluded that the chest and waist are the most appropriate locations of the body to place fall detectors.

Similarly, Gjoreski et al. [75] investigated the placement of sensor nodes on the chest, waist, thigh and ankle. Results showed that placing a fall detector on the chest provided the best performance. The literature reviewed suggests that the chest is the best location to place a sensor node for fall detection. Thus, the work in this thesis compares the performance of a fall detection algorithm based on sensor node placed on the chest.

Acceleration data (in three dimensions) was gathered from Shimmer sensor nodes placed on each subject’s chest. Data were sampled at 50 Hz and transmitted via Bluetooth to a PC for further processing. An overview of the data gathering set-up is shown in Figure 3.6. Subjects were asked to perform various falls and daily activities. The resulting data is collated, analyzed and used to feed the algorithms presented in this thesis.

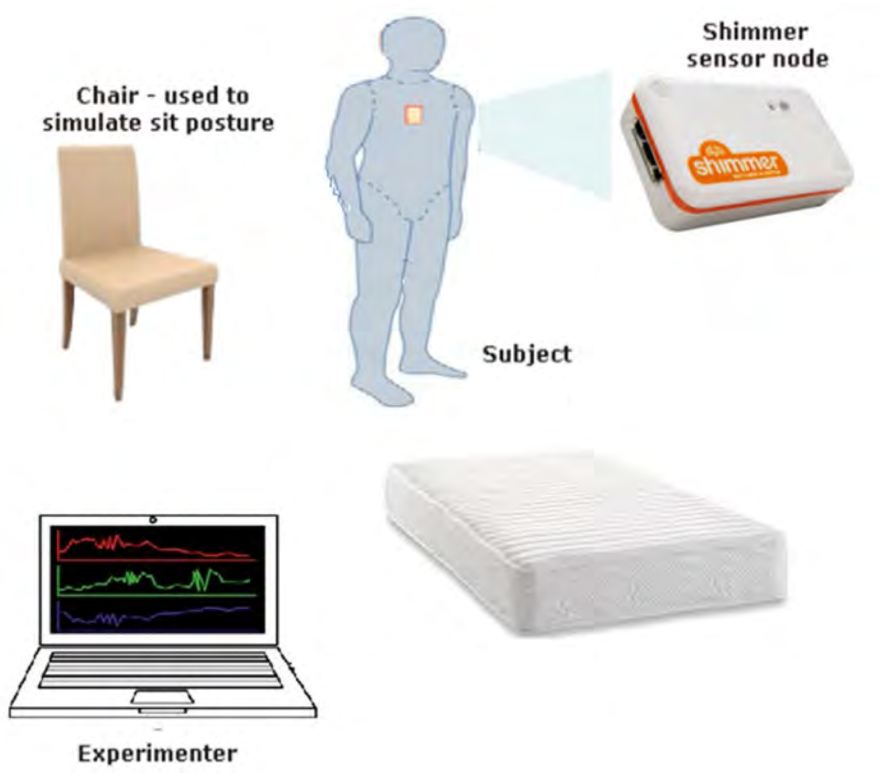


Fig 3.6: Data collection set-up

The falls acted in a laboratory environment are similar to those experienced by fallers under real-life scenarios.

3.7. Participant experimental step

This section describes a set of steps engaged with by subjects during data acquisition.

3.7.1. Subjects recruited for data acquisition

Thirteen subjects were recruited for this work. Before the subjects were recruited, approval was taken from the Director, IICT, BUET. Participation was completely voluntary.

3.7.2. Consent form

Consent form was signed by each participant. Verbal explanations were also provided to each subject in order to ensure that participants understood what was required of them.

3.7.3. Age and height

The youngest subject was 25 years and oldest 55 years old. The mean for the age, height were 37.47 years and 5.37 feet respectively. The details of the thirteen subjects are shown in Table 3.1.

Table 3.1: Ages and heights of the subjects recruited for the experimental fall data collection

Subjects	**Age(years)**	**Height (feet)**	**Gender**
S1	26	5′6″	M
S2	40	5′4″	M
S3	38	5′5″	M
S4	55	5′1″	M
S5	30	5′5″	M
S6	26	5′5″	M
S7	32	5′4″	M
S8	44	5′6″	M
S9	50	5′	M
S10	52	5′4″	M
S11	38	5′9″	M
S12	26	5′5″	M
S13	30	5′4″	M

3.7.4. Data acquisition

The SHIMMER sensor was worn by subjects during data acquisition. The sensor node strapped to the chest (Fig 3.7) of subjects was used for data acquisition and transmission from subjects to a remote PC.

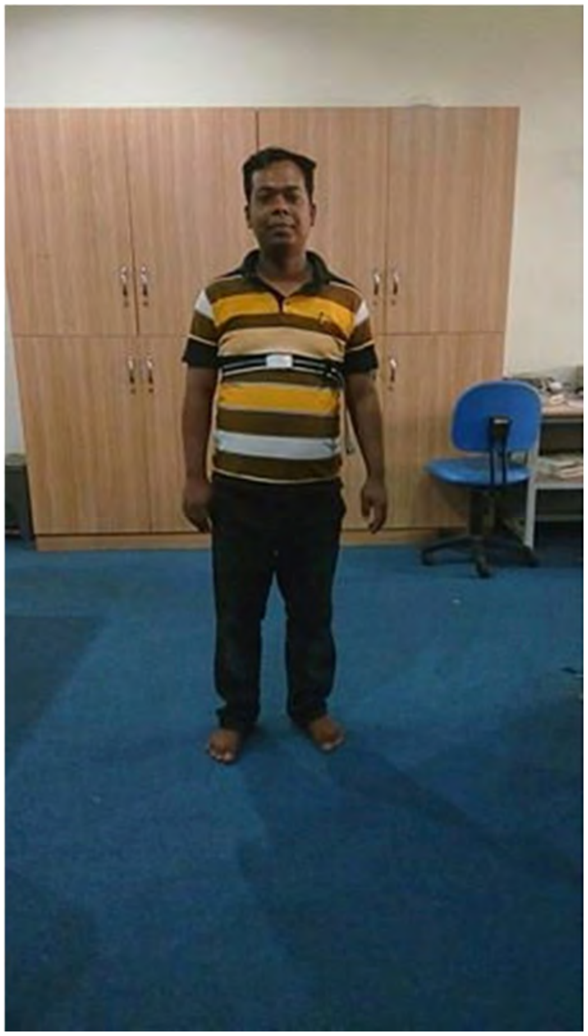
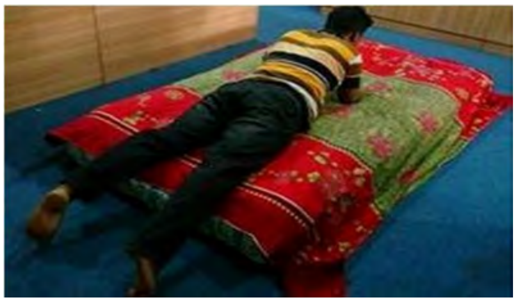
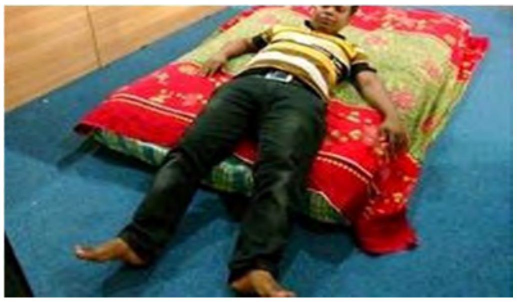


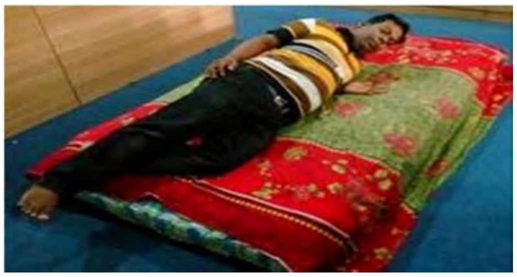
Fig 3.7: Placement of SHIMMER sensor nodes

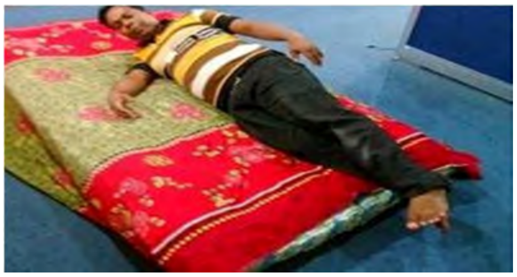
3.8. Falls and Activities of Daily Living

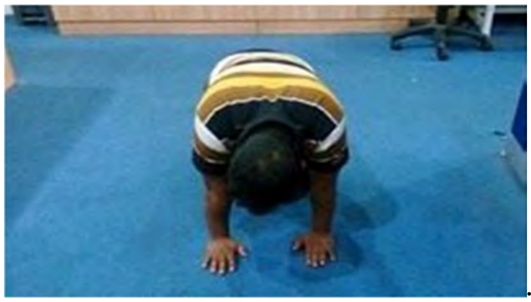
Subjects were directed to (Fig 3.8) fall down in forward, backward, left, right directions and to do the normal activities ADL, like, idling, lying down ,sitting down, bending down, down, going down head, knee, walking, jogging, standing up, up, jumping. It was assumed that in real-life people will normally engage in activities.

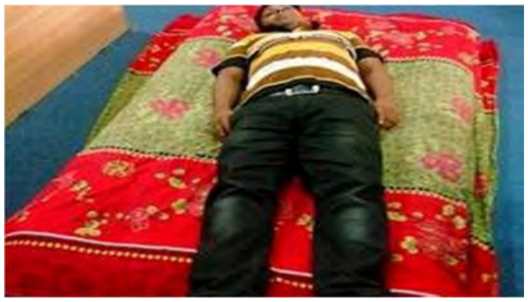

01.Forward fall

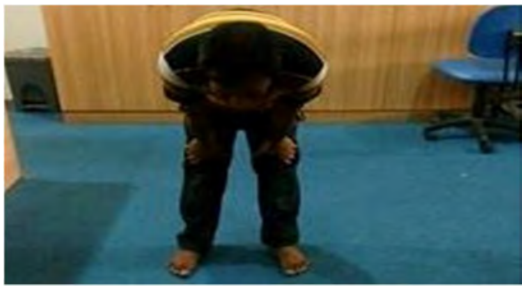

02. Backward fall

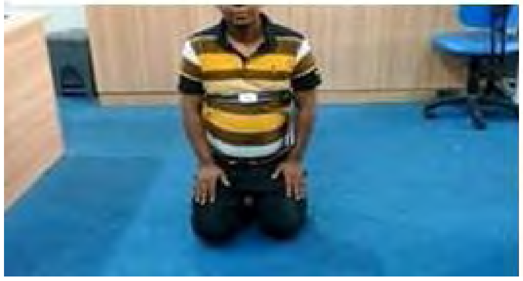

03. Left side Fall

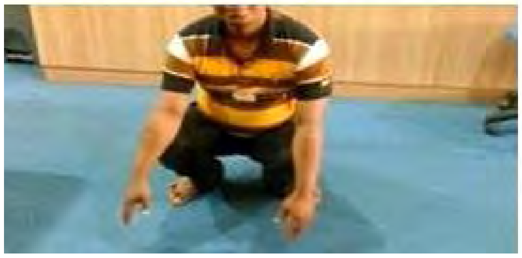

04. Right side fall

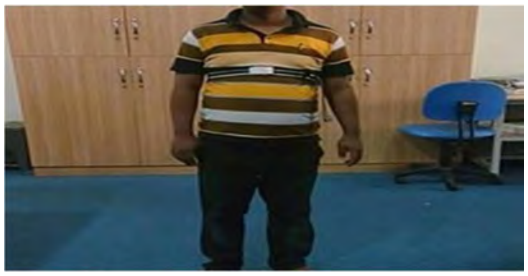

05. Go down head


06. Lying Down


07. Bend down


08. knee


09. Down


10. Up

11. Jogging

12 Walking

13. Sit down

14. Stand up

15. Idle

16. Jumping

Fig 3.8: Falls and ADL data gathering

3.9. Analysis of accelerometer signals

The acceleration signals received from the BSN are segmented into window frames where each window holds 2.56 seconds of data for the three axes. Signal features are then extracted from each window to characterize the signal being received. As we have discussed earlier, the sample rate configured for data acquisition is 50 Hz. By default, low pass filter is configured in SHIMMER sensor. The acceleration values recorded contain positive or negative accelerations as well as null in case there was no change of acceleration recorded at the same unit of time.

3.10. Feature extraction

Feature extraction is the process by which relevant characteristics or attributes are identified from the collected data. Identifying optimum number and type of features is an integral part of fall detection.

Finding the optimal feature subset is as important as selecting an appropriate algorithm. Feature extraction is also known as dimensionality reduction. Raw data are usually filled with meaningless information. By selecting only the features that best describe the input data and discarding redundant features, the size of the dataset is reduced. Feature selection is a key element in the data analysis process, and has a significant impact on subsequent stages of the learning.

Falls are events (sudden motion that occurs for a short period of time) and often result in acceleration signals with higher amplitude (Fig 3.9) than ADL. Primarily, we select the most popular features, which are called a wrapper technique. The computation and extraction of

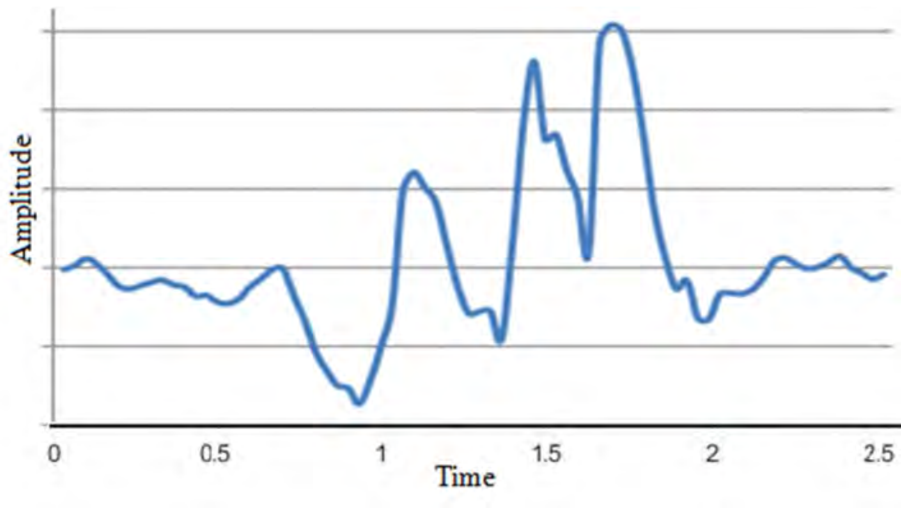


Fig 3.9: Segment of acceleration signal

significant features from the motion signals start from calculating the Mean, Standard deviation, Median, MAX, MIN, Range, Sum, PCA and FFT for each axis (gx, gy and gz) and their resultant g = (gx2+gy2+gz2)½. Since each window represents 2.56 seconds of the data and the sample rate used is equal to 50 Hz, then each window consists of 50 x 2.56 = 128 samples/rows of data. The collected data is analyzed offline. The feature components extracted are based on the computed Vector Magnitude (VM).

The statistical features that we calculated are defined below:

Mean: The mean is the average of the signal- a calculated "central" value of a set of 128 numbers.

Standard deviation: The standard deviation of a probability distribution is defined as the square root of the variance.

Median: The Median is the "middle" of a sorted list of numbers.

MAX: The maximum value of the Vm (Vmmax) over the next 2.56 second window of data.

MIN: The minimum value of the Vm (Vmmin) over the next 2.56 second window of data.

Range: Difference between Max and Min

Sum: Summation of the 128 rows

PCA: Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.

FFT: A fast Fourier transform (FFT) algorithm computes the discrete Fourier transform (DFT) of a sequence, or its inverse. Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

3.11. The 3-axis graph representation of accelerations

The graph represents the accelerations of x, y, and z axis collected from a sensor attached to the chest. The blue curve represents the x-axis, the red curve represents the y-axis, and the green curve represents the z-axis. The horizontal axis of the graph represents the time unit where each 50 unit represents one second. The vertical axis of the graph represents the acceleration reading in g unit.

Figure 3.10 illustrates changes in acceleration that occur during a forward fall. The interval of vertical lines indicates the changes in amplitudes of the accelerometer during this fall. The accelerations during falling are completely different. g_z clearly shows a negative large peak when the forward fall happened. From the sensor configure the value of $-g_z$ should maximum when forward fall occur. We get the same scenario that we are expected after sensor configuration.

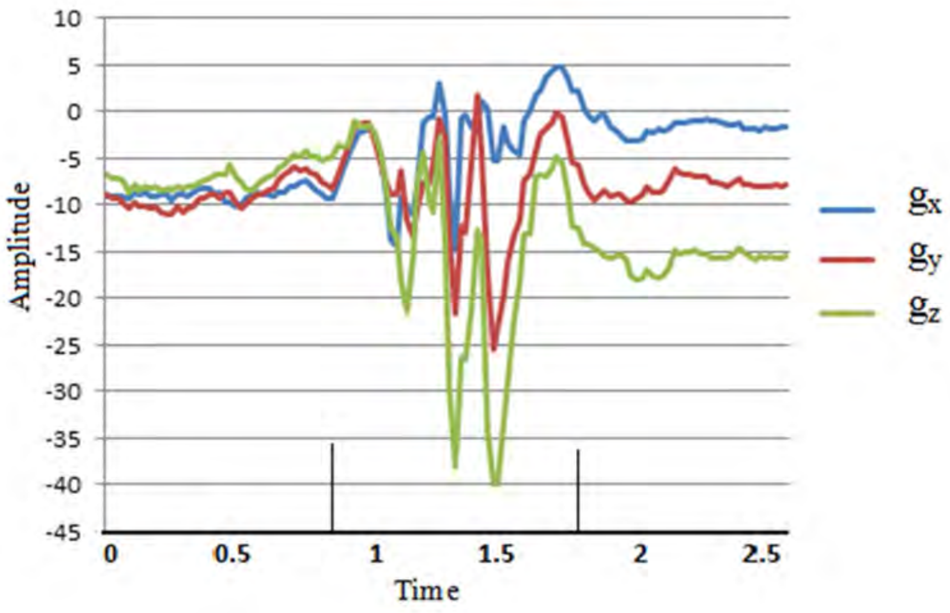


Fig 3.10: The 3-axis readings for a 'forward fall down' recorded using an accelerometer

Figure 3.11 illustrates changes in acceleration that occur during a backward fall. The interval of vertical lines indicates the changes in amplitudes the accelerometer during this fall. g_z clearly shows a positive large peak when the forward fall happened. From the sensor configure the value of g_z should maximum when backward fall occur.

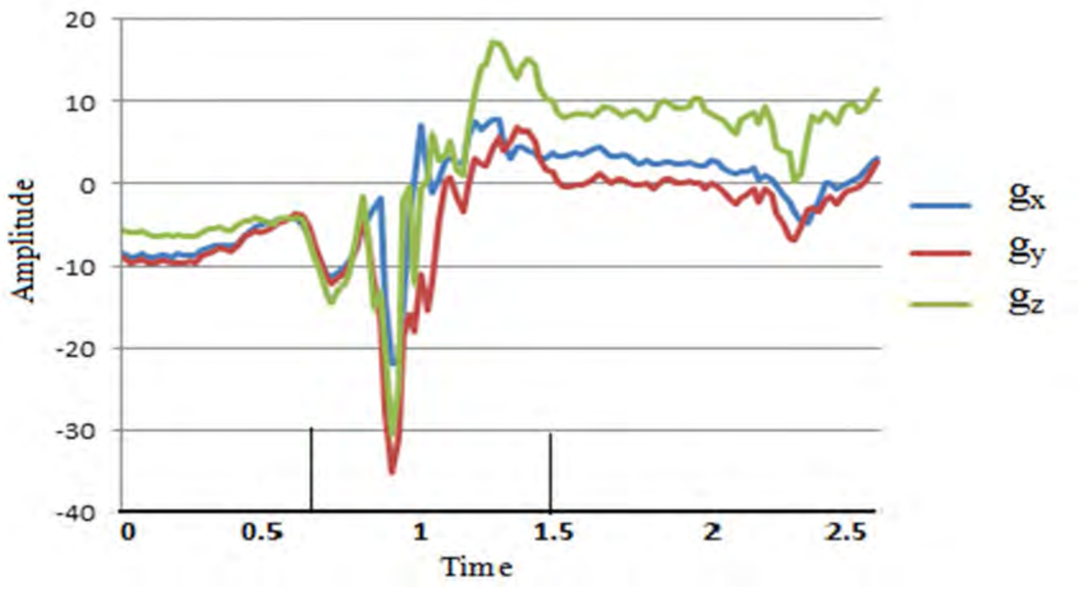


Fig 3.11: The 3-axis readings for a 'backward fall down' recorded using an accelerometer

Figure 3.12 illustrates changes in acceleration that occur during a left fall. The interval of

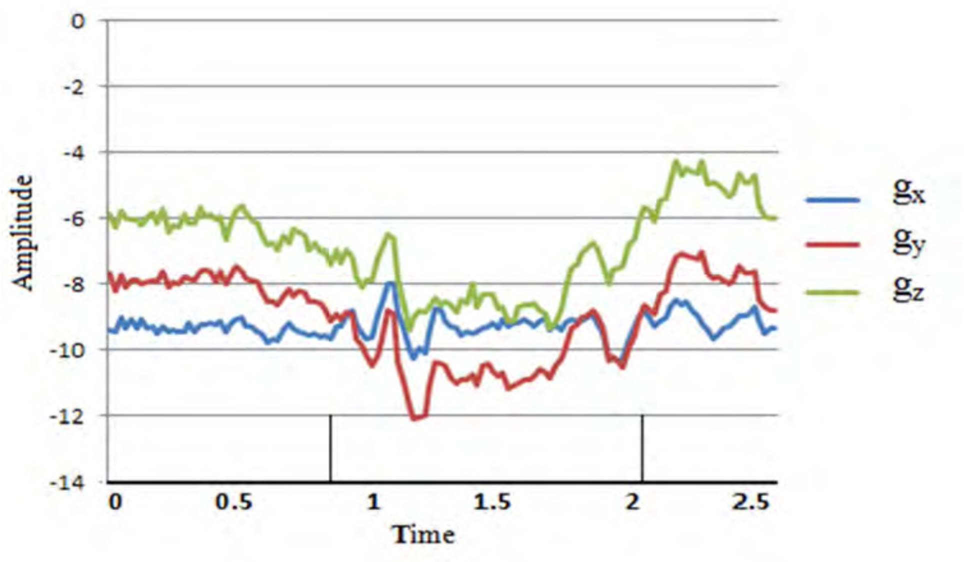


Fig 3.12: The 3-axis readings for a 'left fall down' recorded using an accelerometer.

vertical lines indicates the changes of amplitudes of the accelerometer during this fall. g_y clearly shows a negative large peak when the forward fall happened. From the sensor configure the value of $-g_y$ should maximum when left fall occur.

Figure 3.13 illustrates changes in acceleration that occur during a right fall. The interval of vertical lines indicates the changes in amplitudes of the accelerometer during this fall. g_y clearly shows a positive large peak when the forward fall happened. From the sensor configure the value of $-g_y$ should maximum when left fall occur.

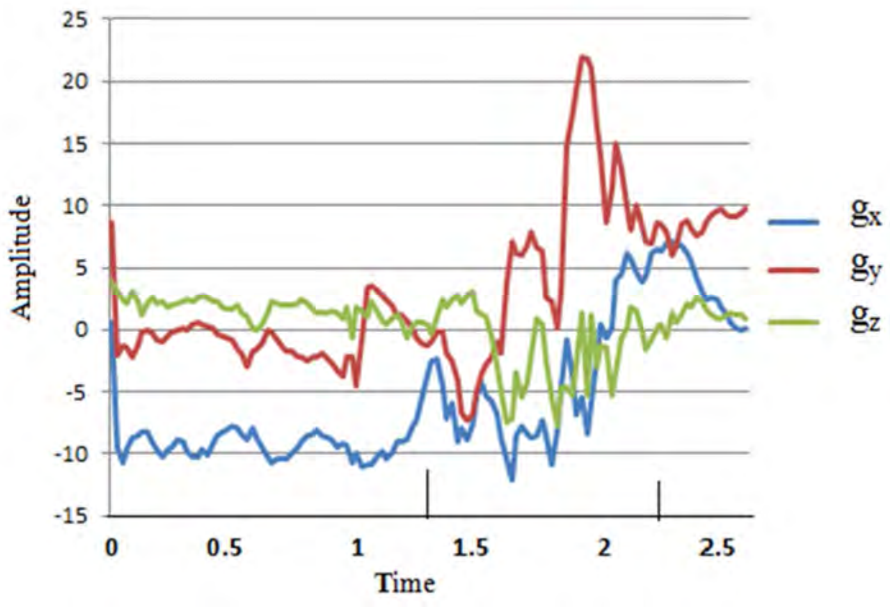


Fig 3.13: The 3-axis readings for a ‘right fall down’ recorded using an accelerometer.

Figure 3.14 illustrates changes in acceleration that occur during activities of daily living (Idle)

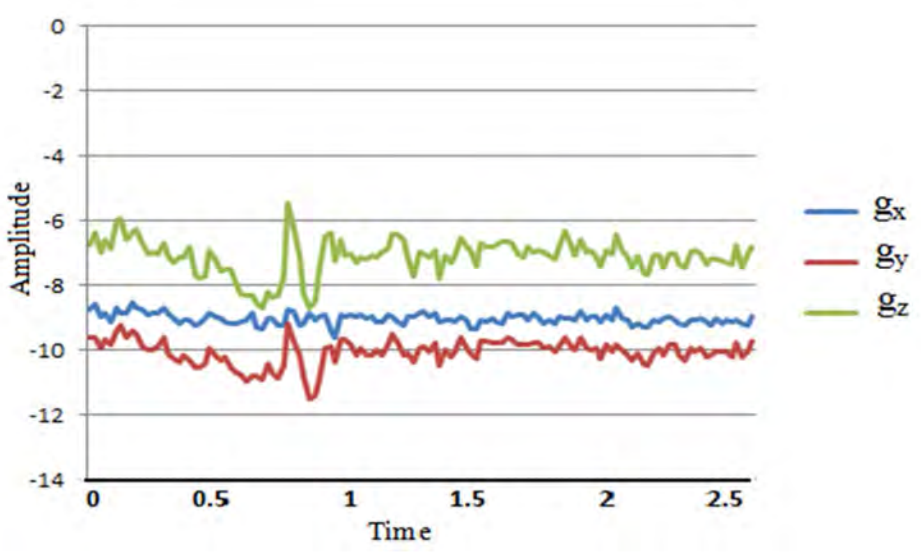


Fig 3.14: The 3-axis readings for an ‘idle’ activity recorded using an accelerometer.

Figure 3.15 illustrates changes in acceleration that occur when activities of daily living (Lying Down). The interval of vertical lines indicates that amplitude changes of accelerometer during Lying Down.

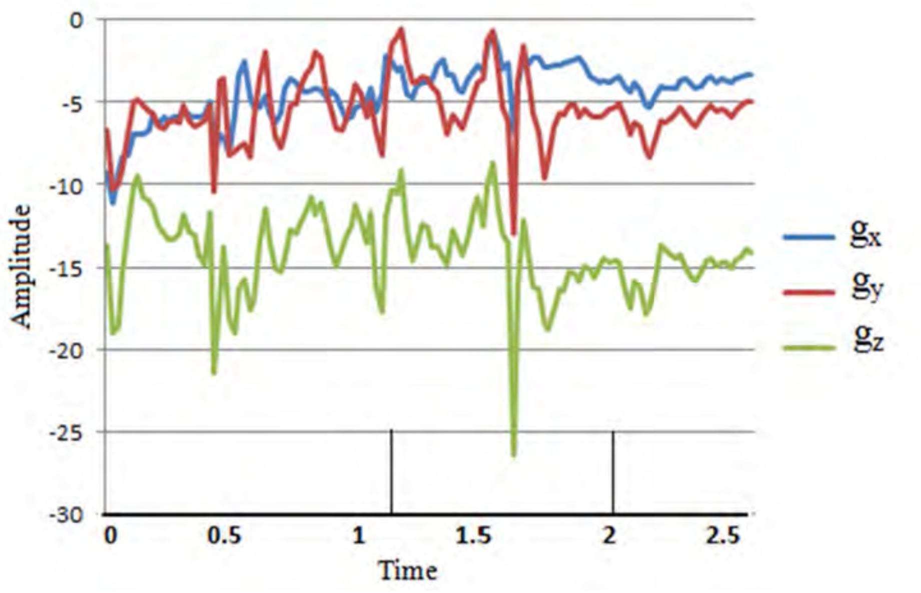


Fig 3.15: The 3-axis readings for a ‘lying down’ activity recorded using an accelerometer.

Figure 3.16 illustrates changes in acceleration that occur when activities of daily living (Sit down). The interval of vertical lines indicates that amplitude changes of accelerometer during sit down.

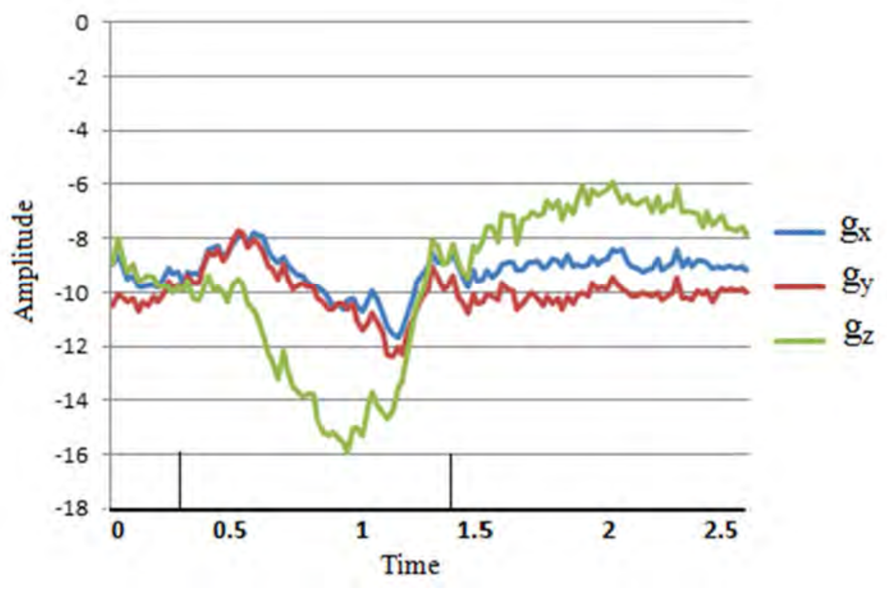


Fig 3.16: The 3-axis readings for a 'sit down' activity recorded using an accelerometer.

Figure 3.17 illustrates changes in acceleration that occur when activities of daily living (Bend

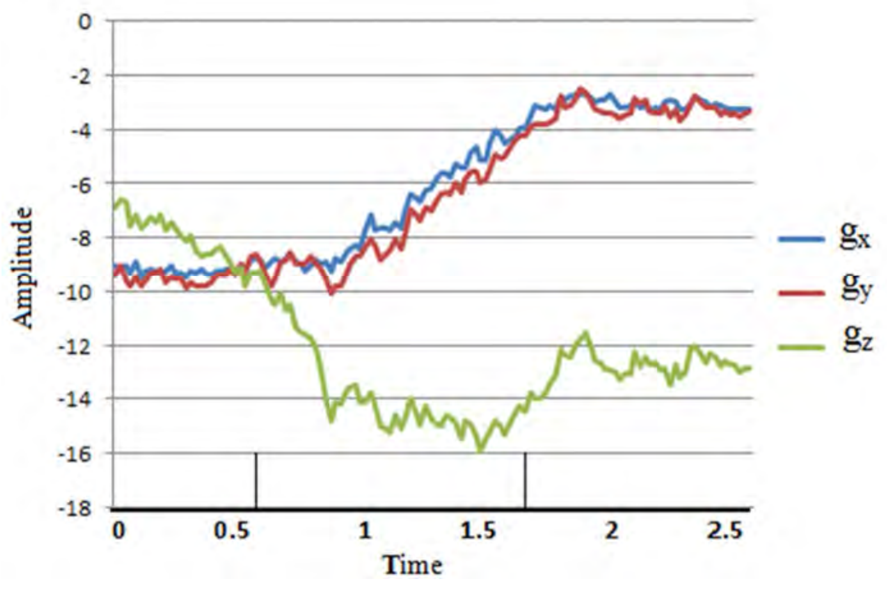


Fig 3.17: The 3-axis readings for a 'bend down' activity recorded using an accelerometer.

down). The interval of vertical lines indicates that amplitude changes of accelerometer during bend down.

Figure 3.18 illustrates changes in acceleration that occur when activities of daily living (Full Down). The interval of vertical lines indicates that amplitude changes of accelerometer during full down.

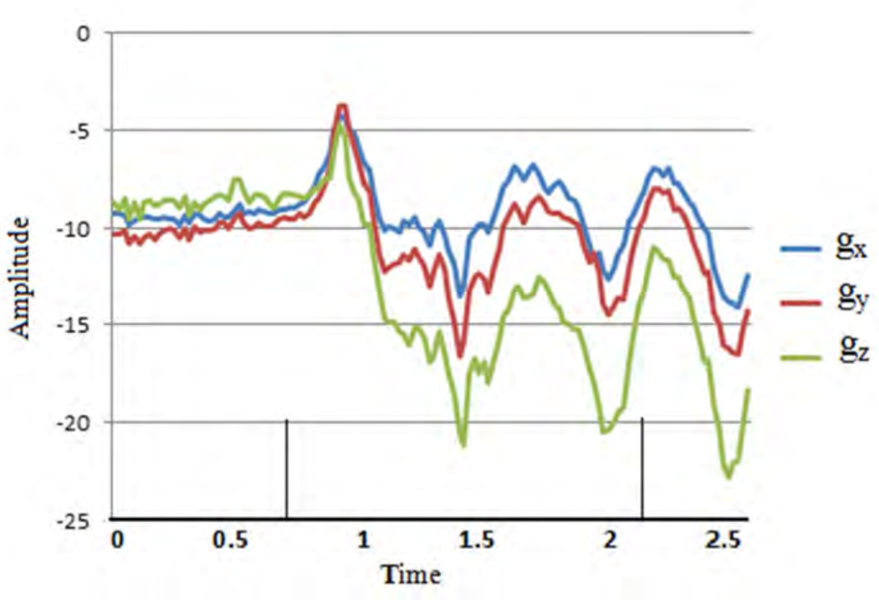


Fig 3.18: The 3-axis readings for a ‘full down’ activity recorded using an accelerometer.

Figure 3.19 illustrates changes in acceleration that occur when activities of daily living (Go down head). The interval of vertical lines indicates that amplitude changes of accelerometer during Go down head.

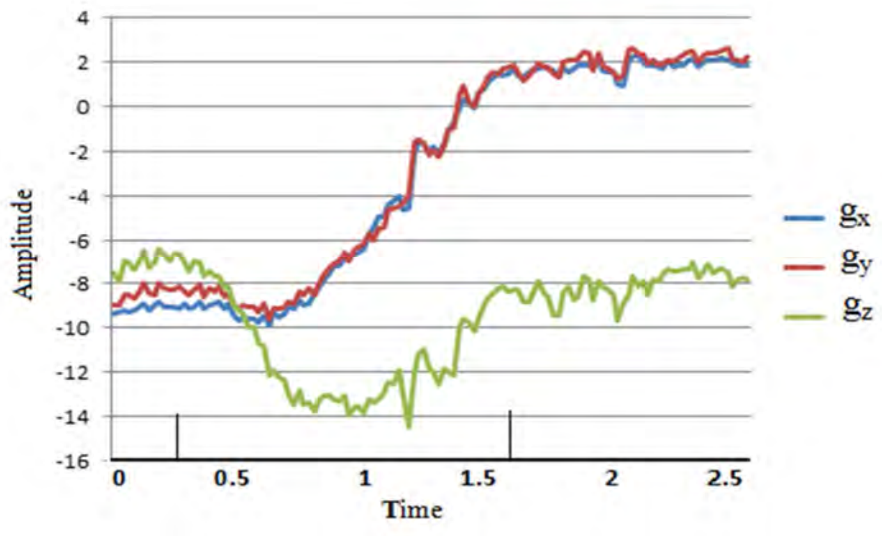


Fig 3.19: The 3-axis readings for a 'go head down' activity recorded using an accelerometer.

Figure 3.20 illustrates changes in acceleration that occur when activities of daily living (knee). The interval of vertical lines indicates that amplitude changes of accelerometer during knee.

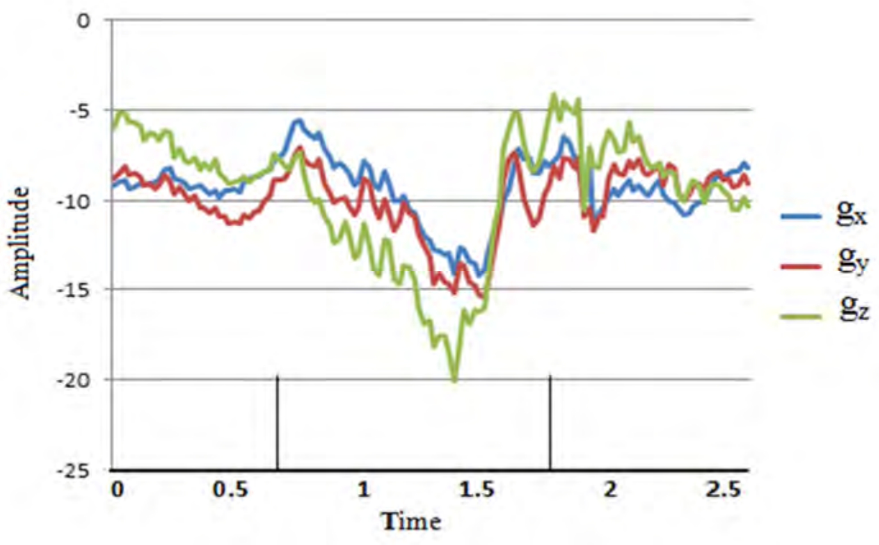


Fig 3.20: The 3-axis readings for a 'knee' activity recorded using an accelerometer.

Figure 3.21 illustrates changes in acceleration that occur when activities of daily living (Walking).

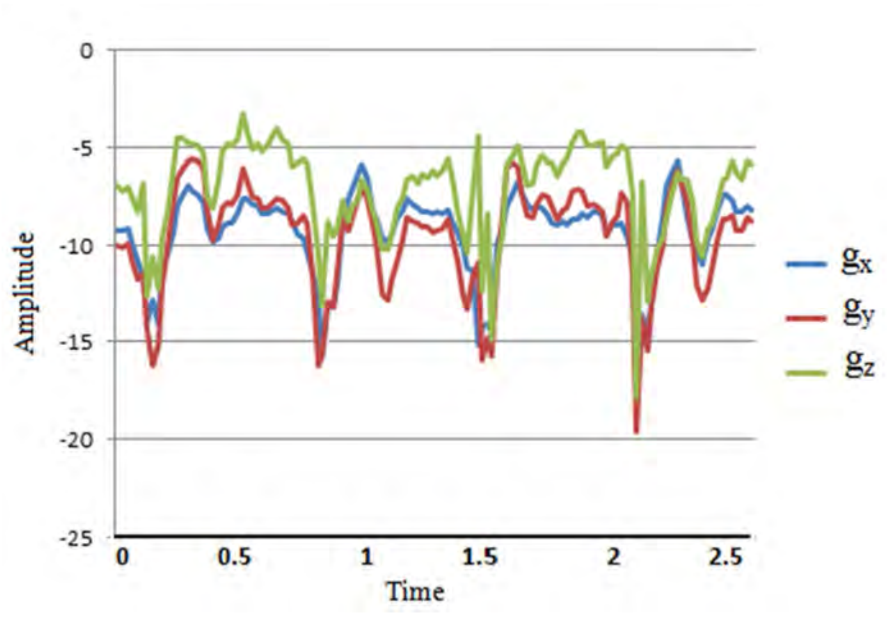


Fig 3.21: The 3-axis readings for a ‘walking’ activity recorded using an accelerometer

Figure 3.22 illustrates changes in acceleration that occur when activities of daily living (Jogging).

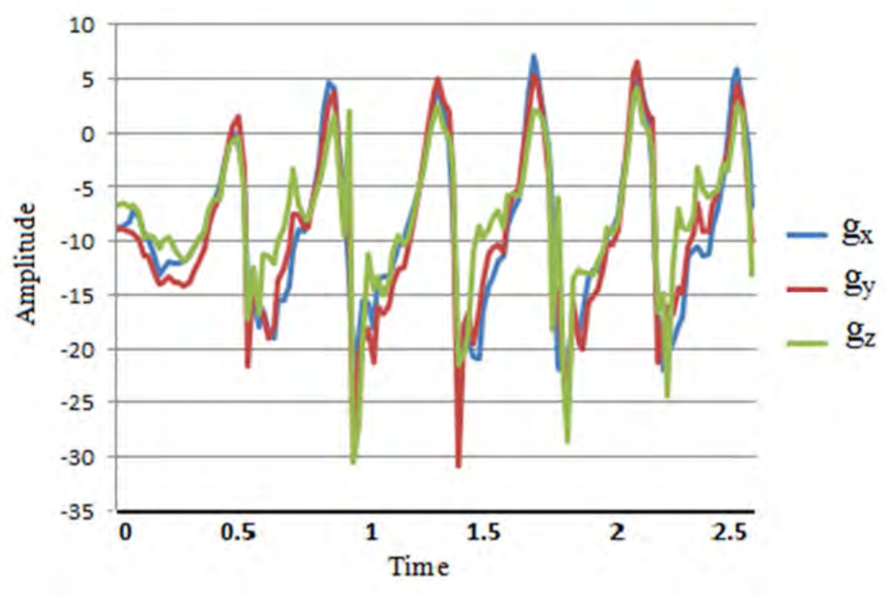


Fig 3.22: The 3-axis readings for a ‘jogging’ activity recorded using an accelerometer

Figure 3.23 illustrates changes in acceleration that occur when activities of daily living (Stand up).The interval of vertical lines indicates that amplitude changes of accelerometer during up.

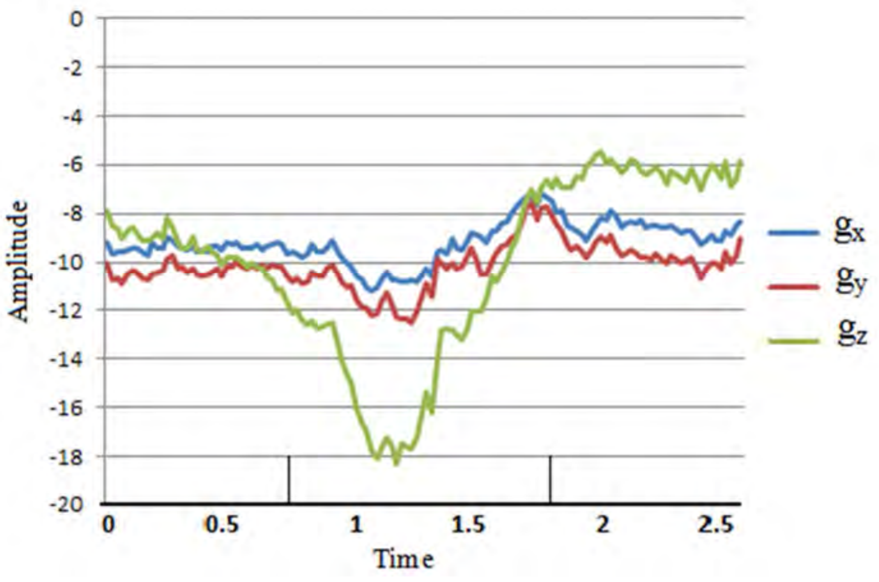


Fig 3.23: The 3-axis readings for a ‘stand up’ activity recorded using an accelerometer

Figure 3.24 illustrates changes in acceleration that occur when activities of daily living (Up).

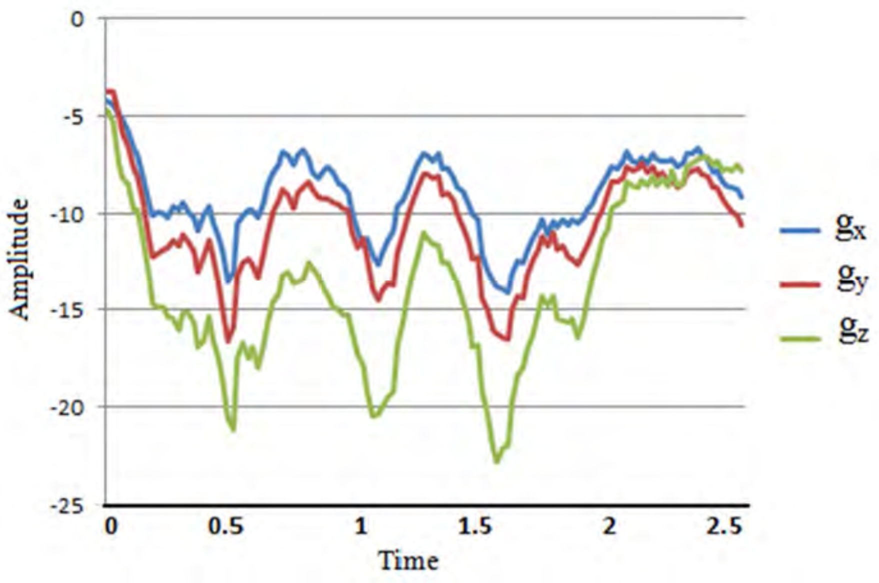


Fig 3.24: The 3-axis readings for an ‘up’ activity recorded using an accelerometer.

Figure 3.25 illustrates changes in acceleration that occur when activities of daily living (Jumping). The interval of vertical lines indicates that amplitude changes of accelerometer during Jump.

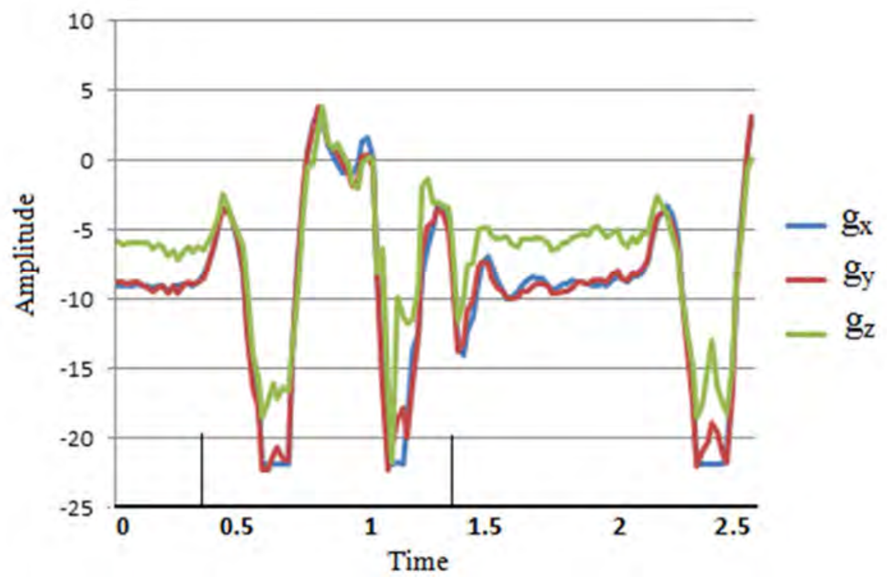
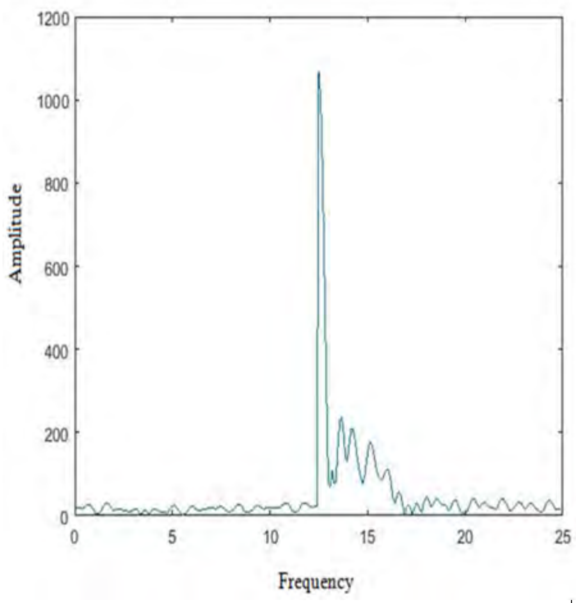
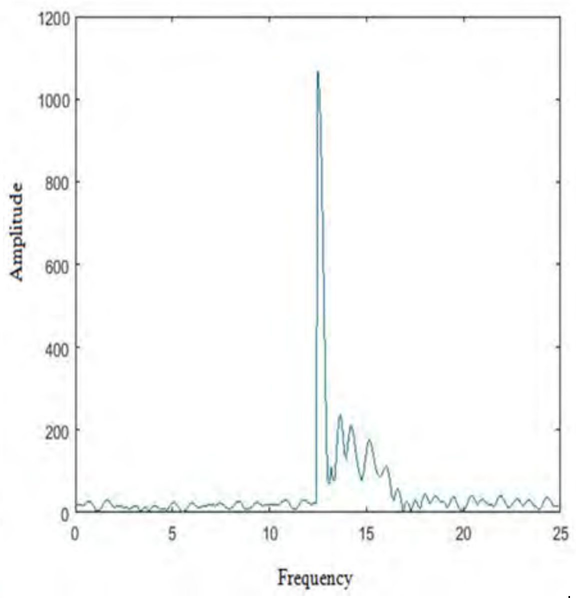
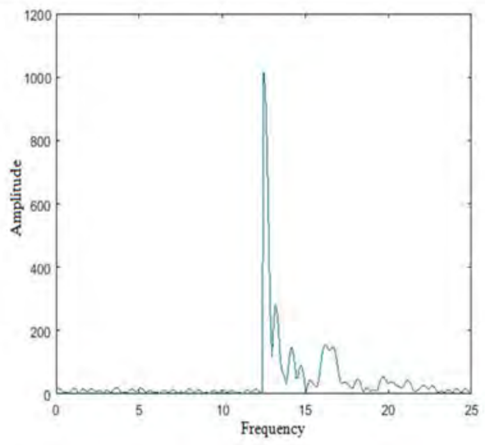
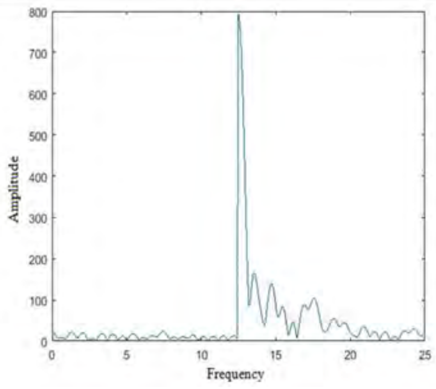


Fig 3.25: The 3-axis readings for a ‘jumping’ activity recorded using an accelerometer.

3.12. Frequency Domains

The amplitude spectrum of g obtained by FFT analysis is shown in Fig. 3.26 to 3.39.

 Fig 3.26: FFT of acceleration signal (Forward fall)	 Fig 3.27: FFT of acceleration signal (Backward fall)
 Fig 3.28: FFT of acceleration signal (Left side Fall)	 Fig 3.29: FFT of acceleration signal (Right side fall)

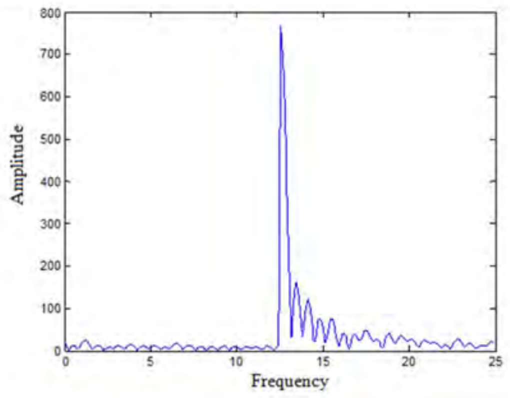


Fig 3.30: FFT of acceleration signal (Lying Down)

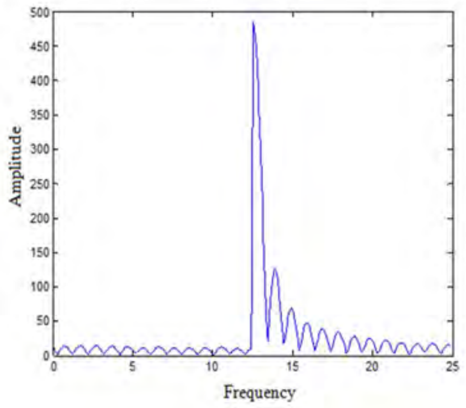


Fig 3.31: FFT of acceleration signal (Sit down)

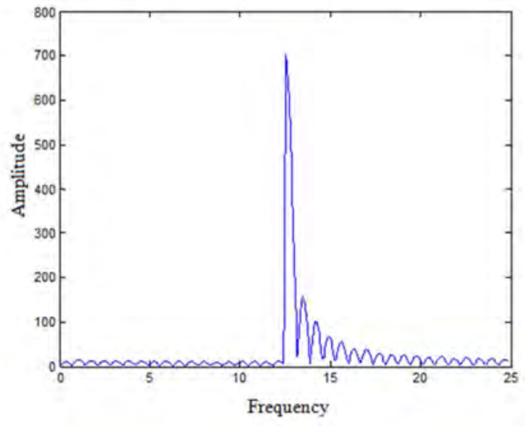


Fig 3.32: FFT of acceleration signal (Bend down)

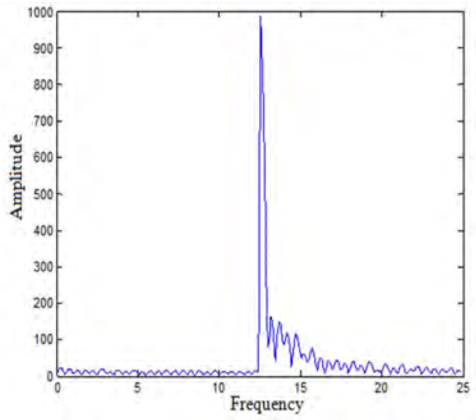


Fig 3.33: FFT of acceleration signal (Down)

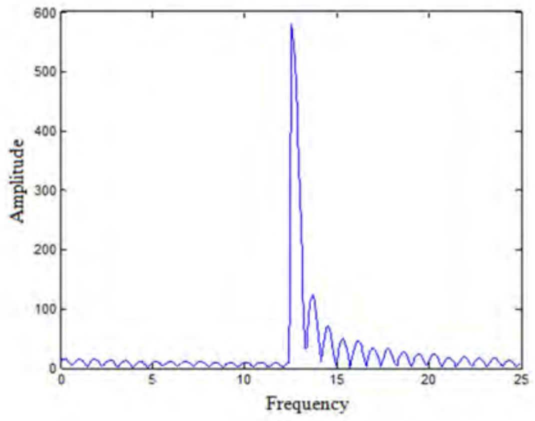


Fig 3.34: FFT of acceleration signal (Go down head)

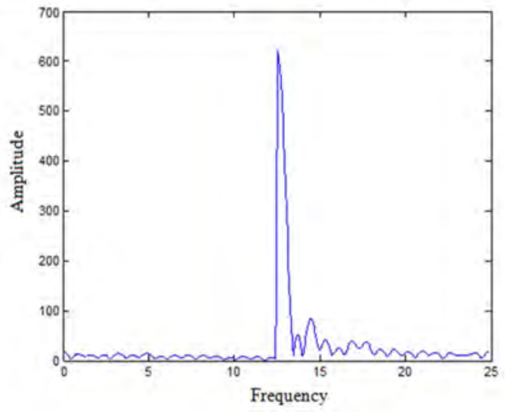
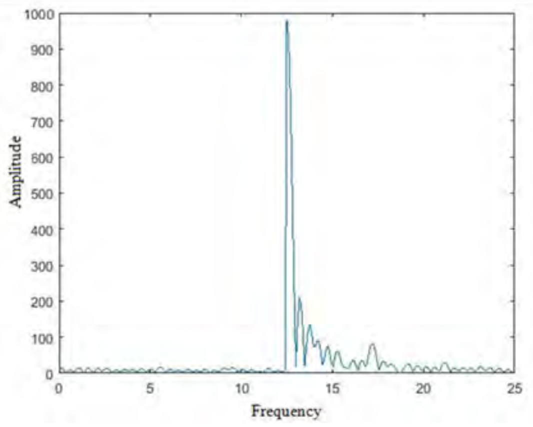
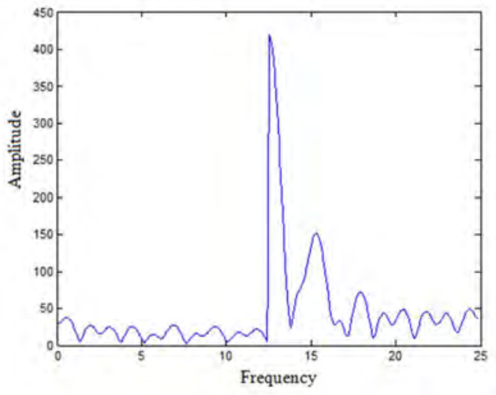
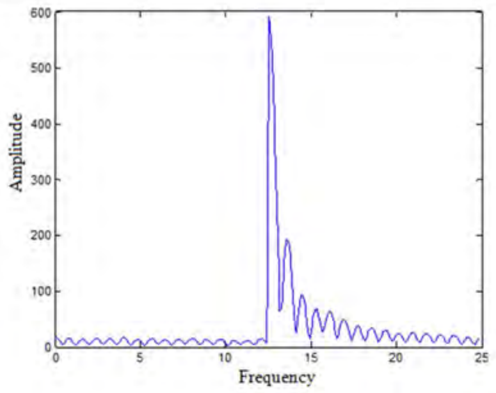
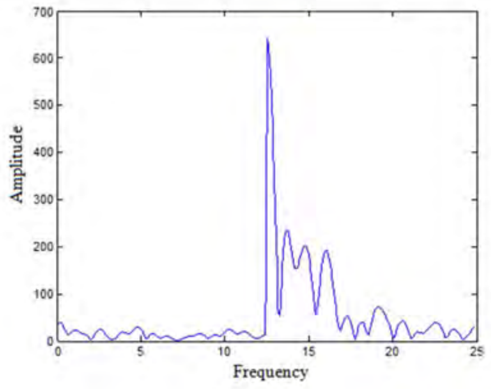


Fig 3.35: FFT of acceleration signal (knee)

 Fig 3.36: FFT of acceleration signal (Walk)	 Fig 3.37: FFT of acceleration signal (Jogging)
 Fig 3.38: FFT of acceleration signal (Up)	 Fig 3.39: FFT of acceleration signal (Jumping)

From the FFT analysis, we have not found any significant peck to distinguish from fall and ADL.

3.13. Calculation of Statistical Features

Four types of falls and twelve types of activities were recorded in dat file. We calculated every single record using Mean, Standard deviation, Median, MAX, MIN, Range, Sum, PCA and FFT function for each axis (gx, gy and gz) and their resultant g = (gx2+gy2+gz2)½. Also we recorded the upper and lower value of each function for each record. Recorded result is shown in Table 3.2, Table 3.3, Table 3.4 and Table 3.5.

Table 3.2: Calculated features (g)

Features Selection Table

g

		Mean		Median		Std dev		MAX		MIN		Range		sum		Count		FFT	
		Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
01	Forward fall	11.377	10.06	10.33	9.149	6.35484	4.458	34.33	21.249	4.319	0.287	32.75	19.98	1410	598.4	131	56	1400	600
02	Backward fall	11.867	10.28	10.64	8.698	6.41074	3.829	34.62	25.2929	4.83	0.884	32.51	21.28	1301	583.4	123	51	1200	570
03	Left side Fall	11.271	10	10.26	9.131	4.75733	3.244	29.04	21.0603	5.508	2.277	23.88	16.19	1114	676.3	111	60	1100	670
04	Right side fall	12.296	10.21	11.13	9.17	5.24942	4.319	26.82	23.8978	5.86	3.371	23.24	18.21	757.1	479.5	73	39	750	380
05	Idle	9.64	9.568	9.641	9.544	0.20961	0.128	10.15	9.84566	9.34	9.021	1.134	0.587	771.2	765.5	80	80	770	760
06	Lying Down	11.062	10.2	10.89	10.18	2.33795	0.899	19.23	12.4952	8.384	4.427	13.42	4.565	1294	497.8	122	45	1300	500
07	Sit down	10.169	9.368	9.689	9.171	1.15	0.431	12.95	11.0221	9.255	8.009	3.961	2.323	677.1	374.7	72	40	680	380
08	Bend down	9.7635	9.591	9.722	9.283	0.87691	0.569	11.72	10.8548	8.783	8.285	3.44	2.302	712.7	594.6	73	62	700	590
09	Down	9.9157	9.61	9.735	9.324	2.17455	1.617	16.7	15.0294	7.095	4.363	10.67	8.279	1230	682.3	124	71	1200	590
10	Go down head	9.7534	9.494	9.781	9.4	0.56952	0.389	11.45	10.7487	8.466	8.216	2.986	2.353	1631	579.2	171	61	1600	580
11	knee	10.028	9.817	9.625	9.306	2.3501	1.644	14.97	13.6786	7.108	5.763	9.212	6.571	847.5	621.7	86	62	830	600
12	Walk	10.074	9.584	9.275	9.006	2.68716	2.17	18.43	15.8321	6.384	5.165	12.5	9.673	720.3	562.8	74	57	700	580
13	Jogging	11.643	10.47	11.38	9.46	6.75297	5.83	25.45	22.2236	1.964	1.055	24.3	20.61	477.3	418.9	42	40	480	400
14	Stand up	10.036	9.372	9.867	9.195	1.17522	0.811	12.72	11.7804	8.617	7.652	5.039	3.164	581	500.3	62	50	590	500
15	Up	9.7677	9.397	9.69	9.058	2.22085	1.728	15.82	14.9994	7.62	6.584	8.698	7.57	1065	573.2	109	61	1030	580
16	Jumping	10.155	9.515	8.456	7.275	8.0393	6.887	25.5	22.7507	1.276	0.272	25.23	21.78	838.8	511.7	84	53	830	500

Table 3.3: Calculated features (g_x)

	g_x																
		Mean		**Median**		**Standard deviation**		**MAX**		**MIN**		**Range**		**sum**		**Count**	
		Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
01	Forward fall	-2.216	-5.374	-0.89	-5.186	6.38353	4.084	15.19	4.11	-11.1	-22.16	37.06	19.06	-124	-474.2	131	56
02	Backward fall	0.2459	-5.934	3.188	-6.041	8.09517	5.607	19.87	6.399	-21.9	-22	41.76	28.33	18.44	-456.5	123	51
03	Left side Fall	-1.075	-6.111	0.817	-7.648	6.18867	4.313	11.61	0.801	-13	-22.03	28.38	19.87	-64.5	-551.1	111	60
04	Right side fall	-1.171	-6.539	-2.78	-6.499	6.44517	3.815	9.912	-0.241	-11.8	-15.48	22.83	15.18	-45.7	-316.7	73	30
05	Idle	-9.068	-9.542	-9.04	-9.536	0.2462	0.128	-8.43	-9.075	-9.35	-10.02	1.216	0.641	-725	-763.3	80	80
06	Lying Down	11.062	-6.544	10.89	-7.467	4.29028	0.899	19.23	1.738	8.384	-16.49	18.23	4.565	1294	-798.4	122	45
07	Sit down	-8.829	-9.724	-8.76	-9.466	0.71439	0.271	-7.78	-8.97	-9.63	-11.47	2.861	1.288	-361	-635.7	72	40
08	Bend down	-7.758	-8.169	-8.5	-8.736	1.70003	0.999	-4.04	-5.29	-9.17	-9.844	5.42	3.881	-500	-587.7	73	62
09	Down	-8.826	-9.244	-8.67	-9.071	1.92184	1.211	-4.22	-6.985	-13.3	-14.81	9.288	6.413	-627	-1146	124	71
10	Go down head	-5.213	-8.207	-6.77	-8.911	4.23593	1.73	1.743	-2.941	-9.5	-10.08	11.62	7.12	-412	-1403	171	61
11	knee	-9.31	-9.657	-8.96	-9.319	2.40455	1.672	-5.32	-6.489	-13.5	-14.36	8.645	7.122	-589	-823.4	86	62
12	Walk	-9.076	-9.601	-8.52	-8.794	2.49421	1.954	-5.06	-6.161	-14.7	-17.76	12.09	8.567	-539	-688.9	74	57
13	Jogging	-9.066	-9.592	-8.82	-10.85	8.52171	6.553	7.102	3.568	-21.9	-22.05	27.87	25.62	-368	-402.9	42	40
14	Stand up	-8.779	-9.603	-8.57	-9.378	1.06161	0.575	-7.53	-8.387	-10.7	-12.41	4.487	2.555	-480	-545.4	62	50
15	Up	-8.954	-9.142	-8.73	-9.086	1.90582	1.434	-6.41	-7.454	-13.4	-14.22	7.471	6.319	-546	-987.7	109	61
16	Jumping	-8.315	-8.791	-6.79	-7.984	8.97932	7.67	8.035	3.378	-22	-22.01	30	25.34	-457	-733.7	84	53

Table 3.4: Calculated features (g_y)

g_y

		Mean		**Median**		**Standard deviation**		**MAX**		**MIN**		**Range**		**sum**		**Count**	
		Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
01	Forward fall	-2.055	-5.42	-0.54	-6.919	5.56881	4.049	16.9	2.039	-12.9	-20.43	34.42	16.09	-123	-510.2	131	56
02	Backward fall	6.7083	-4.239	5.627	-4.373	7.22797	2.573	22.33	1.583	-1.01	-19.59	32.68	13.93	462.9	-360.3	123	51
03	Left side Fall	-4.326	-7.999	-1.78	-8.621	7.27736	4.925	8.816	2.161	-16.8	-20.34	29.16	18.99	-322	-497.9	111	60
04	Right side fall	8.4636	3.745	8.795	0.752	8.92874	6.792	22.17	21.89	-4.72	-8.542	30.43	26.82	330.1	170.7	73	30
05	Idle	-0.394	-1.129	-0.36	-1.048	0.38572	0.143	-0.07	-0.699	-0.95	-2.611	2.202	0.622	-31.6	-90.33	80	80
06	Lying Down	1.105	-2.948	0.972	-3.232	3.35498	1.211	7.056	3.066	-1.53	-10.71	17.61	5.739	108.3	-263.5	122	45
07	Sit down	-0.473	-1.075	-0.43	-1.116	0.4381	0.233	0.221	-0.487	-1.32	-1.67	1.726	1.122	-24.1	-77.37	72	40
08	Bend down	-0.462	-0.737	-0.51	-0.769	0.36903	0.217	0.37	-0.079	-1.04	-1.217	1.532	1.089	-33.3	-53.8	73	62
09	Down	-1.141	-1.625	-1.13	-1.643	0.80086	0.397	0.7	-0.581	-2.13	-3.159	3.859	1.763	-87.8	-201.5	124	71
10	Go down head	0.4887	0.23	0.484	0.236	0.37937	0.322	1.638	1.002	-0.18	-1.28	2.437	1.444	47.37	20.47	171	61
11	knee	-0.345	-1.189	-0.32	-1.256	2.17934	0.598	4.63	1.078	-2.01	-7.293	11.92	3.092	-26.9	-78.38	86	62
12	Walk	-0.092	-0.421	-0.11	-0.535	1.41632	1.081	4.846	1.807	-2.3	-3.59	7.559	4.498	-5.72	-26.95	74	57
13	Jogging	0.1844	-0.645	0.275	-0.742	2.88615	2.112	6.537	4.603	-4.38	-9.035	13.47	9.287	7.561	-27.07	42	40
14	Stand up	-0.936	-1.198	-0.96	-1.275	0.4783	0.22	0.537	-0.672	-1.51	-1.82	2.357	0.837	-53.7	-73.11	62	50
15	Up	-0.922	-1.327	-0.87	-1.372	0.68229	0.372	0.613	-0.515	-2.4	-2.724	3.337	1.948	-80.9	-100.4	109	61
16	Jumping	0.188	-0.645	0.193	-0.696	1.25948	0.72	4.49	1.469	-1.47	-5.595	9.167	3.269	12.03	-41.91	84	53

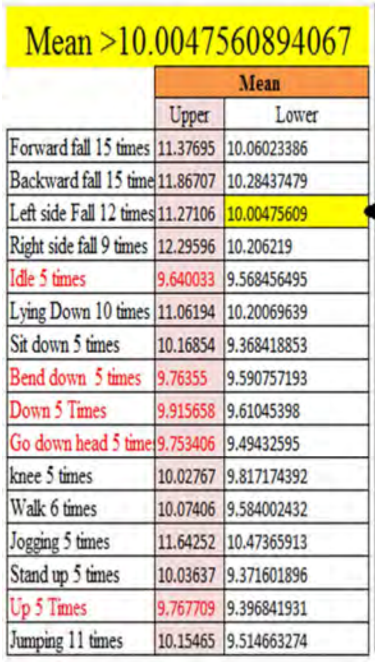
Table 3.5: Calculated features (g_z)

g_z

		Mean		Median		Standard deviation		MAX		MIN		Range		sum		Count	
		Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower	Upper	Lower
01	Forward fall	-2.804	-6.223	-0.61	-7.146	7.17958	4.886	16.08	1.284	-16.3	-17.07	32.45	18.08	-227	-667.5	131	56
02	Backward fall	7.4385	1.978	9.171	1.154	6.41005	2.895	25.42	9.68	-0.61	-12.28	37.07	13.7	763.1	134.5	123	51
03	Left side Fall	2.5262	-1.43	3.103	-0.01	4.50593	2.578	11.66	3.431	-5.31	-16.24	20.61	14.74	209.7	-94.4	111	60
04	Right side fall	1.8959	-2.39	1.108	-2.215	3.58025	2.725	11.08	3.065	-5.61	-9.38	18.35	10.82	119.4	-110	73	30
05	Idle	2.9713	1.269	2.964	1.278	0.44683	0.144	3.708	1.923	2.683	0.644	2.057	0.758	237.7	101.5	80	80
06	Lying Down	-4.983	-10.11	-3.57	-10.15	4.72263	0.929	5.009	-7.529	-11.8	-16.46	17.44	4.938	-418	-1000	122	45
07	Sit down	0.1899	-2.051	0.793	-1.792	3.06568	2.04	3.916	2.022	-4.33	-6.534	9.239	6.708	13.68	-98.46	72	40
08	Bend down	-2.504	-3.962	-1.93	-4.392	4.4929	3.758	3.713	2.017	-9.48	-10.44	14.15	12.31	-155	-285.2	73	62
09	Down	-2.202	-3.218	-2.88	-3.869	3.0274	1.694	2.824	1.998	-6.41	-8.293	11.12	8.659	-156	-389.3	124	71
10	Go down head	0.7346	-5.554	3.233	-6.99	4.89896	4.162	4.604	1.617	-10.6	-11.21	15.17	12.82	125.6	-438.7	171	61
11	knee	-0.092	-1.262	0.426	-1.615	2.73197	1.653	5.188	3.353	-3.19	-5.344	10.31	6.541	-7.92	-81.77	86	62
12	Walk	2.7867	2.291	2.942	2.293	1.69297	1.252	7.874	5.676	-0.19	-1.054	8.717	5.863	178.3	135	74	57
13	Jogging	1.5676	1.038	1.649	0.788	3.79407	2.844	10.93	6.325	-3.19	-8.445	18.27	12.12	62.7	42.54	42	40
14	Stand up	1.2922	0.093	2.362	0.659	3.20862	2.549	4.853	3.155	-4.67	-6.327	10.65	7.828	74.95	5.697	62	50
15	Up	-0.621	-2.735	-0.54	-3.417	3.00662	2.391	3.559	3.236	-6.04	-7.691	11.13	9.276	-38.5	-298.1	109	61
16	Jumping	2.1339	0.89	1.975	0.451	3.1317	2.38	13.2	8.887	-2.6	-6.643	17.69	12.1	148.9	74.73	84	53

3.14. Data Analysis / Feature selection

Feature selection is the process of selecting a subset of relevant features for use in model construction. It eliminates the redundant and meaningless values without losing significant information. Irrelevant input features induce greater computational cost. Figure 3.39 and Figure 3.40, show that the Mean and Standard deviation features are the top 2 most important features in the dataset and median, max, min, range, sum, count features are the less important. Among the falls, the least value of mean is 10.0047560. If we choose, mean 10.0047560 is threshold than idle, bend down, down, go down head and up can be discarded (Figure 3.40). Similarly if we choose, Standard deviation 3.244006244 is threshold than lying down , sit down, knee, walking , stand up can discard (Figure 3.41).



Mean >10.0047560894067

	Mean	
	Upper	Lower
Forward fall 15 times	11.37695	10.06023386
Backward fall 15 time	11.86707	10.28437479
Left side Fall 12 times	11.27106	10.00475609
Right side fall 9 times	12.29596	10.206219
Idle 5 times	9.640033	9.568456495
Lying Down 10 times	11.06194	10.20069639
Sit down 5 times	10.16854	9.368418853
Bend down 5 times	9.76355	9.590757193
Down 5 Times	9.915658	9.61045398
Go down head 5 time	9.753406	9.49432595
knee 5 times	10.02767	9.817174392
Walk 6 times	10.07406	9.584002432
Jogging 5 times	11.64252	10.47365913
Stand up 5 times	10.03637	9.371601896
Up 5 Times	9.767709	9.396841931
Jumping 11 times	10.15465	9.514663274

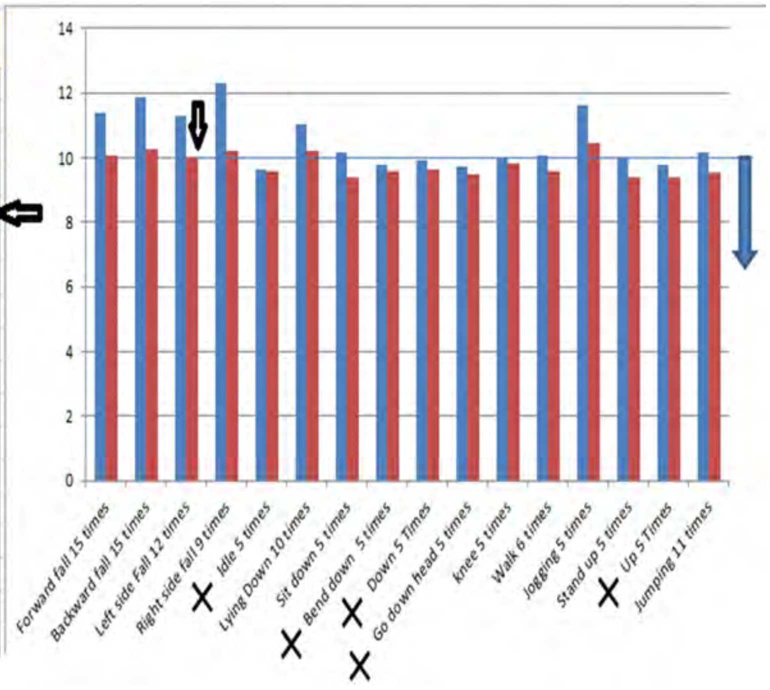
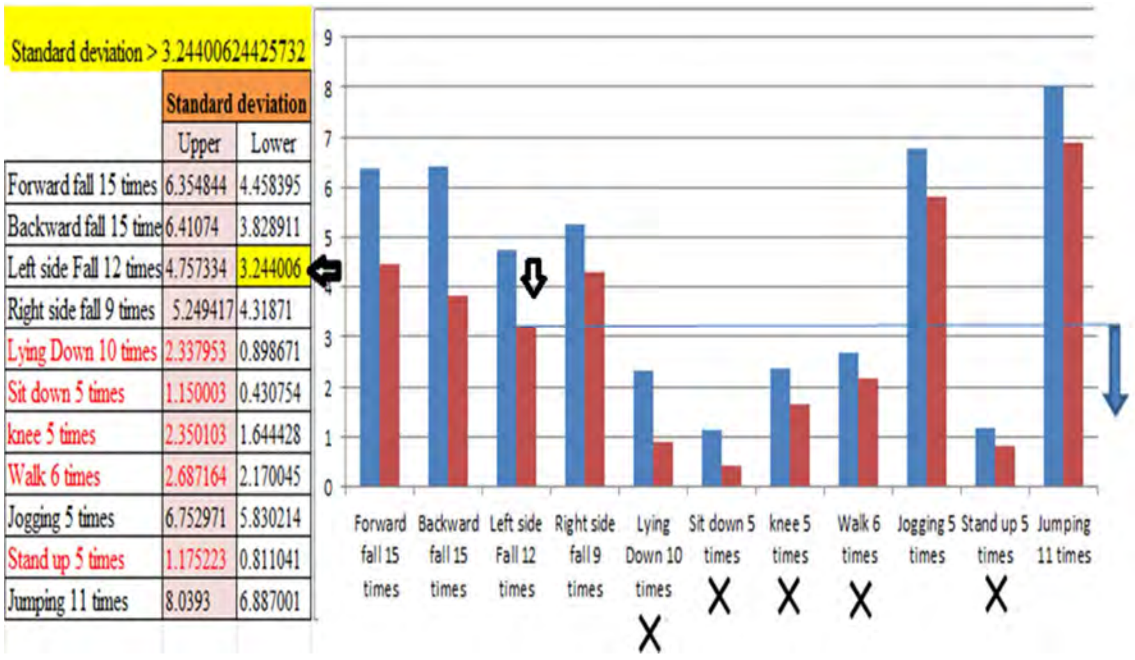


Fig 3.40: Significant feature- fall Mean


Standard deviation > 3.24400624425732

	Standard deviation	
	Upper	Lower
Forward fall 15 times	6.354844	4.458395
Backward fall 15 time	6.41074	3.828911
Left side Fall 12 times	4.757334	3.244006
Right side fall 9 times	5.249417	4.31871
Lying Down 10 times	2.337953	0.898671
Sit down 5 times	1.150003	0.430754
knee 5 times	2.350103	1.644428
Walk 6 times	2.687164	2.170045
Jogging 5 times	6.752971	5.830214
Stand up 5 times	1.175223	0.811041
Jumping 11 times	8.0393	6.887001

Fig 3.41: Significant feature- Standard Deviation

3.14.1. Accelerometer Axis configuration

The axes of the accelerometer were configured as shown below (with reference to the Fig 3.42)

Forward	= -Z Max
Backward	= +Z Max
Left	= -Y Max
Right	= +Y Max
Up	= +X Max
Down	= -X Max

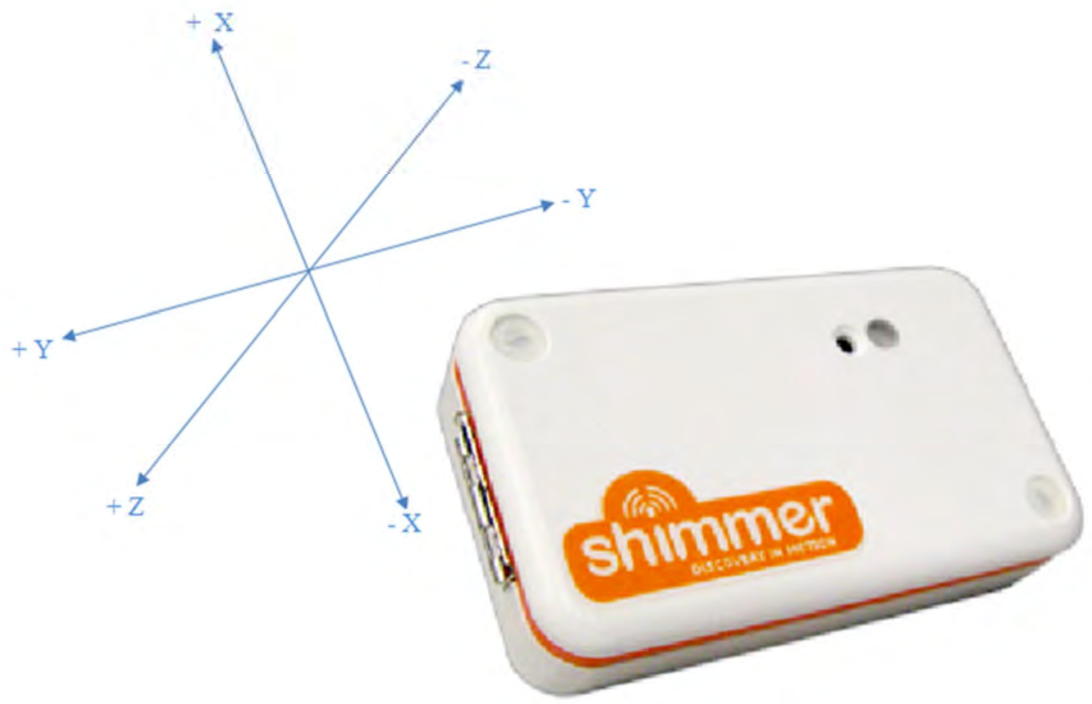


Fig 3.42: X-Y-Z coordinate system [79]

3.14.2. To Detect Direction

The mainboard of the Shimmer contained within the Shimmer casing. The mechanical design of the mainboard is such that the direction of the X, Y and Z axis of the accelerometer are forward direction - Z, backward direction + Z, left direction - Y , right direction + Y, up direction + X, down direction - X. Figure 3.43 illustrates the hardware co-ordinate system for the sensor on the Shimmer2r with mainboard. The accelerometer has a right handed coordinate system.

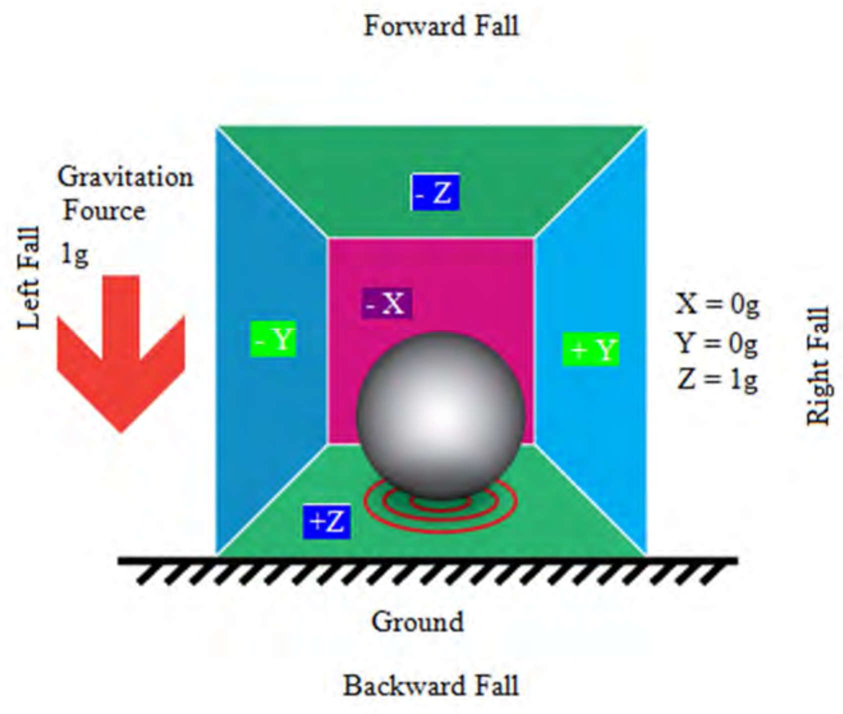


Fig 3.43: Axis value and direction [91]

Setting up the configuration, we should get direction

Forward Fall	= - Z should be max value
Backward Fall	= + Z should be max value
Left Fall	= - Y should be max value
Right Fall	= + Y should be max value
Up	= + X should be max value but no effect on Fall Direction
Down	= - X should be max value but no effect on Fall Direction

3.14.3. Direction analysis

From our data analysis, we find out some important characteristics of the feature that can be contributed to the fall direction and this outcome matches with axis value concept (Table 3.6).

Table 3.6: Direction findings

Axis Max value	**Direction**	**Conclusion**
+Z (max)	Backward Fall	Confirm
-Z (max)	Forward Fall	Confirm but Rare, -Y problem
+Y (max)	Right Fall	Confirm
-Y (max)	Left or Forward	it should be Left Fall

3.14.4. Decision

The important thing is to decide on which statistical features are significant for a classification of falls. We find the following significant parameters to detect falls with their directions based on fall characteristics: “Mean, Standard Deviation and Principal component analysis (PCA)”. The mean is calculated for x, y, and the z component of the acceleration signal of the 128 (2.56s) samples according to Eq. 3.1:

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k \text{ ---------- } 3.1$$

The standard deviation is calculated for each axis according to Eq. 3.2:

$$\sigma = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - \bar{x})^2} \text{ -- ------ } 3.2$$

Principal component analysis (PCA) is calculated for each axis according to Eq. 3.3:

$$S = \sum_{k=1}^{n}(x_k - \bar{x})(x_k - \bar{x})^T \text{ -- --- } 3.3$$

$$\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k \text{ ---- ------ } 3.4$$

3.15. Classification

We need to develop an algorithm to be able to classify directive fall by recognizing the signal patterns and matching a vector of significant statistical features with pre-learned ones. Thereafter, the resulting computation is fed to the classifier in order to detect the directive fall. We have used “classificationLearner” [MATLAB R2016a] tool to train and to test. Two learning classifiers namely support vector machines (SVM) and k-Nearest Neighbors (KNN) have been used to classify directive fall.

3.16. Training and Testing data

Machine learning based algorithms require an optimum number of subjects to simulate falls and ADL in order to provide a good performance [76]. On the other hand, training with more subjects than needed do not necessarily lead to improved performance. Sometimes it can affect the performance negatively [76]. It is necessary that the optimum number of subjects required for training is identified during algorithm implementation. In this thesis work, individual learning classifier was used to learn to distinguish among falls and ADLs. Whole data are divided in two parts randomly with 50% overlap .One for training (66%) and another for testing (34%). Summary of the methodology is provided in Table 3.7.

Table 3.7: Accelerometer, features and algorithm

Sensor	**Accelerometer**
Segmentation	Sample frequency 50 Hz
Window Size	2.56 seconds
Window Overlap	50 %
Features	Mean, standard deviation, Principal component analysis (PCA)
Classification	SVM and KNN

Chapter 4
Algorithms and Feature Implementation

4.1. Introduction

There are two main approaches to detect falls using acceleration signals: thresholding techniques and machine learning methods. Applications based on the first approach are simple to implement and their computational work is minimal. They are able to detect when a fall occurs. However, the rate of false positives is a significant issue due to the complex nature of human movement. The machine learning approach is more sophisticated and leads to better detection rates. Machine learning is difficult from implementation point of view (for example: requirement of high mathematical skills, use of more computation resources etc.) although they are currently the prevailing trend, since thresholding methods are proved to be ineffective. In addition to the complexities mentioned above, no single implementation has been widely accepted and different paper presents different approach among the variety of machine learning algorithms. This chapter presents the following two fall detection algorithms in detail:

1. Support vector machines (SVM)
2. K-Nearest Neighbors (KNN)

SVM and KNN both are supervised algorithms. We use supervised algorithms because supervised algorithms are much more powerful than unsupervised algorithm.

4.2. Support vector machines (SVM)

SVMs are a relatively new type of supervised machine learning algorithms. In a two-class classification problem, the main goal is to create a model that places every new example in the correct class. SVMs algorithms try to solve this problem by taking the training examples into a higher dimension where they are linearly separable and can be assigned to a class with little uncertainty. Binary class datasets that are linearly separable are easy to classify because the decision boundary of the two classes is just a straight line (Fig 4.1) or plane (Fig 4.2) that divides the feature space into two regions [77].

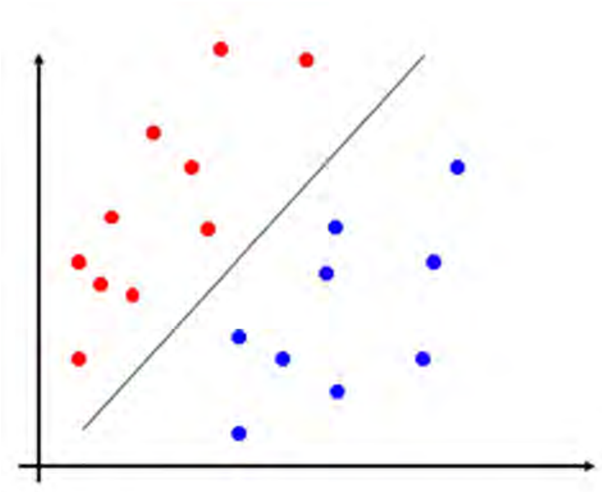


Fig 4.1: A linearly classifiable problem

In SVMs, the input space is transformed into a higher dimensional space using a non-linear mapping (Fig 4.2).

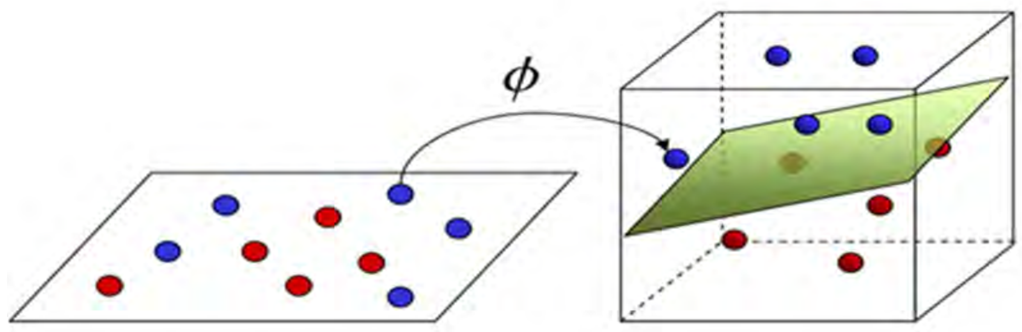


Fig 4.2: A non-linearly classifiable problem [93]

The idea is to take the instances from the original feature space where they are not linearly separable to a new feature space where they are. On this new space, a hyperplane (a straight line in 2 dimensions) is created, and it works as a decision boundary that separates the data; this boundary is also known as the maximum margin hyperplane.

The training points that are closest to the decision boundary are called support vectors. The support vectors uniquely define the maximum-margin hyperplane for the learning problem. In this manner, support vector machines search for a maximum margin hyperplane to separate the data with the examples on the border called support vectors (Fig: 4.3).

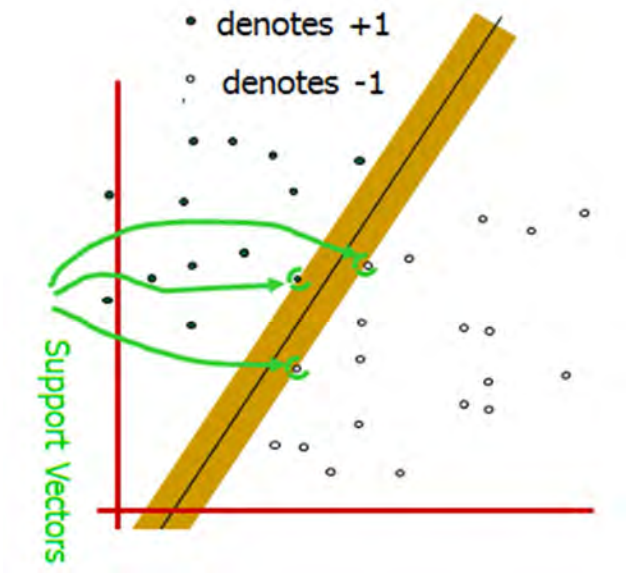


Fig 4.3: Support vectors with margin [94]

Every new entry will be taken to this new space where it will be classified depending on the region. Figure (Fig 4.3) shows an example of a decision boundary.

```
ALGORITHM 1: SVM training pseudocode.
    Input   : Training data
    Output  : Maximum margin hyperplane, $H_{max}$.
              calculate support vectors a(i)
              calculate maximum-margin hyperplane, $H_{max}$

  return $H_{max}$
```

The math involved in SVMs is extremely complex and therefore difficult to implement. The steps in order to realize a SVM training algorithm are described in Algorithm 1.

Calculating the maximal margin hyperplane can be achieved by solving the following equation, Eq. 4.1.

$$X = b + \sum \alpha_i \, c_i a(i).a \text{ -- } (4.1)$$

where

b	= Numeric parameter (i)
α_i	= Numeric parameter
$a(i)$	= Support Vector
a	= Test vector

Finding b, α_i and the support vectors $a(i)$ is a type of optimization problem known as constrained quadratic optimization.

4.2.1. Multiclass SVM

In this work, five classes are to be detected, which is why a multi-class classifier is needed. Multi-class SVMs handle this problem by combining several binary SVMs, using either one-versus-one or one-versus-all as training strategy. In this work, one-versus-one strategy is utilized, where for training purposes one class is considered positive and one other class negative. To get a classification result, a voting strategy is used, where for all pairs of classes the current feature vector is assigned to one of the two classes and finally, the class that receives most votes is considered the correct class.

An illustration of one-versus-one multi-class SVMs is displayed in the following figure (Fig 4.4). In the above figure for each pair of classes, a separating hyperplane is learned. To assign a new sample to a class, it is classified by all pairs of classes, the votes/wins are counted and the sample is assigned to the class with most votes/wins. The one-versus-one classification in this example happens as follows: the first pair of classes is (A, B), the new sample lies on the

’B-side’ of the separating hyperplane and therefore B gets one vote. The second pair of classes is (A, C) and the new sample is classified as A. The last pair is (B, C) classifying the sample as B. Summing up, A has one vote, B has two votes, C has zero votes; therefore, the new sample is classified as B.

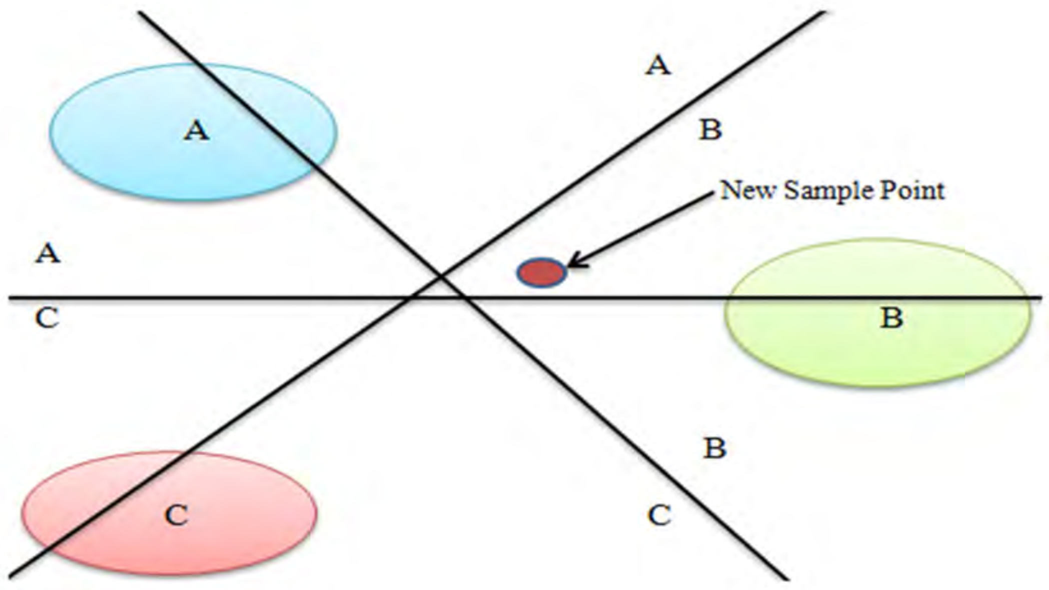


Fig 4.4: Example of a multi-class SVM [92]

4.2.2. Implementation

The implement of the fall detection system comprises of a number of stages as explained here briefly. A 3D accelerometer is attached to chest of human subjects (An overview of the system implement is shown in Figure 4:5). We use the SHIMMER accelerometer module for this study. A fixed orientation of this module is maintained to keep the directions of its three axes the same with respect to bodies for all the subjects. The signals from the accelerometer are streamed wirelessly to a PC. The signals are segmented into window frames and the frames are analyzed to extract statistical features (Mean, Std_dev, and PCA) to characterize the signals related to falls

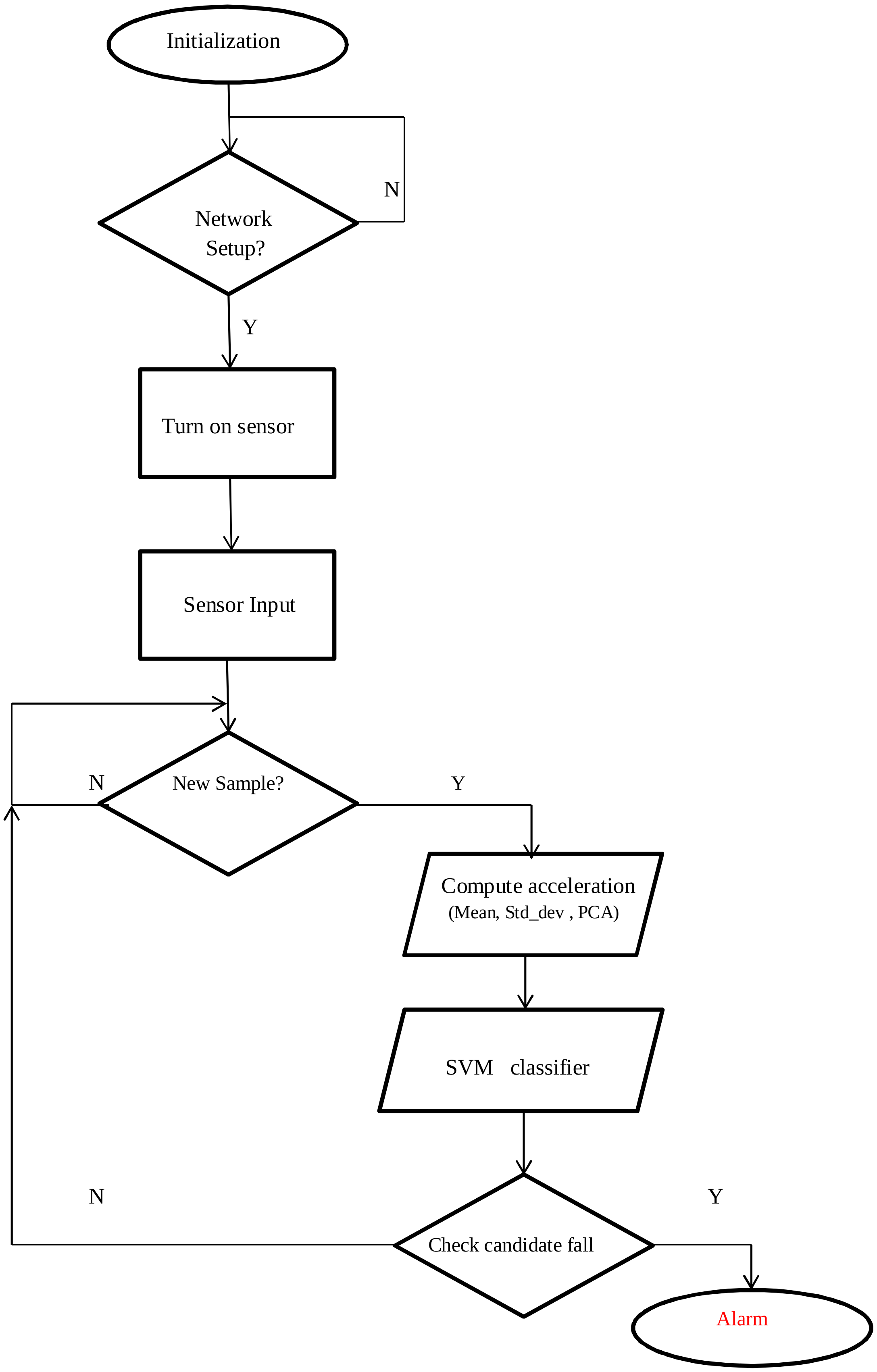


Fig 4.5: Flow chart of our SVM method

The segments are resized and the analysis is repeated to cover the probability that an activity may be divided into one or more windows, overlapping between the previous and next windows. The resulting statistical features are fed to the SVM classifiers for fall detection. The classifier SVM algorithm are set to classify directive fall by recognizing the signal pattern and matching a vector of features with pre-learned ones. SVM algorithm is implemented using “classificationLearner” of MATLAB R2016a toolkit. The algorithm takes in features as input attributes and computes information gain for each attribute in order to determine which attribute leads to the shortest route for a fall or no-fall decision. Sampling frequency, training size and sensor location have impact on performance. We determine the efficacy of SVM classifiers using this method and compare the results with KNN classifier.

4.2.3. Results

SVM learning classifier was used to learn to distinguish among falls and ADL. Whole data

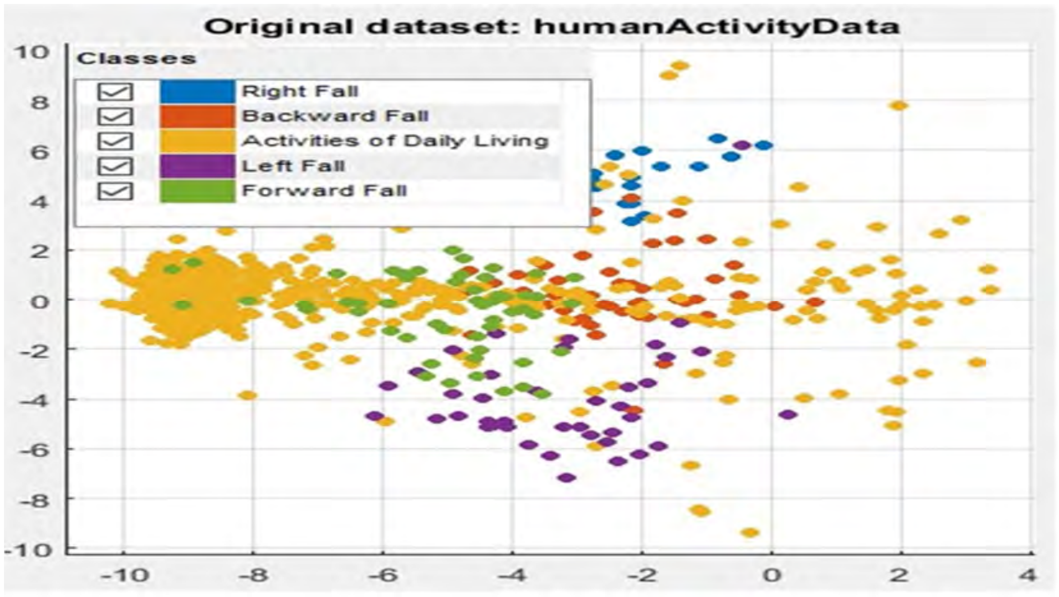


Fig 4.6: Learning Classifier SVM

are divided into two parts randomly, one for training (66%) and another for testing (34%). K-fold cross-validation is used to train and test. Figure 4.6 shows the classification of four types of fall and ADL. Twelve types of activities are considered as one class called ADL. A total of five classes are classified (Four type falls and ADL). Classifier was successful in scoring 97.2 for the learning rate. In testing, total 110 fall data and 366 ADL are taken. Table 4.1 shows the detection result of SVM classifier.

Table 4.1.: Detection result of SVM classifier

Algorithm	**Learning Rate (%)**	**Fall detected. (/110)**	**Fail to Fall detected. (/110)**	**ADL detected. (/366)**	**ADL show in fall detected. (/366)**
SVM	97.2	104	6	360	6

4.2.4. Data Analysis Interface

We develop a GUI to show the detection result of fall detection for SVM algorithm. Figure 4.7 is the graphical user interface (GUI) of fall detection. 3D accelerometer testing data is

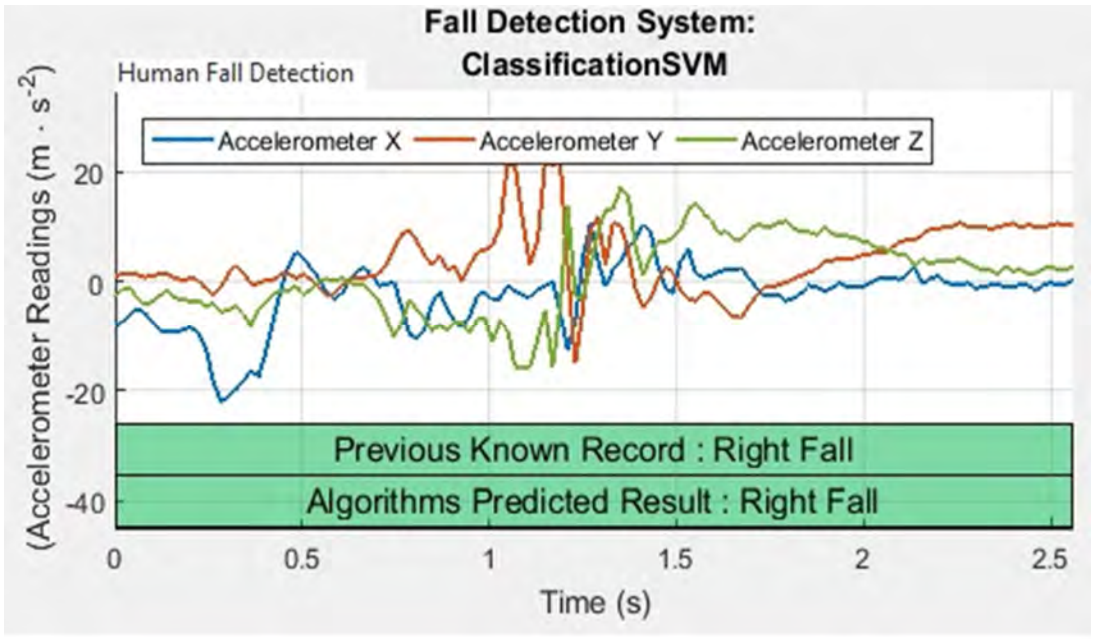


Fig 4.7: Using trained model SVM to predict activity

recorded in dat file. MATLAB is capturing the signals and calculating the value that are previously defined .After calculating the functional value, features vector is fed to KNN classifier. SVM is applying its classification logic to find out the class of imputed vector. SVM shows the result after “Algorithm Predicted Result:” string. From the beginning, we also insert the actual class result in system to show the actual vs. algorithm predicted result. “Previous Known Record…” string indicates the actual result. If actual and predicted result is same than background color remain same as blue. If result is mismatch than background color becomes change as red.

4.3. K-nearest neighbor (KNN)

This algorithm belongs to a subgroup of supervised learning algorithms known as instance-based classifiers. New and unseen instances are compared with instances that are stored in the training set. K algorithms are also called lazy classifiers because there is no training involved. The basic algorithm uses the closest neighbors of the not yet classified new instances to classify them. Every time that a new example needs to be classified, it is compared with all the examples in the dataset. Consequently, k-neighbor algorithms use a straightforward approach to solve classification problems.

Suppose there is a dataset with n classified examples. Each classified example acts as a point in the feature space. A way to calculate the k-nearest neighbors for unclassified examples would be to find the k already classified examples that are closest to the unclassified data. Once the k neighbors have been identified, a majority class vote will take place among them to classify the new instances. Since the attributes are numeric, distance measurements can be used to determine which the k closest neighbors (Fig 4.8) are. Euclidean, Manhattan and city-block distances are commonly used in KNN algorithms.[78]

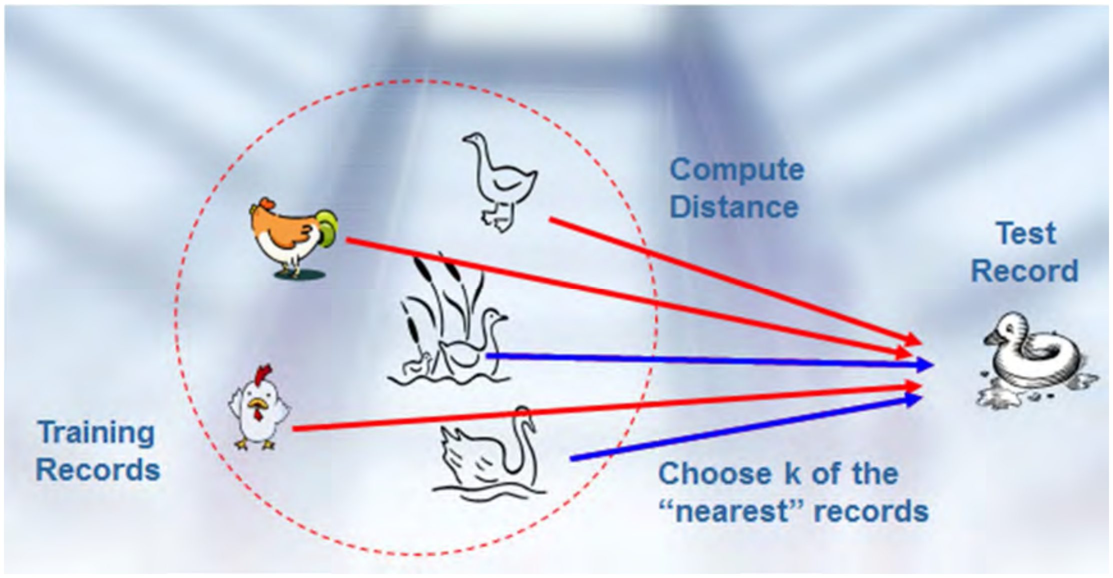


Fig 4.8: KNN algorithms [95]

On KNN algorithms, most of the time, the information or collected data are stored in matrices. Moreover, since every instance must be checked in order for a new entry to be classified.

A basic KNN pseudocode is shown in Algorithm 2.

ALGORITHM 2: KNN training pseudocode.

```
    Input     : Dataset D= {(x1, c1)... (xN ,cN )}, and unlabeled instance x=(x1,...,xN ).
    Output    : predicted class Ci.
    for
        each classified example (xi,ci) do
        calculate distance d(xi,x)
        order d(xi,x) from lowest to highest select k nearest neighbors to x
        vote for majority class among k neighbors, Ci
        return Ci
    end
```

4.3.1. Implementation

The implement of the fall detection system comprises of a number of stages as explained here briefly. A 3D accelerometer is attached to chest of human subjects (An overview of the system implement is shown in Figure Fig 4:9).

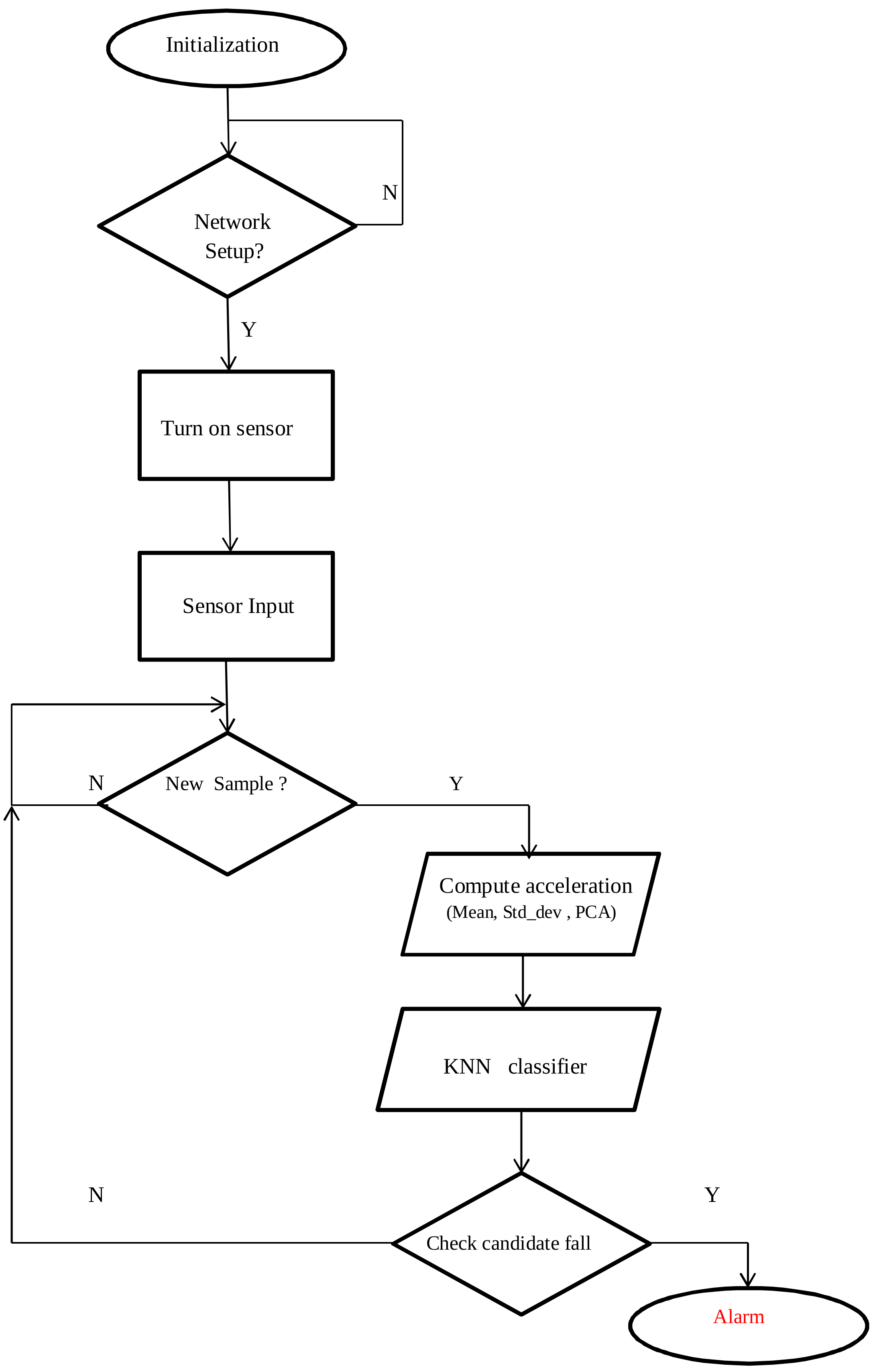


Fig 4.9: Flow chart of the KNN method

We use the SHIMMER accelerometer module for this study. A fixed orientation of this module is maintained to keep the directions of its three axes the same with respect to bodies for all the subjects. The signals from the accelerometer are streamed wirelessly to a PC. The signals are segmented into window frames and the frames are analyzed to extract statistical features (Mean, Std_dev, and PCA) to characterize the signals related to falls. The segments are resized and the analysis is repeated to cover the probability that an activity may be divided into one or more windows, overlapping between the previous and next windows. The resulting statistical features are fed to the KNN classifiers for fall detection. The classifier KNN algorithm are set to classify directive fall by recognizing the signal pattern and matching a vector of features with pre-learned ones. KNN algorithm is implemented using "classificationLearner" of MATLAB R2016a toolkit. The algorithm takes in features as input attributes and computes information gain for each attribute in order to determine which attribute leads to the shortest route for a fall or no-fall decision. Sampling frequency, training size and sensor location have impact on performance. We determine the efficacy of KNN classifiers using this method and compare the results with SVM classifier.

4.3.2. Results

KNN learning classifier was used to learn to distinguish among falls and ADL. Whole data are divided in two parts randomly, one for training (66%) and another for testing (34%). K-fold cross-validation is used to train and test. Figure 4.6 shows the classification of four types of fall and ADL. Twelve types of activities are considered as one class called ADL. A total of five classes are classified (Four type falls and ADL). Classifier was successful in scoring 93.3

for the learning rate. In testing, total 110 fall data and 366 ADL are taken.

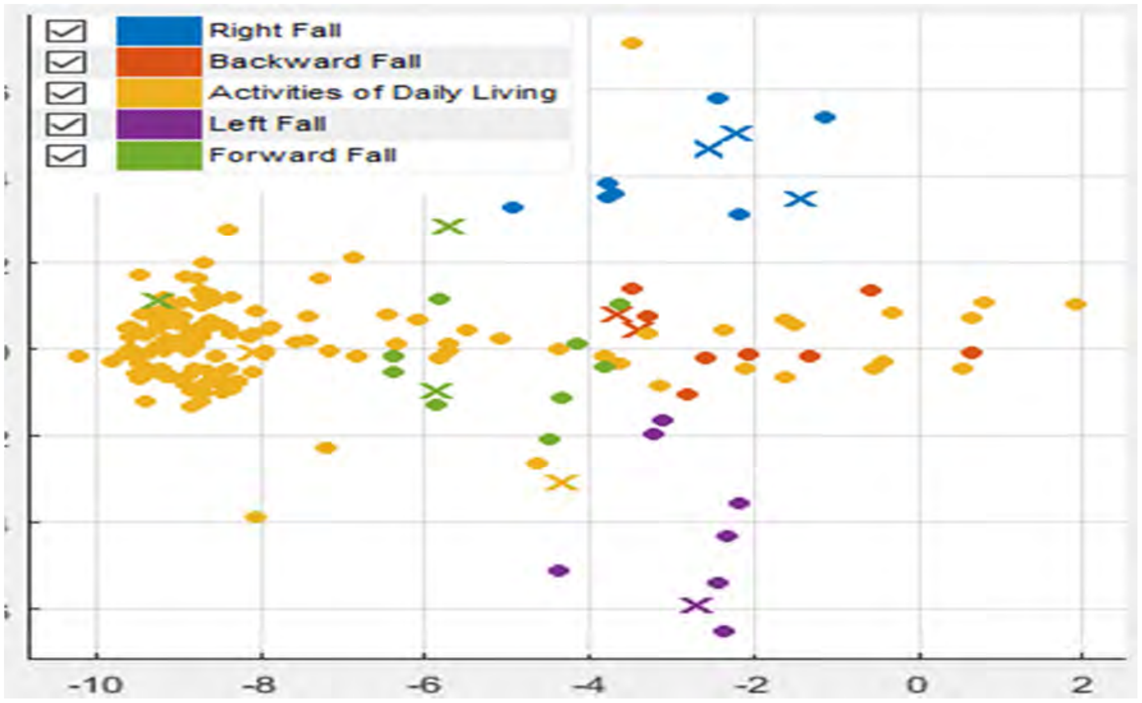


Fig 4.10: Learning Classifier KNN

Table 4.2 shows the detection result of KNN classifier.

Table 4.2.: Detection result of KNN classifier

Algorithm	Learning Rate (%)	Fall detected. (/110)	Fail to Fall detected. (/110)	ADL detected. (/366)	ADL show in fall detected. (/366)
KNN	93.5	101	9	362	4

4.3.3. *Data Analysis Interface*

We develop a GUI to show the detection result of fall detection for KNN algorithm. Figure 4.11 is the graphical user interface (GUI) of fall detection. 3D accelerometer testing data is recorded in dat file. MATLAB is capturing the signals and calculating the value that are previously defined .After calculating the functional value, features vector is fed to KNN classifier. KNN is applying its classification logic to find out the class of imputed vector. KNN shows the result after “Algorithm Predicted Result:” string. From the beginning, we also insert the actual class result in system to show the actual vs. algorithm predicted result. “Previous Know Record…” string indicates the actual result. If actual and predicted result is same than background color remain same as blue. If result is mismatch than background color becomes change as red.

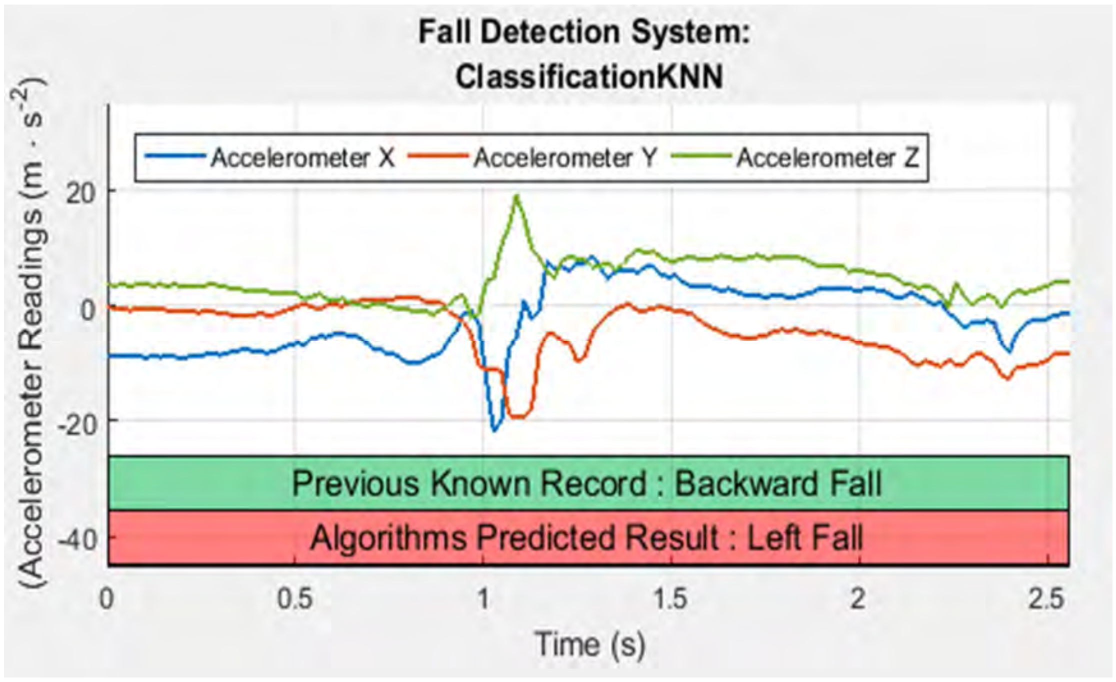


Fig 4.11: Using trained model KNN to predict activity

4.4. Algorithms Comparison

Table 4.3 shows the algorithms complexity comparison between SVM & KNN

TABLE 4.3: Comparison between SVM and KNN learning algorithms

Algorithm	**Data Structure**	**Approach**	**Time Complexity**
K-nearest Neighbor	Matrices	Brute Force	$O(n^2)$
Support Vector Machine	Matrices	Optimization	$O(n a^2 + a^3)$

Chapter 5
Experimental Results

5.1. Introduction

In this work, individual learning classifier was used to learn to distinguish among falls and ADL. A total of 13 subjects have been recruited to perform for the experiments. The results of a classifier are stored in an array known as confusion matrix. It visualizes the learning algorithm's performance (True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN)). Performance is evaluated based on the efficiency of learning algorithms. The accuracy of the system is the most extensively used performance.

$$\text{Accuracy} = \frac{(TN + TP)}{(TP + TN + FP + FN)} \text{-----------------} 5.1$$

The recall or sensitivity or true positive rate is the ratio of the correctly classified positive instances over the entire set of positive instances.

$$\text{Recall} = \frac{TP}{(TP + FN)} \text{-------------------------} 5.2$$

The precision or positive predicted value is the ratio of the number of correctly classified positive instances to the entire set of instances classified as positives.

$$\text{Precision} = \frac{TP}{(TP + FP)} \text{------------------------} 5.3$$

Only fall (not considering direction) and ADL testing results for SVM and KNN classifiers are shown in Table 5.1.

Table 5.1: Test result of different classifiers (only fall & ADL)

Algorithm	**Learning Rate (%)**	**True Pos.**	**False Neg.**	**True Neg.**	**False Pos.**
SVM	97.2	94.54	5.45	98.36	1.60
KNN	93.5	91.81	8.10	98.90	1.09

Only fall (not considering direction) and ADL summary results for SVM and KNN classifiers are shown in Table 5.2.

Table 5.2: Summary results of SVM & KNN classifiers (only fall & ADL)

Algorithm	**Accuracy (%)**	**Precision (%)**	**Recall (%)**
SVM	96.45	98.28	94.54
KNN	95.36	98.82	91.81

5.2. *Classifier 1 (SVM)*

Confusion matrix and summary results of directive fall and ADLs of test data, using SVM classifier are shown in Table 5.3 and Table 5.4, respectively.

Table 5.3: Confusion matrix (SVM)

	Task					
True Class	Right F	22		1		
	Backward F	2	26	2	1	1
	ADL		3	360	3	
	Left F		1	3	24	
	Forward F					27
		Right F	Backward F	ADL	Left F	Forward F
	Predicted Class					

Table 5.4: Summary results of SVM

Task	Total	Accuracy (%)	Precision (%)	Recall (%)
Right F	23	97.60	99.54	95.65
Backward F	32	90.17	98.90	81.25
ADL	366	96.47	94.79	98.36
Left F	28	92.41	98.96	85.71
Forward F	27	99.88	99.77	100

5.3. Classifier 2 (KNN)

Confusion matrix and summary results of directive fall and ADLs of test data, using KNN classifier are shown in Table 5.5 and Table 5.6, respectively.

Table 5.5: Confusion matrix (KNN)

True Class	Task					
	Right F	23				
	Backward F	2	25	4	1	
	ADL		3	362	1	
	Left F		2	2	24	
	Forward F		1	3		23
		Right F	Backward F	ADL	Left F	Forward F
	Predicted Class					

Table 5.6: Summary results of KNN

Task	Total	Accuracy (%)	Precision (%)	Recall (%)
Right F	23	99.77	99.56	100
Backward F	32	88.38	98.29	78.125
ADL	366	95.36	92.35	98.90
Left F	28	92.63	99.48	85.71
Forward F	27	92.59	100	85.18

5.4. Result assessment

We assess the misclassification and classification rate over the different variants of classifiers. The optimal classifier is selected by tuning parameters True Positive (TP), True Negative (TN), False Negative (FN), False Positive (FP), Accuracy, Precision and Recall.

The vertical axis of the graph (Fig 5.1) represents the True positive and True Negative peck of SVM and KNN respectively. SVM classifier is providing better results as its classification True Positive is high. SVM clearly shows a large peak than KNN of True Positive tuning parameter. True Negative pecks are all most same of SVM and KNN.

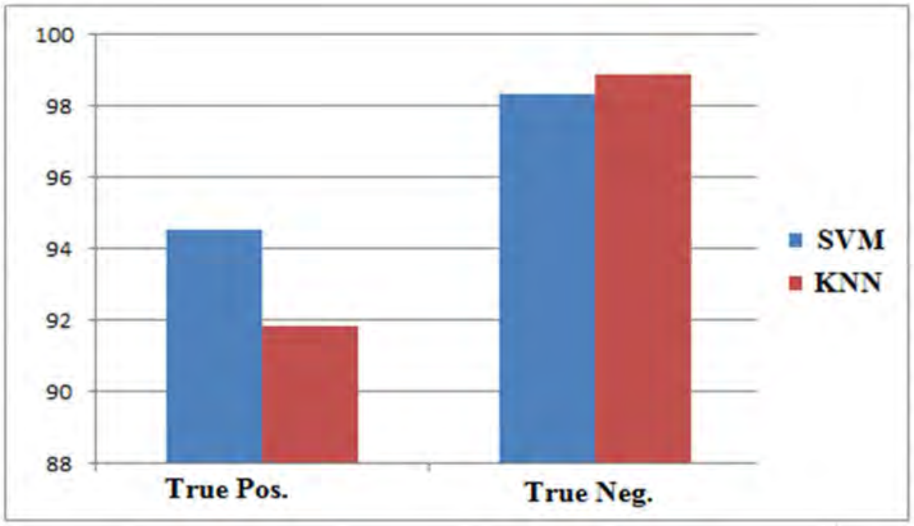


Fig 5.1: Fall detection performance (True Pos. & True Neg.) of SVM & KNN

The vertical axis of the graph (Fig 5.2) represents the False Positive and False Negative peck

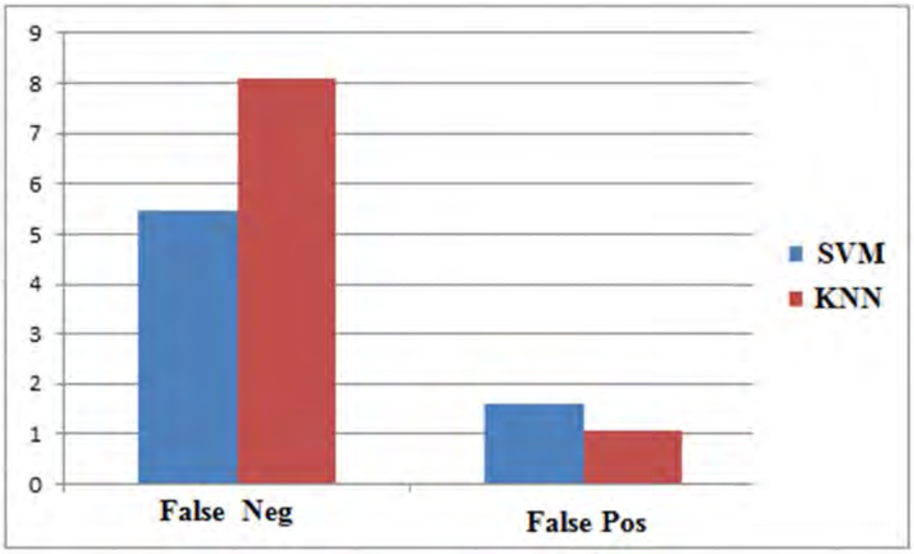


Fig 5.2: Fall detection performance (False Neg. & False Pos.) of SVM & KNN

of SVM and KNN respectively. SVM classifier is providing better results as its classification False Negative is low. SVM clearly shows a small peak than KNN of False Negative tuning parameter. Though KNN has low rate of False Positive but it has also high rate of False Negative.

The vertical axis of the graph (Fig 5.3) represents the accuracy peck of SVM and KNN

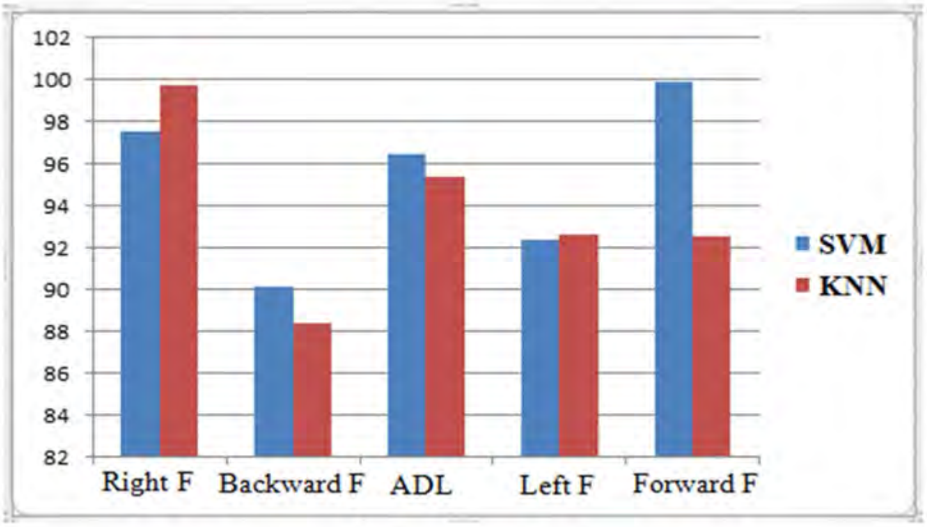


Fig 5.3: Accuracy of SVM & KNN classifiers

respectively. SVM classifier is providing better results as its classification of Forward Fall, Backward Fall and ADL high. KNN classifier is providing better of Right Fall, Left Fall high.

The vertical axis of the graph (Fig 5.4) represents the precision peck of SVM and KNN respectively. SVM classifier is providing better results as its classification Backward Fall, Right Fall and ADL high. KNN classifier is providing better of Forward Fall, Left Fall high. In forward and left fall, the False Positive (FP) is low rate as a result KNN precision high than SVM.

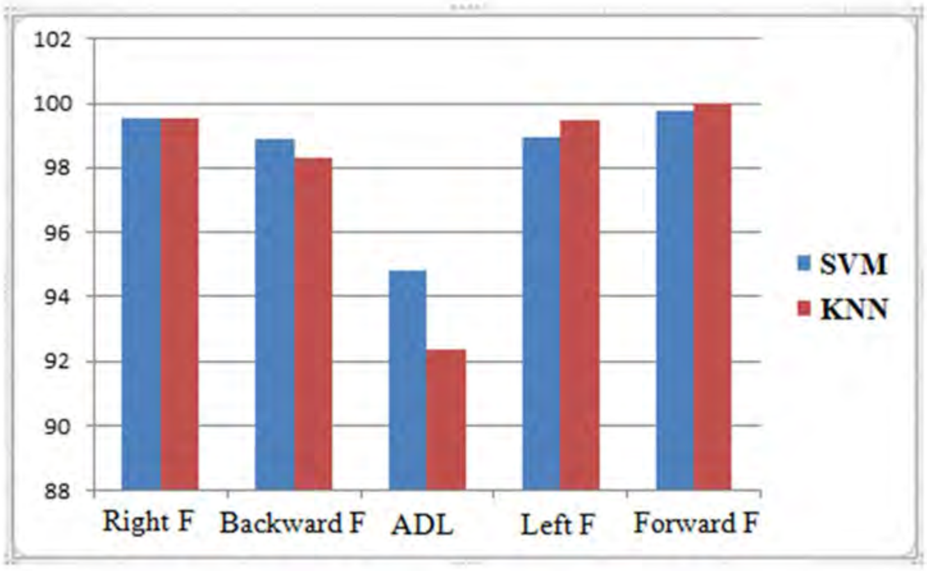


Fig 5.4: Precision of SVM & KNN classifiers

The vertical axis of the graph (Fig 5.5) represents the Recall peck of SVM and KNN respectively. SVM classifier is providing better results as its classification Forward Fall, Backward Fall and Left Fall high. KNN classifier is providing better of Backward Fall and ADL high. In forward and backward fall, the False Negative (FN) is low rate as a result SVM recalls high than KNN.

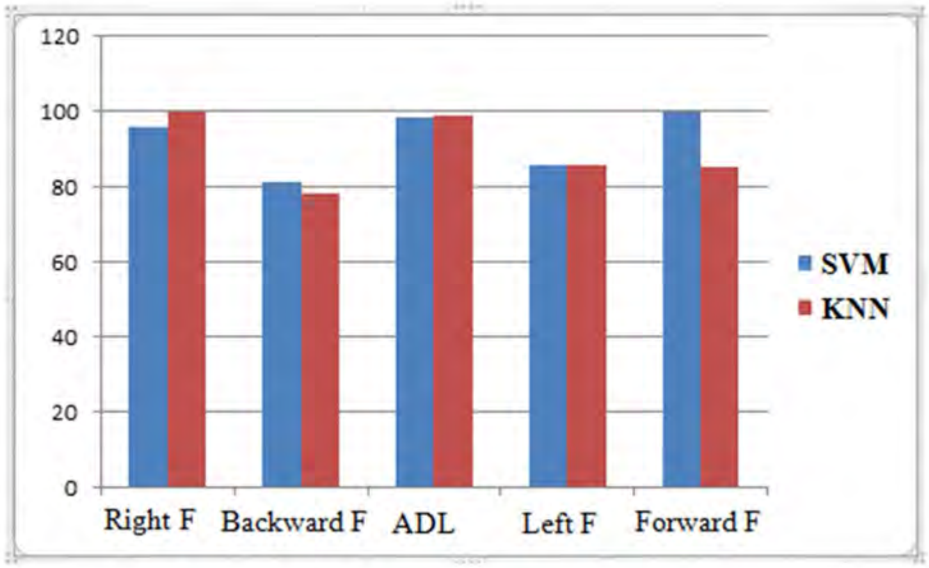


Fig 5.5: Recall of SVM & KNN classifiers

5.4.1. Conclusion

We can conclude that SVM classifier is providing better results as its classification accuracy is high and error rate is minimal over the investigated classifiers based on tuning parameters.

5.4.2. Discussion

This result describes two machine learning algorithm-based methods for direction-sensitive fall detection using single 3D accelerometer. Fall data are collected and analyzed to extract important and sensitive statistical features related to a fall and its direction. After successful completion of this work, we get the result in optimum classifier for direction-sensitive fall detection system. Four types of falls were identified with high accuracy, precision and recall using the optimum classifier, SVM.

5.5. Real time fall detection

As SVM classifier is providing better results based on tuning parameters. So we discard the KNN algorithm and implement the SVM algorithm in real time. Real time fall detection is developed in MATLAB language to be executed on the SHIMMER MSPP430 microprocessor. Third party software Realterm version 2.0.0.70 is used for PC port scanning (recommended by SHIMMER [79]). Port is scanned every .256s and data is sent to MATLAB. MATLAB extracts statistical features (Mean, Std_dev, and PCA) and fed to the SVM classifier to determine directive fall or ADL decision. SVM algorithm is implemented using "classificationLearner" of MATLAB R2016a toolkit. After developing the real time system, again five human subjects who are recruited for this testing work. A total of 52 falls and 144 ADL have been performed and we get the result (table 5.7 and table 5.8)

Table 5.7: Confusion matrix (SVM) to detect real time fall

	Task					
Directive Fall Class	Right F	12	1			
	Backward F	1	11	1		
	ADL		2	140	1	1
	Left F			2	10	1
	Forward F			1		12
		Right F	Backward F	ADL	Left F	Forward F
	Real Time Fall Detection Result					

Table 5.8: Summary results of SVM to detect real time fall

Task	Total	Accuracy (%)	Precision (%)	Recall (%)
Right F	13	95.88	99.41	92.30
Backward F	13	91.48	98.09	84.61
ADL	144	94.76	92.66	97.22
Left F	13	88.18	99.29	76.92
Forward F	13	95.60	98.82	92.30

From the table, we compare the simulation result and real time result table 5.2 vs. table 5.6 and table 5.3 vs. 5.7. We point out the performance of the real time is very similar to the simulation result. In fact, the similar accuracy, precision, recall were obtained. These results confirm the quality of the real time system to accurately classify fall and ADL events. Developed real time fall detection system is providing similar result like simulation result with less technical error rate and high classification accuracy.

5.6. Comparing the performance with existing works

In this thesis, we have used single accelerometer and SVM learning algorithm to detection direction sensitive fall and found accuracy leads to 96.45%. Table 5.9, presents the performance with existing works.

Table 5.9: Performance comparison with existing works

SL	Authors	Hardware Platform	Algorithms	Accuracy (%)
01	Beevi et al [16]	Accelerometer	Linear prediction	84

SL	Authors	Hardware Platform	Algorithms	Accuracy (%)
02	Ojetola et al [17]	2 Accelerometers, Gyroscopes	C4.5 decision trees	90
03	Jantaraprim et al. [54]	Accelerometer	Threshold based	96.11
04	Anania et al. [57]	Accelerometer	Threshold based, Kalman filter	90
05	Zhang et al. [60]	Accelerometer	One class support vector machine	96.7
06	Zhang et al. [65]	Sun SPOT Transceiver, Accelerometer	SVM, Bayesian network	93
07	Gjoreski et al. [58]	3 Accelerometers	Random Forest, Threshold set manually	90
08	Lee et al. [80]	Accelerometer	-	93.2
09	Noury et al. [82]	2 Accelerometers	-	81
10	Noury et al. [83]	Accelerometer, Posture, Vibrator	-	85
11	Tapia et al. [84]	5 Accelerometers, Heart rate sensor	C4.5 Decision Tree	80.6
12	Hwang et al. [85]	Accelerometer, Gyroscope, Tiltsensor	Threshold based	96.7
13	Degen et al. [86]	2 Accelerometers	Threshold based	65

5.6.1. Our proposed system

Table 5.10 represents our proposed system.

Table 5.10: Our proposed system

SL	**Authors**	**Hardware Platform**	**Algorithms**	**Accuracy (%)**
01.	Farhad et al.	Single Accelerometer	SVM	96.45

5.6.2. Discussion

Four types of falls were simulated by data-set and its performance leads to 96.45 % accuracy a sampling rate of 50 Hz, exceeding the performance provided by the literature. From the table 5.8, most of the fall detection algorithms are based on thresholds set. A major challenge in fall detection is identifying appropriate thresholds. Such algorithms do not generalize well for unseen data sets. To minimize the false alarm rate of fall detection, some researchers embedded extra sensor with main sensor like Acoustic sensors, Gyroscopes, Cardio tachometer, Magnetometer, Barometric Pressure. Existing works can detect only lateral fall not direction. It shows only fall when forward, backward, left, right fall occur. In addition to fall detection , it is also important to determine the direction of a fall, which could help in the location of joint weakness or post-fall fracture and help decrease reaction time. This work not only shows a machine learning algorithm that provides accuracy beyond the currently available algorithms but also shows direction-sensitive and cost-effective fall detection system using single 3D accelerometer.

Chapter 6
Conclusions and Future Work

6.1. Summary

In this thesis, we have analyzed sensor accelerometer signal to determine their reliability to discriminate between falls and ADL. We extract important and sensitive statistical features related to a fall and its direction. We analyze the accelerometer data for a detected fall to decide on the fall direction. We have explored the features of fall detection. Based on our results, the accelerometer appears to be the most reliable sensor. This directive fall detection system uses single accelerometer which is of low cost. Using the information provided by the sensor, two algorithms (SVM & KNN) are implemented and tested. The algorithms are simple and can easily be implemented in MATLAB platforms. In our data-set, its performance leads to 96.45 % accuracy and 92% precision. The SVM analysis confirms the good performance of the method. A comparative study with the performance of two machines learning to fall detection algorithms, shows that the implemented SVM & KNN is very competitive. We use more advanced pattern recognition and machine learning techniques that increase the robustness of the fall detection algorithm. We implement the algorithm in real world environment and found the performance of the real time is very similar to the simulation that increases the acceptance of the fall detection system. After successful completion of this thesis, we get the result in optimum classifier for direction-sensitive fall detection system. Four types of falls are identified with high accuracy, precision and recall using the optimum classifier, SVM.

6.2. Limitations

The elderly and infirm people are the primary end-users of fall detection solutions. However, due to ethical concerns, the algorithms developed in this thesis were only evaluated using data from young and healthy subjects. Thus, it is necessary that the proposed SVM based machine learning algorithm is evaluated on data gathered from the elderly and disabled subjects.

6.3. Future Work

There are several areas of future work that can serve to improve the system functionality and provide additional evaluation of its performance.

1. Chest is the best body location for fall detection accuracy but waist is the best comfortable body locations for placement of sensor according to subjects. In future, fall detection studies may be done for a sensor placed in the waist while keeping the accuracy on the same level.
2. Fall pre-impact stage consists of when a fall begins before a faller's body makes impact with the floor. This stage is characterized by acceleration of the faller approaching zero just before the impact. For preventive fall, it is also important to identify pre-impact stage of fall and alert the user for minimizing risk.
3. Further work may focus on: online monitoring and messaging system to mobile phone.

References:

[1] Abbate, S., Avvenuti, M., Corsini, P., Vecchio, A., and Light, J., “Monitoring of Human Movements for Fall Detection and Activities Recognition in Elderly Care Using Wireless Sensor Network : A Survey,” in Yen Kheng Tan (Ed.), Wireless Sensor Networks: Application-Centric Design, Chap 1, pp. 1-20,InTech, Rijeka, Croatia, 2010.

[2] Jara, A.J., Izquierdo, M.A., and Skarmeta, A.F., "An ambient assisted living system for telemedicine with detection of symptoms,” International Work conference on the Interplay between Natural and Artificial Computing, vol.2, pp. 75- 84, 2009.

[3] Scanaill, C.N., Carew, S., Barralon, P., Noury, N., Lyons, D., and Lyons, G.M., “A review of approaches to mobility telemonitoring of the elderly in their living environment,” Annals of Biomedical Engineering, vol. 34, pp. 547-563, 2006.

[4] Pigot, H., Mayers, A., and Giroux, S., “The intelligent habitat and everyday life activity support," International conference on Simulations in Biomedicine, pp. 507-516, 2003.

[5] Baraka, A., Shokry, A., Omar, I., Kamel, S., Fouad, T., El-Nasr, M. A., and Shaban, H., “A WBAN for Human Movement Kinematics and ECG Measurements,” J. E-Health Telecommunication Systems and Networks, vol. 1, pp. 19-25, 2012.

[6] Tinetti, M. D., Liu W. L., and Claus E. B., “Predictors and prognosis of inability to get up after falls among elderly persons,” JAm Med Assoc View ArticleGoogle Scholar vol 269, pp. 65-70 Jan 1993.

[7] http://www.misa.ie/node/936, http://www.trilcentre.org [accessed last on 19 June, 2016]

[8] Stevens J. A., Corso, P. S, Finkelstein E. A., and Miller, T.R., “The costs of fatal and nonfatal falls among older adults,” Injury Prevention pp.290–295, Dec 2006.

[9] https://www.cdc.gov/homeandrecreationalsafety/falls/fallcost.html [accessed last on 19 June, 2016]

[10] Burns, A., Greene, B. R., McGrath, M. J., O'Shea, T. J. ; Kuris, B., Ayer, S. M. , and Stroiescu, F., “SHIMMER™ –Wireless Sensor Plat form for Noninvasive Biomedical Research,” IEEE Sensors Journal, Vol.10, No.9, pp. 1527-1534, Sept. 2010.

[11] Tolkiehn, M., Atallah, L., Lo, B., and Yang, G. Z., "Direction Sensitive Fall Detection Using A Triaxial Accelerometer and A Barometric Pressure Sensor.", in IEEE Annual International Conference on Engineering in Medicine and Biology Society (EMBC) , Sept. 2011, pp. 369-372 *(2011)*

[12] Quoc T. H., Uyen, D. N. , Su V. T., Nabili, A., and Binh Q. T., “ Fall Detection System Using Combination Accelerometer And Gyroscope”, Institute of Research Engineers and Doctors, Proc. of the Second Intl. Conf. on Advances in Electronic Devices and Circuits EDC -2013.

[13] Liu, L., Popescu, M., Rantz, M., and Skubic, M., “ Fall Detection Using Doppler Radar And Classifier Fusion, ” Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics ,Jan. 2012.

[14] Anderson, D., Luke, R. H., Keller, J. M., Skubic, M., Rantz, M. and Aud, M., “Linguistic summarization of video for fall detection using voxel person and fuzzy logic,” Computer Vision and Image Understanding, Vol. 113 No. 1, pp. 80-89, 2009.

[15] Ge, Y., and Xu, B., “Detecting Falls Using Accelerometers By Adaptive Thresholds In Mobile Devices,” Journal of Computers, Vol.9, No.7, July-2014, pp. 1553-1559.

[16] Beevi, F. H. A., Pedersen C. F., Wagner, S., and Hallerstede, S., “Lateral Fall Detection via Events in Linear Prediction Residual of Acceleration,” Ambient Intelligence - Software and Applications, Advances in Intelligent Systems and Computing, Springer International Publishing Switzerland, June 2014, vol. 291, pp. 201-209.

[17] Ojetola, O., Gaura, E.I., and Brusey, J., "Fall Detection with Wearable Sensors - SAFE (SmArt Fall dEtection),” in Seventh International Conference on Intelligent Environments, Jul. 2011, pp. 318-321 *(2011)*.

[18] Javed, K., Babri, H. A., and Saeed, M., “Feature Selection Based on Class Dependent Densities for High Dimen sional Binary Data,” IEEE Transactions on Knowledge and Data Engineering, vol. 24, No. 3, pp. 465-477, 2012.

[19] Suriani, N. S., and Hussain, A., "Sudden Fall Classification using Motion Features," in IEEE 8th International Colloquium on Signal Processing and its Applications, March 2012, pp. 519-524 *(2012)*

[20] Lim, D., Park, C., Kim, N.H., Kim, S. H., and Yu, Y. S., “Fall-Detection Algorithm Using 3-Axis Acceleration: Combination With Simple Threshold And Hidden Markov Model,” Journal of Applied Mathematics, pp. 1-8, 2014.

[21] Vallejo, M., Isaza, C. V., and Lopez, J. D., " Artificial Neural Networks as an Alternative to Traditional Fall Detection Methods," in 35th Annual International Conference of the IEEE EMBS, Jul. 2013, pp. 1648-1651 *(2013)*.

[22] Sengto, A.; and Leauhatong, T., “Human falling detection algorithm using back propagation neural network,” In Proceedings of the 5th Biomedical Engineering International Conference, Ubon Ratchathani, IEEE: Ubon Ratchathani, Thailand, 2012; pp. 1–5., Thailand, Dec. 2012 ;

[23] Anderson, D., Luke, R. H., Keller, J. M., Skubic, M., Rantz, M., and Aud, M., “Linguistic Summarization of Video for Fall Detection Using Voxel Person and Fuzzy Logic,” Comput Vis Image Und, vol. 113, No 1, pp. 80-89, 2009.

[24] Takeda, T.; Sakai, Y.; Kuramoto, K.; Kobashi, S.; Ishikawa, T.; and Hata, Y. , “Foot age estimation for fall-prevention using sole pressure by fuzzy logic,” In Proceedings of the International Conference on Systems, Man, and Cybernetics, Anchorage, AK, USA, 9–12; pp. 769–774, Oct. 2011.

[25] Fu, Z., Delbruck, T., Lichtsteiner, P., and Culurciello, E., "An Address-Event Fall Detector for Assisted Living Applications," in IEEE Transactions on Biomedical Circuits And Systems, vol. 2, No. 2, pp. 88- 96, June, 2008.

[26] Khawandi, S., Daya, B., and Chauvet, P., “Implementation of a monitoring system for fall detection in elderly healthcare”, Procedia Computer Science, Vol 3 pp. 216-220, 2011.

[27] Yu, M., Naqvi, S. M., and Chambers, J., "A Robust Fall Detection System for the Elderly in A Smart Room," in IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), March 2010, pp. 1666-1669 (2010).

[28] Moylan, K. C. and Binder, E. F., “Falls in older adults: Risk assessment, management and prevention,” The American Journal of Medicine, Vol. 120 Issue 6 pp. 493-496, 2007.

[29] Liu and Cheng, W., “Fall detection with the support vector machine during scripted and continuous unscripted activities,” Sensors, Vol 12 issue 9 pp. 12301-12316, 2012.

[30] Overstall, P. W., Smith, A. N., Imms F. J., and Johnson, A. L., “Falls in the elderly related to postural imbalance,” British Medical Journal,Vol 1 pp. 261-264, 1977.

[31] Tinetti, M., E., “Speechley M. Prevention of falls among the elderly,” New England journal of medicine, pp.1055-1059. 1989

[32] Campbell, A. J., Spears, G. F., and Borrie M. J., “Examination by logistic regression modelling of the variables which increase the relative risk of elderly women falling compared to elderly men,” Journal of clinical epidemiology, Vol. 43, pp. 1415-1420. 1989

[33] Robbins, A. S., “Predictors of falls among elderly people”, Results of two population-based studies,” Archives of internal medicine, Vol 149, pp. 1628-1633, 1989.

[34] Todd, C., and Skelton, D., “What are the main risk factors for falls among older people and what are the most effective interventions to prevent these falls?” Copenhagen, WHO Regional Office for Europe (Health Evidence Network report; http://www.euro.who.int/document/ E82552. pdf, accessed 21 Dec. 2016).

[35] Skelton, D., A., "Effects of physical activity on postural stability," Age and ageing, Vol. 30, issue 4 pp. 33-39, 2001

[36] Ray, W., Thapa, P., and Gideon, P., "Benzodiazepines and the risk of falls in nursing home residents," Journal of the American Geriatrics Society, Vol. 48, pp. 682-685, 2000.

[37] Dionyssiotis, Y., "Analyzing the problem of falls among older people," International Journal of General Medicine, 2012

[38] Jack, C., I., "Prevalence of low vision in elderly patients admitted to an acute geriatric unit in Liverpool: elderly people who fall are more likely to have low vision," Gerontology, Vol. 41, pp. 280-285. 1995.

[39] Lord, S. R., and Bashford, G.M., "Shoe characteristics and balance in older women," Journal of the American Geriatrics Society, Vol. 44, pp. 429-433, 1996.

[40] Sturnieks, D. L., George, R. S, and Lord, S. R., "Balance disorders in the elderly," Neurophysiologie Clinique/Clinical Neurophysiology, Vol. 38 Issue 6,pp. 467-478, 2008.

[41] Salkeld, G., "Quality of life related to fear of falling and hip fracture in older women: a time trade off study," BMJ, Vol. 320, pp.341-346, 2000.

[42] Lord, C. J. and Colvin, D. P., "Falls in the elderly: Detection and assessment," In Proc. Ann. Intl. Conf. Engineering in Medicine and Biology Society, pp. 1938-1939, Oct. 1991

[43] http://www.mass.gov/eohhs/gov/departments/dph/ [accessed last on 19 June, 2016]

[44] Khawandi, S., Daya, B., and Chauvet, P., "Implementation of a monitoring system for fall detection in elderly healthcare," Procedia Computer Science,Vol. 3 pp. 216-220, 2011.

[45] Ozcan, K.; Mahabalagiri, A.; Casares, M. and Velipasalar, S., "Automatic fall detection and activity classification by a wearable embedded smart camera," IEEE J. Emerg. Sel. Top. Circuits Syst. Vol.3, pp. 125–136, 2013.

[46] Crispim C. F., Bremond, F., and Joumier, V. , "A multi-sensor approach for activity recognition in older patients," In The Second International Conference on Ambient Computing, Applications, Services and Technologies - AMBIENT, September 2012.

[47] Fu, Z., Delbruck, T., Lichtsteiner, P., and Culurciello, E., "An address-event fall detector for assisted living applications," Biomedical Circuits and Systems, IEEE Transactions on, Vol. 2, issue 2, pp. 88-96, June 2008.

[48] Litvak, D., Zigel, Y., and Gannot I., "Fall detection of elderly through floor vibrations and sound," In Engineering in Medicine and Biology Society, 30th Annual International Conference of the IEEE, pp. 4632-4635, August 2008.

[49] Luo, X., Liu, T., Liu, J., Guo, X., and Wang, G., "Design and implementation of a distributed fall detection system based on wireless sensor networks," EURASIP Journal on Wireless Communications and Networking, Vol. 1, pp. 1-13, 2012.

[50] Khawandi, S., Daya, B., and Chauvet, P., "Implementation of a monitoring system for fall detection in elderly healthcare," Procedia Computer Science, Vol. 3 pp. 216-220, 2011.

[51] Liu, H. and Zuo, C., "An improved algorithm of automatic fall detection," AASRI Procedia, Vol. 1, pp. 353-358, 2012.

[52] Olivieri, D. N., Conde, I. G. M., and Sobrino, X. A. V., "Eigenspace-based fall detection and activity recognition from motion templates and machine learning," Expert Systems with Applications, Vol. 39 issue 5, pp. 5935-5945, 2012.

[53] Bashir, F., "Real life applicable fall detection system based on wireless body area network," In Proceedings of the 10th Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA, pp. 62–67, 11–14 January 2013.

[54] Jantaraprim, P., Phukpattaranont, P., Limsakul, C., and Wongkittisuksa, B., "Evaluation of fall detection for the elderly on a variety of subject groups," In Proceedings of the 3rd International Convention on Rehabilitation Engineering & Assistive Technology, pp. 11. ACM, 2009.

[55] Li, Q., Stankovic, J. A., Hanson, M. A., Barth, A. T., Lach, J., and Zhou, G., "Accurate, fast fall detection using gyroscopes and accelerometer-derived posture information," In BSN '09: Proceedings of the 2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks, IEEE Computer Society, pp. 138-143. 2009.

[56] C. Wang, C. Chiang, P. Lin, Y. Chou, I. Kuo, C. Huang, and C. Chan, " Development of a fall detecting system for the elderly residents," In Bioinformatics and Biomedical Engineering, The 2nd International Conference on, pp. 1359-1362, 2008.

[57] Anania, G., Tognetti, A., Carbonaro, N., Tesconi, M., Cutolo, F., Zupone, G., and Rossi, D., "Development of a novel algorithm for human fall detection using wearable sensors" In Sensors, IEEE, pp. 1336-1339, Oct. 2008.

[58] Sposaro, F., and Tyson, G., " iFall: An android application for fall monitoring and response," Engineering in Medicine and Biology Society, Annual International Conference of the IEEE, pp. 6119-6122, sept., 2009

[59] Kaenampornpan, M., Anuchad, T., and Supaluck, P., “Fall detection prototype for thai elderly in mobile computing era,” In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology , 8th International Conference on, pages 446-449, May 2011.

[60] Zhang, T., Wang, J., Xu, L., and Liu, P., “Fall detection by wearable sensor and one-class svm algorithm,” In Intelligent Computing in Signal Processing and Pattern Recognition, vol. 345, pp. 858-863. 2006

[61] Liu, S.H., and Cheng, W. C., “Fall Detection with the Support Vector Machine during Scripted and Continuous Unscripted Activities,” Sensors, Vol. 12 issue 9, pp. 12301-12316, 2012

[62] Zhao, Z., Chen, Y., Wang, S., and Chen, Z., “ Fallalarm: Smart phone based fall detecting and positioning system,” Procedia Computer Science, Vol. 10, pp. 617-624, 2012

[63] Shi, Y.; Shi, Y.C.; and Wang, X. “Fall detection on mobile phones using features from a five-phase model,” In Proceedings of the 9th International Conference on Ubiquitous Intelligence and Computing, and Autonomic and Trusted Computing, Fukuoka, Japan, 4–7 pp. 951–956, Sept. 2012;

[64] Humenberger, M.; Schraml, S.; Sulzbachner, C.; Belbachir, A.N.; Srp, A.; and Vajda, F. “Embedded fall detection with a neural network and bio-inspired stereo vision,” In Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, RI, USA, pp. 60–67, June 2012.

[65] Zhang, M., and Sawchuk, A. A., “Context-aware fall detection using a bayesian network,” In Proceedings of the 5th ACM International Workshop on Context Awareness for Self-Managing Systems, pp. 10-16, 2011

[66] Lan, M., Nahapetian, A., and Vahdatpour, A., Au, L., Kaiser, W., and Sarrafzadeh, M., “Smartfall: an automatic fall detection system based on subsequence matching for the smartcane,” In Proceedings of the Fourth International Conference on Body Area Networks, page 8, 2009.

[67] Chen, Y.T.; Lin, Y.C.; and Fang, W.H., “A hybrid human fall detection scheme,” In Proceedings of the International Conference on Image Processing, Hong Kong, China pp. 3485–3488, Sept. 2010.

[68] Gjoreski, H., Lustrek, M., and Gams, M., “Accelerometer placement for posture recognition and fall detection,” In Intelligent Environments (IE), 7th International Conference on, pp. 47-54, July 2011.

[69] Martin T., Majeed B., and Lee B.S., "Nick Clarke Fuzzy Ambient Intelligence for Next Generation Telecare ," IEEE International Conference on Fuzzy Systems Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July, 2006

[70] Bagalà, F., Becker, C., Cappello, A., Chiari, L., Aminian, K., Hausdorff, J. M., Zijlstra, W., and Klenk, J., " Evaluation of accelerometer-based fall detection algorithms on real-world falls," PLoS ONE, Vol 7 issue 5 pp 37062, 2012.

[71] Ward, G., Holliday,N., Fielden, S., and Williams, S., "Fall detectors: a review of the literature," Journal of Assistive Technologies, vol. 6 issue 3, pp 202-215, 2012.

[72] Polastre, J., Szewczyk, R., and Culler, D., "Telos: Enabling ultra-low power wireless research," in Proc. 4th Int. Symp. Inf. Process. Sens. Networks, Los Angeles, CA, pp. 364–369, 2005.

[73] Lo, B. P. L., Thiemjarus, S., King, R., and Yang, G. Z., "Body sensor network A wireless sensor platform for pervasive healthcare monitoring," in Proc. 3rd Int. Conf. Pervasive Computing, Munich, Ger- many, pp. 77–80, 2005.

[74] Doughty, K., Lewis, R., and McIntosh, A., "The design of a practical and reliable fall detector for community and institutional telecare," Journal of Telemedicine and Telecare, vol. 6 issue 1 pp. 150-154, 2000.

[75] Gjoreski, H., Lustrek, M., and Gams, M., "Accelerometer placement for posture recognition and fall detection" In Intelligent Environments (IE), 7th International Conference on, pp. 47-54, July 2011.

[76] Ojetola, O., Detection of Human Falls using Wearable Sensors, Phd. Thesis, Faculty of Engineering and Computing, Coventry University, 2013

[77] https://en.wikipedia.org/wiki/Support_vector_machine [accessed last on 19 June, 2016]

[78] http://www.saedsayad.com/k_nearest_neighbors.htm [accessed last on 19 June, 2016]

[79] http://www.shimmersensing.com/menu/products/matlab-id [accessed last on 11 June, 2014]

[80] Lee, Y., Kim, J.,Son, M., and Lee, M.,"Implementation of accelerometer sensor module and fall detection monitoring system based on wireless sensor network", Engineering in Medicineand Biology Society, EMBS 29th Annual International Conference of the IEEE, pp.2315-2318, 2007

[81] Velasco P., M., Cidoncha, M. G., and Marin, O, R., "The inescapable smart impact detection system (ISIS): An ubiquitous and personalized fall detector based on a distributed 'divide and conquer strategy," Engineeringin Medicine and Biology Society, 2008. EMBS 30th Annual International Conference of the IEEE, pp.3332-3335, 2008

[82] Noury, N., Barralon, P., Virone, G., Boissy, P., Hamel, M., and Rumeau, P.,"A smart sensor based on rules and its evaluation in daily routines", Engineeringin Medicine and Biology Society , Proceedings of the 25th Annual International Conferenceofthe IEEE,Vol. 4, pp.3286-3289. 2003.

[83] Noury, N., Herve, T., Rialle, V., Virone, G., Mercier, E., Morey, G., Moro, A. and Porcheron, T, "Monitoring behavior in home using a smart fall sensor and position sensors," Microtechnologies in Medicine and Biology, 1st Annual International,Conference On., pp. 607-610, 2000

[84] 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," ISWC'07: Proceedings of the 11th IEEE International Symposiumon Wearable Computers, IEEEComputer Society, Washington, DC, USA, pp.154, 2007.

[85] Hwang, J., Kang, J., Jang,Y. and Kim, H., "Development of novel algorithm and real-time monitoring ambulatory system using Bluetooth module for fall detection in the elderly.," Engineering in Medicineand Biology Society, IEMBS 304. 26th Annual International Conference of the IEEE, Vol.1, pp.2204-2207, 2004.

[86] Degen, T., Jaeckel, H., Rufer, M., and Wyss, S., "SPEEDY: A fall detector in a wrist watch," Proceedings of the 7th IEEE International Symposium on Wearable Computers, pp. 184-187, 2003.

[87] https://www.academia.edu/11716277/Fall_detection_through_vertical_velocity_thresholding_using_a_tri-axial_accelerometer_characterized_using_an_optical_motion-capture_system [accessed last on 19 June, 2016]

[88] https://jneuroengrehab.biomedcentral.com /articles/10.1186/1743-0003-9-21 [accessed last on 19 June, 2016]

[89] https://www.shutterstock.com/search/gyroscope+vector [accessed last on 19 June, 2016]

[90] http://stackoverflow.com/questions/35227321/how-to-know-if-a-vector-on-the-right-hand- side [accessed last on 19 June, 2016]

[91] http://www.instructables.com/id/Accelerometer-Gyro-Tutorial/ [accessed last on 19 June, 2016]

[92] http://courses.media.mit.edu/2006fall/mas622j/Projects/aisen-project/ [accessed last on 19 June, 2016]

[93] http://stackoverflow.com/questions/9480605/what-is-the-relation- between- the-number- of-support-vectors-and-training-data-and [accessed last on 19 June, 2016]

[94] http://7xt8es.com1.z0.glb.clouddn.com/zhimind/ml/margin_sv.png [accessed last on 19 June, 2016]

[95] http://user.it.uu.se/~kostis/Teaching/DM-05/Slides/classification01.pdf [accessed last on 19 June, 2016]

Publication

1. Farhad Hossain, Liakot Ali, Zahurul Islam and Hossen A Mustafa, “A Direction-Sensitive Fall Detection System Using Single 3D Accelerometer and Learning Classifier”, In the Proceedings of the International Conference on Medical Engineering, Health Informatics and Technology (MediTec), Dec 2016.

Appendix A: SIMULATION OF FALL DETECTION

The following code generates the direction sensitive fall detection as described in the thesis.

```
function saveSensorDataAsMATFiles


if exist('rawSensorData_train.mat','file') && exist('rawSensorData_test.mat','file')
    fprintf(1,'rawSensorData_train.mat and rawSensorData_test.mat already exists at
location:\n');
    disp(['* ', which('rawSensorData_train.mat')]);
    disp(['* ', which('rawSensorData_test.mat')]);
    disp(' ')
else
    %% Load training data from files
    activity_labels = {'Forward Fall','WUs','WDs','Backward Fall','Left Fall','Right Fall'};
    trainActivity = categorical(importdata('D:\Fall
Detection\Fall_ADL_DataSet\Train\y_train.txt'),1:6,activity_labels);
    trainActivity = mergecats(trainActivity,{'WUs','WDs'},'Activities of Daily Living');
    trainActivity = reordercats(trainActivity ,{'Right Fall','Backward Fall','Activities of Daily
Living','Left Fall','Forward Fall'});

    filestoload = strcat('D:\Fall Detection\Fall_ADL_DataSet\Train\',{'total*'});



    disp('Loading training data from files:')
    try
        dstrain = datastore(filestoload,'TextscanFormats',repmat({'%f'},1,128),
'ReadVariableNames',false);
    catch err
        if strcmp(err.identifier,'MATLAB:datastoreio:pathlookup:fileNotFound')
            error('File not found. Please make sure that you download and extract the data first
using "downloadSensorData" function')
        end
    end
    [~,fnames] = cellfun(@fileparts,dstrain.Files,'UniformOutput',false);
    iter = 1;
    while hasdata(dstrain)
        fprintf('Importing: %16s ...',fnames{iter})
        M = table2array(read(dstrain));
        rawSensorDataTrain.(fnames{iter}) = M;
        iter = iter + 1;
        fprintf('Done\n')
    end
    rawSensorDataTrain.trainActivity = trainActivity;
    disp(' ')
    %% Load test data from files
    testActivity = categorical(importdata('D:\Fall
Detection\Fall_ADL_DataSet\Test\y_test.txt'),1:6,activity_labels);
    testActivity = mergecats(testActivity,{'WUs','WDs'},'Activities of Daily Living');
    testActivity = reordercats(testActivity ,{'Right Fall','Backward Fall','Activities of Daily
Living','Left Fall','Forward Fall'});

    filestoload = strcat('D:\Fall Detection\Fall_ADL_DataSet\Test\',{'total*'});

    disp('Loading test data from files:')
    dstest =
datastore(filestoload,'TextscanFormats',repmat({'%f'},1,128),'DatastoreType','tabulartext',...
        'ReadVariableNames',false);
    [~,fnames] = cellfun(@fileparts,dstest.Files,'UniformOutput',false);
    dstest.ReadSize = 'file';
    iter = 1;
    while hasdata(dstest)
        fprintf('Importing: %16s ...',fnames{iter})
        M = table2array(read(dstest));
        rawSensorDataTest.(fnames{iter}) = M;
        iter = iter + 1;
        fprintf('Done\n')
    end
    rawSensorDataTest.testActivity = testActivity;
disp(' ')
    %% Saving MAT file with raw data
    fprintf('Saving MAT files: rawSensorData_train.mat ...')
    save rawSensorData_train.mat -struct rawSensorDataTrain
    disp('Done')
    fprintf('Saving MAT files: rawSensorData_test.mat ...')
    save rawSensorData_test.mat -struct rawSensorDataTest
    disp('Done')
end


% A Matlab code for Function Building
function Y = Wmean(X)

    Y = mean(X,2);
end
      ------------------------------------------------------------

function Y = Wmedian(X)

    Y = median(X,2);
end
      --------------------------------------------------------------


function Y = Wpca1(X)

    [~,Y] = pca(X,'NumComponents',1);
end
      ------------------------------------------------------------------


function Y = Wstd(X)
```

```
    Y = std(X,[],2);
end
      ------------------------------------------------------------------------

% GUI of fall simulation
function plotActivityResults(mdl,rawSensorDataTest,humanActivityTest,delay)

if nargin < 4
delay = 0.02;
end

time = linspace(0,2.56,128);

fig = figure('Name','Human Fall Detection','NumberTitle','off','Visible','off');
fig.Position(3:4) = 600;
movegui('center')
fig.Visible = 'on';

 ax1 = subplot(2,1,1,'Parent',fig,'Xgrid','on','Ygrid','on',...
'XLim',[time(1) time(end)],'YLim',[-45 35]);

clr = get(groot,'DefaultAxesColorOrder');
L(1) =
line(time,rawSensorDataTest.total_acc_x_test(1,:),'color',clr(1,:),'Parent',ax1,'LineWidth',1.5,'
DisplayName','Accelerometer X');
L(2) =
line(time,rawSensorDataTest.total_acc_y_test(1,:),'color',clr(2,:),'Parent',ax1,'LineWidth',1.5,'
DisplayName','Accelerometer Y');
L(3) =
line(time,rawSensorDataTest.total_acc_z_test(1,:),'color',clr(5,:),'Parent',ax1,'LineWidth',1.5,'
DisplayName','Accelerometer Z');

xlabel(ax1,'Time (s)')
ylabel(ax1,'(Accelerometer Readings (m \cdot s^{-2})')
%legend(ax1,'show')
legend(ax1,'show','Location','northwest','Orientation','horizontal')
title(ax1,['Fall Detection System: ', getClassifierName(mdl)]);



ann1 = annotation(fig,'textbox',[ax1.Position(1:3) 0.04],...
'String','Algorithms Predicted Activity : NA','FontSize',12,'FitBoxToText','off',...
'BackgroundColor',[0 0.7
0.3],'HorizontalAlignment','Center','VerticalAlignment','middle','FaceAlpha',0.5);
ann2 = annotation(fig,'textbox',[ax1.Position(1) ax1.Position(2)+0.04 ax1.Position(3) 0.04],...
'String','Previous Recorded Activity : NA','FontSize',12,'FitBoxToText','off',...
'BackgroundColor',[0 0.7
0.3],'HorizontalAlignment','Center','VerticalAlignment','middle','FaceAlpha',0.5);

%% Loop through the raw data and plot the sensor values
try
for ii = 1:height(humanActivityTest)
mycell1 = fieldnames(mdl);
myclassifier1 = strcat('mdl.',mycell1(3));
activity = predict(eval(myclassifier1{:}),humanActivityTest{ii,1:end-1});



if activity == humanActivityTest.activity(ii)

   predclr = [0 0.7 0.3];
else
predclr = [1 0 0];
end
set(ann1,'String',['Algorithms Predicted Result : ' char(activity)],...
'BackgroundColor',predclr);
set(ann2,'String',['Previous Known Record : ' char(humanActivityTest.activity(ii))],...
'BackgroundColor',[0 0.7 0.3]);

L(1).YData = rawSensorDataTest.total_acc_x_test(ii,:);
```

```
L(2).YData = rawSensorDataTest.total_acc_y_test(ii,:);
L(3).YData = rawSensorDataTest.total_acc_z_test(ii,:);

drawnow
pause(delay)
end
catch err
end

function cname = getClassifierName(trainedClassifier)
mycell = fieldnames(trainedClassifier);
cname = mycell(3);


%Algorithm learning and export model
function [trainedClassifier, validationAccuracy] = trainClassifier(trainingData)
% trainClassifier(trainingData)
% classifier.
inputTable = trainingData;
predictorNames = {'Wmean_total_acc_x_train', 'Wmean_total_acc_y_train',
'Wmean_total_acc_z_train', 'Wstd_total_acc_x_train', 'Wstd_total_acc_y_train',
'Wstd_total_acc_z_train', 'Wpca1_total_acc_x_train', 'Wpca1_total_acc_y_train',
'Wpca1_total_acc_z_train'};
predictors = inputTable(:, predictorNames);
response = inputTable.activity;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
template = templateSVM(...
    'KernelFunction', 'polynomial', ...
    'PolynomialOrder', 2, ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    predictors, ...
    response, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', categorical({'Right Fall'; 'Backward Fall'; 'Activities of Daily Living'; 'Left
Fall'; 'Forward Fall'}, {'Right Fall' 'Backward Fall' 'Activities of Daily Living' 'Left Fall'
'Forward Fall'}));

% Create the result struct with predict function
predictorExtractionFcn = @(t) t(:, predictorNames);
svmPredictFcn = @(x) predict(classificationSVM, x);
trainedClassifier.predictFcn = @(x) svmPredictFcn(predictorExtractionFcn(x));

% Add additional fields to the result struct
trainedClassifier.RequiredVariables = {'Wmean_total_acc_x_train',
'Wmean_total_acc_y_train', 'Wmean_total_acc_z_train', 'Wstd_total_acc_x_train',
'Wstd_total_acc_y_train', 'Wstd_total_acc_z_train', 'Wpca1_total_acc_x_train',
'Wpca1_total_acc_y_train', 'Wpca1_total_acc_z_train'};
trainedClassifier.ClassificationSVM = classificationSVM;
trainedClassifier.About = 'This struct is a trained classifier exported from Classification
Learner R2016a.';
trainedClassifier.HowToPredict = sprintf('To make predictions on a new table, T, use: \n  yfit
= c.predictFcn(T) \nreplacing "c" with the name of the variable that is this struct, e.g.
"trainedClassifier". \n \nThe table, T, must contain the variables returned by: \n
c.RequiredVariables \nVariable formats (e.g. matrix/vector, datatype) must match the original
training data. \nAdditional variables are ignored. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, "stats", "stats.map"),
"appclassification_exportmodeltoworkspace")">How to predict using an exported
model</a>.');

% classifier.
inputTable = trainingData;
predictorNames = {'Wmean_total_acc_x_train', 'Wmean_total_acc_y_train',
'Wmean_total_acc_z_train', 'Wstd_total_acc_x_train', 'Wstd_total_acc_y_train',
'Wstd_total_acc_z_train', 'Wpca1_total_acc_x_train', 'Wpca1_total_acc_y_train',
'Wpca1_total_acc_z_train'};
predictors = inputTable(:, predictorNames);
response = inputTable.activity;
isCategoricalPredictor = [false, false, false, false, false, false, false, false, false];

% Set up holdout validation
cvp = cvpartition(response, 'Holdout', 0.2);
trainingPredictors = predictors(cvp.training,:);
trainingResponse = response(cvp.training,:);
trainingIsCategoricalPredictor = isCategoricalPredictor;

% Train a classifier
% This code specifies all the classifier options and trains the classifier.
```

```
template = templateSVM(...
    'KernelFunction', 'polynomial', ...
    'PolynomialOrder', 2, ...
    'KernelScale', 'auto', ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    trainingPredictors, ...
    trainingResponse, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', categorical({'Right Fall'; 'Backward Fall'; 'Activities of Daily Living'; 'Left
Fall'; 'Forward Fall'}, {'Right Fall' 'Backward Fall' 'Activities of Daily Living' 'Left Fall'
'Forward Fall'}));

% Create the result struct with predict function
svmPredictFcn = @(x) predict(classificationSVM, x);
validationPredictFcn = @(x) svmPredictFcn(x);



% Compute validation accuracy
validationPredictors = predictors(cvp.test,:);
validationResponse = response(cvp.test,:);

[validationPredictions, validationScores] = validationPredictFcn(validationPredictors);
correctPredictions = (validationPredictions == validationResponse);
validationAccuracy = sum(correctPredictions)/length(correctPredictions);


%Computing the features and run program

%Human Fall Detection System as well as Direction
 %% Human Directive Fall Detection  Using
%Raw accelerometer sensor data &  Learning Algorithm
%Directive Fall (Forward Backward, Left, Right).
%The goal of this thesis is to build a classifier that can  automatically
%identify the Fall from ADL as well as Fall Direction.


%% Description of the Data
%  The dataset consists of accelerometer data captured at 50Hz.
%  The raw sensor data contain fixed-width sliding windows of 2.56 sec
% (128 readings/window).

% * |saveSensorDataAsMATFiles| : This will create two MAT files:
% |rawSensorData_train|  and |rawSensorData_test| with the raw sensor data

if ~exist('rawSensorData_train.mat','file') && ~exist('rawSensorData_test.mat','file')
    saveSensorDataAsMATFiles;
end

%Load Train Data
load rawSensorData_train

%Display data summary
% 923(Train data) will be changed w.r to data number
plotRawSensorData(total_acc_x_train, total_acc_y_train, ...
    total_acc_z_train,trainActivity,923);

%Create Table variable for train data
rawSensorDataTrain = table(...
    total_acc_x_train, total_acc_y_train, total_acc_z_train);

%Extract features from Train raw sensor data
T_mean = varfun(@Wmean, rawSensorDataTrain);
T_stdv = varfun(@Wstd,rawSensorDataTrain);
T_pca  = varfun(@Wpca1,rawSensorDataTrain);



humanActivityData = [T_mean, T_stdv, T_pca];
humanActivityData.activity = trainActivity;

%T_medn  = varfun(@Wmedian,rawSensorDataTrain);
%humanActivityData = [T_mean, T_stdv, T_medn];
```

```
%classificationLearner calling,learning & export model
classificationLearner



%Load Test Data (476 test data)
load rawSensorData_test

% Create Table variable for test data
rawSensorDataTest = table(...
    total_acc_x_test, total_acc_y_test, total_acc_z_test);

% Extract features from Test raw sensor data
T_mean = varfun(@Wmean, rawSensorDataTest);
T_stdv = varfun(@Wstd,rawSensorDataTest);
T_pca  = varfun(@Wpca1,rawSensorDataTest);

humanActivityData = [T_mean, T_stdv, T_pca];
humanActivityData.activity = testActivity;

%T_medn  = varfun(@Wmedian,rawSensorDataTest);
%humanActivityData = [T_mean, T_stdv, T_medn];

%Using trained model(called trainedClassifier),
%Features and Test raw sensor data
%calling |*plotActivityResults| function & Show the result

plotActivityResults(trainedClassifier,rawSensorDataTest,humanActivityData,0.1)
```

Appendix B: REAL TIME IMPLEMENT

```
%  Human Fall Detection System as well as Direction

if ~exist('rawSensorData_train.mat','file')
        saveSensorDataAsMATFiles;
end

%Load Train Data
load rawSensorData_train

%Create Table variable for train data
rawSensorDataTrain = table(...
    total_acc_x_train, total_acc_y_train, total_acc_z_train);

%Extract features from Train raw sensor data
T_mean = varfun(@Wmean, rawSensorDataTrain);
T_stdv = varfun(@Wstd,rawSensorDataTrain);
T_pca  = varfun(@Wpca1,rawSensorDataTrain);

humanActivityData = [T_mean, T_stdv, T_pca];
humanActivityData.activity = trainActivity;

%classificationLearner calling,learning & export model
classificationLearner


% Enable Accelerometer Sensor


classdef SetEnabledSensorsMacrosClass < handle


    properties (Constant = true)
        ACCEL='Accel';   % Accelerometer; for Shimmer3 Low Noise Accelerometer will be
selected.
        LNACCEL='LowNoiseAccel';       % Low Noise Accelerometer for Shimmer3
        WRACCEL='WideRangeAccel';      % Wide Range Accelerometer for Shimmer3
        ALTACCEL='AlternativeAccel';    % MPU9150 Accelerometer for Shimmer3
        GYRO = 'Gyro';                 % Gyroscope
        MAG = 'Mag';                   % Magnetometer
        ALTMAG = 'AlternativeMag';      % MPU9150 Magnetometer for Shimmer3
        ECG = 'ECG';                   % ECG
        ECG24BIT ='ECG 24BIT';         % ECG 24BIT for Shimmer3
        ECG16BIT ='ECG 16BIT';         % ECG 16BIT for Shimmer3
        EMG = 'EMG';                   % EMG
        EMG24BIT = 'EMG 24BIT';        % EMG 24BIT for Shimmer3
        EMG16BIT = 'EMG 16BIT';        % EMG 16BIT for Shimmer3
        EXG1 = 'EXG1';                 % EXG1 for Shimmer3
        EXG124BIT = 'EXG1 24BIT';       % EXG1 24BIT for Shimmer3
        EXG116BIT = 'EXG1 16BIT';       % EXG1 16BIT for Shimmer3
        EXG2 = 'EXG2';                  % EXG2 for Shimmer3
        EXG224BIT = 'EXG2 24BIT';       % EXG2 24BIT for Shimmer3
        EXG216BIT = 'EXG2 16BIT';       % EXG2 16BIT for Shimmer3
        GSR = 'GSR';                    % GSR
        EXPA0 = 'ExpBoard_A0';          % External Expansion Board A0 for Shimmer2r
        EXPA7 = 'ExpBoard_A7';          % External Expansion Board A7 for Shimmer2r
        EXTA7 = 'EXT A7';               % External ADC A7 for Shimmer3
        EXTA6 = 'EXT A6';               % External ADC A6 for Shimmer3
        EXTA15 = 'EXT A15';             % External ADC A15 for Shimmer3
        STRAIN = 'Strain Gauge';        % Strain Gauge for Shimmer2r
        BRIDGE = 'Bridge Amplifier';    % Bridge Amplifier for Shimmer3
        HEART = 'Heart Rate';           % Heart Rate for Shimmer2r
        BATT = 'BattVolt';              % Battery Voltage
        INTA1 = 'INT A1';               % Internal ADC for Shimmer3
        INTA12 = 'INT A12';             % Internal ADC for Shimmer3
        INTA13 = 'INT A13';             % Internal ADC for Shimmer3
        INTA14 = 'INT A14';             % Internal ADC for Shimmer3
        PRESSURE = 'Pressure';          % BMP180 Pressure (and Temperature) for Shimmer3
    end

    %%%%%%%%%%%%%%%%%%%%%%%%
    % Constructor Method

    methods

        function theseMacros = SetEnabledSensorsMacrosClass


        end % function SetEnabledSensorsMacrosClass

    end % methods (Constructor)
end



% Real time data streaming and features calculation

function void = plotandwriteexample(mdl, comPort, captureDuration, fileName)

%%  EXAMPLE: plotandwriteexample(trainedClassifier, '3', 30, 'testdata.dat')

if nargin < 4
delay = 0.04;
end

shimmer = ShimmerHandleClass(comPort);
SensorMacros = SetEnabledSensorsMacrosClass;

DELAY_PERIOD = 0.2;
```

```
nn = 1;
M  = [1.33 2.33 3.33];
M1  =  dlmread('farhad.dat','','B1..D30');
M2  =  dlmread('farhad.dat','','B1..D30');
M3  =  dlmread('farhad.dat','','B1..D30');
M4  =  dlmread('farhad.dat','','B1..D30');
M5  =  dlmread('farhad.dat','','B1..D30');

M = vertcat(M1,M2,M3,M4,M5);
M = M(1:128,1:3);
N = M.';
total_acc_x_test = N(1:1,1:128);
total_acc_y_test = N(2:2,1:128);
total_acc_z_test = N(3:3,1:128);


if (shimmer.connect)

    % Define settings for shimmer
    shimmer.setsamplingrate(51.2);
    shimmer.setinternalboard('9DOF');
    shimmer.disableallsensors;
    shimmer.setenabledsensors(SensorMacros.ACCEL,1);
    shimmer.setaccelrange(0);                                              %

    %GUI define
    time = linspace(0,2.56,128);

    fig = figure('Name','Human Fall Detection','NumberTitle','off','Visible','off');
    fig.Position(3:4) = 600;
    movegui('center')
    fig.Visible = 'on';

    ax1 = subplot(2,1,1,'Parent',fig,'Xgrid','on','Ygrid','on',...
    'XLim',[time(1) time(end)],'YLim',[-45 35]);

    clr = get(groot,'DefaultAxesColorOrder');
    L(1) = 
line(time,total_acc_x_test(1,:),'color',clr(1,:),'Parent',ax1,'LineWidth',1.5,'DisplayName','Acce
lerometer X');
    L(2) = 
line(time,total_acc_y_test(1,:),'color',clr(2,:),'Parent',ax1,'LineWidth',1.5,'DisplayName','Acce
lerometer Y');
    L(3) = 
line(time,total_acc_z_test(1,:),'color',clr(5,:),'Parent',ax1,'LineWidth',1.5,'DisplayName','Acce
lerometer Z');

    xlabel(ax1,'Time (s)')
    ylabel(ax1,'(Accelerometer Readings (m \cdot s^{-2})')
    legend(ax1,'show','Location','northwest','Orientation','horizontal')
    title(ax1,['Real Time Fall Detection System: ']);

    ann1 = annotation(fig,'textbox',[ax1.Position(1:3) 0.04],...
    'String','Algorithms Predicted Activity : NA','FontSize',12,'FitBoxToText','off',...
    'BackgroundColor',[0 0.7
0.3],'HorizontalAlignment','Center','VerticalAlignment','middle','FaceAlpha',0.5);


     if (shimmer.start)
            elapsedTime = 0;
            tic;

        while (elapsedTime < captureDuration)
                pause(DELAY_PERIOD);
                [newData,signalNameArray,signalFormatArray,signalUnitArray] =
shimmer.getdata('c');

            if ~isempty(newData)

                switch nn

                    case 1
                    dlmwrite(fileName, newData, 'delimiter', '\t','precision',6);
                    M1 = dlmread('testdata.dat', '',0,1);
                    M = vertcat(M2,M3,M4,M5,M1);
                    nn = 2;

                    case 2
                    dlmwrite(fileName, newData, 'delimiter', '\t','precision',6);
                    M2 = dlmread('testdata.dat','',0,1);
                    M = vertcat(M3,M4,M5,M1,M2);
                    nn = 3;

                    case 3
                    dlmwrite(fileName, newData, 'delimiter', '\t','precision',6);
                    M3 = dlmread('testdata.dat', '',0,1);
                    M = vertcat(M4,M5,M1,M2,M3);
                    nn = 4;

                    case 4
                    dlmwrite(fileName, newData, 'delimiter', '\t','precision',6);
                    M4 = dlmread('testdata.dat', '',0,1);
                    M = vertcat(M5,M1,M2,M3,M4);
                    nn = 5;

                    case 5
                    dlmwrite(fileName, newData, 'delimiter', '\t','precision',6);
                    M5 = dlmread('testdata.dat', '',0,1);
                    M = vertcat(M1,M2,M3,M4,M5);
                    nn = 1;

                end

                MM = size(M,1)

                if (MM > 127)
                    M = M(1:128,1:3);
                else
                    M  =  dlmread('farhad.dat','','B1..D128');
                end

                N = M.';
                rawSensorDataTest = table(N);
                T_mean = varfun(@Wmean, rawSensorDataTest);
                T_stdv = varfun(@Wstd, rawSensorDataTest);
                T_pca  = varfun(@Wpca1, rawSensorDataTest);

                YourArray = table2array(T_mean);
                YourNewTable1 = array2table(YourArray.');
                YourNewTable1.Properties.VariableNames = {'Wmean_total_acc_x_test'
'Wmean_total_acc_y_test' 'Wmean_total_acc_z_test'};

                YourArray = table2array(T_stdv);
                YourNewTable2 = array2table(YourArray.');
                YourNewTable2.Properties.VariableNames = {'Wstd_total_acc_x_test'
'Wstd_total_acc_y_test' 'Wstd_total_acc_z_test'};

                YourArray = table2array(T_pca);
                YourNewTable3 = array2table(YourArray.');
                YourNewTable3.Properties.VariableNames = {'Wpcal_total_acc_x_test'
'Wpcal_total_acc_y_test' 'Wpcal_total_acc_z_test'};

                C = {'NA'};
                T = cell2table(C,...
                    'VariableNames',{'activity'});

                humanActivityTest = [YourNewTable1, YourNewTable2, YourNewTable3, T];

                pause(0.25);

                total_acc_x_test = N(1:1,1:128);
                total_acc_y_test = N(2:2,1:128);
                total_acc_z_test = N(3:3,1:128);

                %% Loop through the raw data and plot the sensor values
                try
                  for ii = 1:height(humanActivityTest)
                     mycell1 = fieldnames(mdl);
                     myclassifier1 = strcat('mdl.',mycell1(3));
```

```
                    activity = predict(eval(myclassifier1{:}),humanActivityTest{ii,1:end-1});

                     if activity == 'Forward Fall'
                        predclr = [1 0 0];
                        pause(1.01);

                        elseif activity == 'Backward Fall'
                          predclr = [1 0 0];
                          pause(1.01);

                        elseif activity == 'Left Fall'
                           predclr = [1 0 0];
                           pause(1.01);

                        elseif activity == 'Right Fall'
                          predclr = [1 0 0];
                          pause(1.01);

                        else
                          predclr = [0 0.7 0.3];
                     end


                  set(ann1,'String',['Directive  Fall Detection : ' char(activity)],...
                     'BackgroundColor',predclr);

                   L(1).YData = total_acc_x_test;
                   L(2).YData = total_acc_y_test;
                   L(3).YData = total_acc_z_test;

                   drawnow
                   %pause(0.1)
                  end

                catch err
                end

          end
           elapsedTime = elapsedTime + toc;
           tic;

      end
       elapsedTime = elapsedTime + toc;
       shimmer.stop;

   end
    shimmer.disconnect;

   end
```

```
% Call classifier to detect real time directive fall detection
% Calling the function to detect real-time fall detection system.

plotandwriteexample(trainedClassifier, '3', 30, 'testdata.dat')


%Developed by: Shaikh Farhad Hossain, MSc Engg. (IICT, BUET), Bangladesh
                     %Email to:  farhadcse05@yahoo.com
```