THESIS

APPLICATIONS OF INERTIAL MEASUREMENT UNITS IN MONITORING REHABILITATION PROGRESS OF ARM IN STROKE SURVIVORS

Submitted by

Saket Sham Doshi

Department of Electrical and Computer Engineering

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2011

Master's Committee:

Advisor: Anura P. Jayasumana
Co-Advisor: Matthew P. Malcolm

Sudeep Pasricha
Yashwant K. Malaiya

ABSTRACT

APPLICATIONS OF INERTIAL MEASUREMENT UNITS IN MONITORING REHABILITATION PROGRESS OF ARM IN STROKE SURVIVORS

Constraint Induced Movement Therapy (CIMT) has been clinically proven to be effective in restoring functional abilities of the affected arm among stroke survivors. Current CIMT delivery method lacks a robust technique to monitor rehabilitation progress, which results in increasing costs of stroke related health care. Recent advances in the design and manufacturing of Micro Electro Mechanical System (MEMS) inertial sensors have enabled tracking human motions reliably and accurately. This thesis presents three algorithms that enable monitoring of arm movements during CIMT by means of MEMS inertial sensors.

The first algorithm quantifies the affected arm usage during CIMT. This algorithm filters the arm movement data, sampled during activities of daily life (ADL), by applying a threshold to determine the duration of affected arm movements. When an activity is performed multiple times, this algorithm counts the number of repetitions performed. Current technique uses a touch/proximity sensor and a motor activity log maintained by the patient to determine CIMT duration. Affected arm motion is a direct indicator of CIMT session and hence this algorithm tracks rehabilitation progress more accurately. Actual patients' affected arm movement data analysis shows that the algorithm does activity detection with an average accuracy of >90%.

Second of the three algorithms, tracking stroke rehabilitation of affected arm through histogram of distance traversed, evaluates an objective metric to assess rehabilitation progress. The objective metric can be used to compare different stroke patients based on their functional ability in affected arm. The algorithm calculates the histogram by evaluating distances traversed over a fixed duration window. The impact of this window on algorithm's performance is analyzed. The algorithm has better temporal resolution when compared with another standard objective test, box and block test (BBT). The algorithm calculates linearly weighted area under the histogram as a score to rank various patients as per their rehabilitation progress. The algorithm has better performance for patients with chronic stroke and certain degree of functional ability.

Lastly, Kalman filter based motion tracking algorithm is presented that tracks linear motions in 2D, such that only one axis can experience motion at any given time. The algorithm has high (>95%) accuracy. Data representing linear human arm motion along a single axis is generated to analyze and determine optimal parameters of Kalman filter. Cross-axis sensitivity of the accelerometer limits the performance of the algorithm over longer durations. A method to identify the 1D components of 2D motion is developed and cross-axis effects are removed to improve the performance of motion tracking algorithm.

ACKNOWLEDGEMENTS

I would like to thank all the individuals whose encouragement and support has made the completion of this thesis possible.

First, I would like to express my thanks and gratitude to Prof. Anura P. Jayasumana for believing in my abilities and offering me all the help and encouragement needed. I am indebted by the support and guidance offered by Prof. Anura Jayasumana, which has helped me become a better researcher. Your advice and guidance will always be cherished throughout my professional and personal life. Thank you, Prof. Anura Jayasumana, for offering me an opportunity to work with you.

Special thank must go to Professor Matthew Malcolm, co-advisor for this work, for his sound advice and guidance that helped in completion of this complex interdisciplinary work. I express my sincere gratitude for the time and resources provided by Prof. Matthew Malcolm without which this work would not have completed.

I would like to express my sincere gratitude to Prof. Louis Scharf for his invaluable advice on Kalman filter. I would also like to thank my committee member Prof. Yashwant Malaiya and Prof. Sudeep Pasricha for their time and support offered. My sincere gratitude to Prof. Sudeep Pasricha for agreeing to be on my thesis committee over such a short notice.

I would like to thank Crystal Massie and Kari Greteman along with all other student staff members of NeuroRehabilitaion Research Laboratory (NRRL) of

Occupational Therapy Department for their invaluable help in conducting patient trials. Without these data trials, the impact of this work would have been insignificant. I am grateful to my colleagues at Computer Network Research Laboratory (CNRL) Dilum Bandara, M. N. Raghunandan, and Dulanjalie Dhanapala for their unwavering support and invaluable suggestions regarding my work. Special thanks must go to Vidarshana Bandara for his thorough review of this thesis.

I would also like to thank my colleagues from Mechanical Engineering, Guhan Srivatsan and Amit Munshi, for their immense help in the construction of wearable device and sensor characterization respectively. I express my heartfelt gratitude my peers and colleagues at CSU Saurabh Deshpande, Pranay Sanadhya, Swaroop Sahoo, and my colleagues at CNRL who participated in sensor data trials that formed the fundamental and critical part of this work. I would like to thank all my peers and colleagues at CSU, who either directly or indirectly contributed to completion of this work, for all their support and encouragement.

Last, but never the least, I am indebted by the love, support, and encouragement offered by my parents and brother without which none of this work would have been possible.

To my parents, Sham and Kalpana,

and

To my brother, Sachin

Without their support, understanding, encouragement, and love this work would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Each year about 795,000 people suffer a new or recurrent stroke in the United States [3], which results in direct and indirect health care costs totaling $73.7 billion [3]. Over 137,000 of these people die making stroke the third leading cause of death. About 5.7 million U.S. stroke survivors are alive today, many of them with permanent stroke-related disabilities [2]. Partial paralysis (medically known as hemiparesis) is one of the most common effects of stroke that survivors have to live with. Up to 85% of the stroke survivors experience partial paralysis, resulting in impairment of an upper extremity immediately after stroke, and between 55% and 75% of survivors continue to experience upper extremity functional limitations [52] that are associated with diminished health-related quality of life [29], even after 3 to 6 months [20].

To improve the quality of life it is imperative that the survivor undergo rehabilitation therapy to regain partial use of their paralyzed limb. Currently, there are various rehabilitation techniques available for stroke survivors. Neurodevelopment techniques, therapeutic techniques such as repetitive task specific training, sensorimotor training with robotic devices, constraint-induced movement therapy, and virtual reality therapy are some of the examples of rehabilitation techniques currently being used. The physiological mechanisms, through which these therapies result in a beneficial improvement in limb function, are not well understood. However, these therapies have been shown to be effective in patients [10].

Most of these therapies are carried out under the supervision of a trained staff within a laboratory environment. Most of them rely on the subjective evaluation of the effectiveness of the therapy by the same trained staff. To improve the efficiency with which these therapies are delivered as well as to improve the overall therapy by finding its efficacy objectively, researchers

have started to look at various methods by which they can monitor the improvement in the affected limb objectively. Computer software, sensors, and robots are some of the means being used to achieve this purpose.

The advances in the electronic sensing in general and micro-electro mechanical systems (MEMS) technology in particular have enabled a new era of compact, accurate, power efficient, and wearable sensors which can be attached to various parts of human body to measure quantities of interest. In past few years, there has been considerable research [1], [44], [24], [26] validating the use of MEMS based inertial sensors such as accelerometer, gyroscope, and magnetometer to monitor the therapy outcome objectively.

1.1 Motivation

Constraint induced movement therapy (CIMT) has been shown to be effective in arm rehabilitation of stroke survivors in controlled studies. The therapy is based on carrying out repetitive movements involving activities of daily life (ADL) using the affected arm. To track rehabilitation progress the therapy relies on behavioral contract with the participant, which requires the participants to keep track of their affected arm use. Based on these records and some functional test(s) rehabilitation progress is assessed. This form of administering the therapy is not efficient as the assessment is highly subjective. The therapy needs to be administered for some minimum time (from few weeks to months) for the assessment to be reliable. Thus, there is an opportunity to make this therapy more efficient and MEMS sensors lend themselves well for this purpose. An objective method to track rehabilitation progress will make CIMT more robust. Part of the inefficiency associated with CIMT is the lack of an accurate and robust method to monitor rehabilitation progress remotely. With advances in manufacturing of MEMS devices, constructing a wearable device utilizing MEMS sensors has become easier. With such wearable sensor device, a therapist can monitor the rehabilitation progress while the patient is not under his direct supervision. This will reduce the number of visits to therapist's clinic and effectively reducing stroke related health care costs.

1.2 Contributions

Over past few years, there has been a consistent effort to make CIMT more efficient. In CIMT, the affected arm use is encouraged by constraining the unaffected arm with the help of a glove (mitt) or a sling. Currently, wearing a constraint on unaffected arm is assumed to represent the affected arm use, which may not necessarily be true all the time. The log maintained by the participant and/or by some touch-based electronic sensor without any signal processing has been used to establish this relation.

The thesis presents an algorithm implemented using a MEMS three-dimensional (3D) accelerometer that can identify the intervals when the affected arm is being used. The algorithm is verified in controlled studies involving a set of activities done inside the laboratory. Thus, this method gives an accurate measure of when the affected arm is being used rather than just telling us whether the constraint is worn or not.

The greater goal of this work is to come up with some objective measure/metric, which can reflect the rehabilitation progress. The thesis proposes two solutions in this regard. The first solution involves calculating the histogram of distances covered during a pre-defined interval and using it to evaluate the objective metric for tracking rehabilitation progress of stroke-affected arm. The affected arm movements are intermittent as opposed to the fluent movements done with the unaffected arm. Affected arm movements take longer times to finish as compared with those of unaffected arm. Thus, the histogram of distances covered by the affected arm will be dominated by small distances while as that of unaffected arm will be dominated by large distances. As the participant progresses in the rehabilitation program his histogram profile for affected arm tend towards that of unaffected arm. The thesis presents performance metrics for this approach and compares them with those that are being used currently.

The second solution for tracking the rehabilitation progress is to track the affected arm motion in 2D using inertial measurement unit (IMU), under the constraint that at any point of time only one axis can experience motion. A typical IMU has accelerometers, gyroscopes and/or

magnetometers. Sensor fusion and Kalman filtering form the core of the motion-tracking algorithm. With motion-tracking algorithm, there is no need to calculate a performance metric as the movement itself tells the therapist everything to assess rehabilitation progress.

1.3 Outline

Rest of the thesis is organized as follows. Chapter 2 discusses CIMT in detail and describes the MEMS sensors with a primary focus on common sources of errors affecting these sensors. Chapter 3 reviews the current work related to improving efficiency of CIMT using sensor technology and describes the problem statement for the thesis. Chapter 4 presents the algorithm that finds the interval of activity using 3D accelerometers. Chapter 5 discusses tracking rehabilitation progress by calculating the histogram of distances covered in a pre-defined time interval. Chapter 6 presents the motion tracking solution to the problem. Chapter 7 concludes with summary and future work. The appendix offers the source code of all algorithms implemented in MATLAB environment.

Chapter 2

Background Information

This chapter introduces two of the most important concepts this thesis is based upon. It is crucial that the reader understands these concepts, as they will be referred throughout the thesis quite often. The first subsection introduces the reader to CIMT with all the necessary medical background information. The second subsection introduces the MEMS sensors in general and common sources of error that affect their performance, in particular.

2.1 Constraint Induced Movement Therapy (CIMT)

The EXCITE clinical trial [52] proved the efficacy of CIMT for stroke rehabilitation and is the primary source of information given in this section. Traditional methods for stroke rehabilitation, such as neurodevelopment techniques [6], have not shown to be effective enough in controlled studies. However, recent approaches involving repetitive training of paretic (paralyzed) arm on task-oriented activities, has been shown to be effective among stroke survivors with some functional ability in their paretic arm [5], [8].

One such approach involves intense functionally oriented task practice of the affected (paralyzed) arm while restraining the unaffected arm with the use of a special kind of mitt (glove) or putting unaffected arm in a sling. This approach encourages use of the affected arm in ADLs [40]. This approach is thought to help overcome what Taub [38] first described in a deafferented monkey model as “learned nonuse” of the affected upper arm. It has shown substantial evidence of being effective with individuals having long-term stroke disabilities (>1 year after the occurrence of stroke). This treatment, without supervised task practice, is referred to as “forced use” and has been applied to long-term [51], [45], [33] and subacute [21] stroke patients.

CIMT involves ipsilesional limb restraint with training of affected arm use conducted by a clinician. This training is based on shaping (adaptive task practice) and repetitive task practice principles [50], [41], [30]. Each day, participants receive training for up to 6 hours a day. Shaping is based on the principles of behavioral training [P, Q] that can also be described in terms of motor learning derived from adaptive or part-task practice [R, S]. Standard task practice is less structured (i.e., repetition of tasks is not considered as individual discrete activities), and involves functional activities performed continuously for a period of 15 to 20 minutes (e.g., eating and writing) as anyone would do in daily life.

While the participants are in the research laboratory, they wear the restraining mitt consistently. To enhance mitt use outside of the laboratory, behavioral techniques described in detail in [49], [27] are employed. Behavioral contract, caregiver contract, mitt compliance device, and daily schedule are some of the techniques that have been used earlier. After each in-lab therapy session, patients are encouraged to practice two to three tasks daily at home. The therapy performed outside the lab is monitored regularly via a physical sensor and timer placed in the mitt and by a log of activities maintained by the participant ('home diary'). When patient activity log reports do not match with outputs from the mitt monitoring device, the discrepancy is pointed out to patients and they are asked to adhere to the protocol as accurately as possible.

2.2 MEMS Inertial Sensors (MIS)

Oliver J. Woodman's technical report on inertial navigation [53] introduces MEMS sensors and common error sources affecting their performance. This section is based on that introduction. For a thorough and detailed understanding of MEMS sensors and inertial navigation systems, we recommend the reader to read that technical report.

Inertial sensors measure the inertial quantities of an object, which can help us understand the motion of the object. Some of the examples are accelerometers measuring linear acceleration, gyroscopes measuring angles (and hence the orientation in 3D), and rate gyroscopes measuring the angular velocity. Until recently, majority of these sensors were based on mechanical systems

thus were very accurate and reliable as compared with any other technology. However, these mechanical sensors were too bulky to be worn for human motion tracking. Recent advances in the construction of MEMS devices have made it possible to manufacture small and light inertial navigation systems. These devices offer ruggedness and low-cost that has widened the range of possible applications to include areas such as human and animal motion capture [53]. Advantages of MEMS sensors include [42]:

- small size
- low weight
- rugged construction
- low power consumption
- short start-up time
- inexpensive to produce (in high volumes)
- high reliability
- low maintenance
- compatible with operations in hostile environments

However, the main disadvantage of MEMS sensors is that they are not as accurate as those manufactured using traditional techniques, although the performance of MEMS sensors is improving rapidly.

2.2.1 Linear Accelerometers

Accelerometers can be broadly classified in two classes – mechanical and solid state device. A mechanical accelerometer consists of a mass (m) suspended by springs, as shown in Figure 2.1. The displacement of the mass is measured using a displacement pick-off, giving a signal that is proportional to the force (F) acting on the mass in the direction of the input axis. Newton's second law of motion, $F = ma$, is then used to calculate the acceleration acting on the device.

Solid-state accelerometers can be divided into various sub-groups, including surface acoustic wave (SAW), vibratory, silicon, and quartz devices. MEMS based accelerometers are of two types – either the mechanical type or those that measure the change in frequency of a vibrating element caused by a change of tension, as in SAW accelerometers, shown in Fig. 2.2.

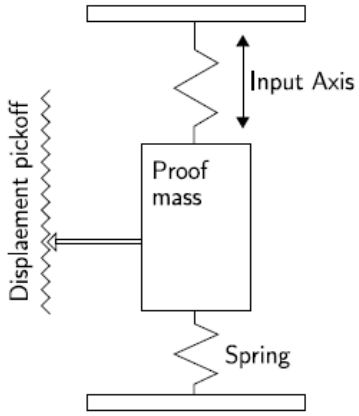

Fig. 2.1 Mechanical Accelerometer [53]

The majority of inaccuracies in the MEMS accelerometers arise because of its manufacturing technique. Some of these error sources are inherent for any silicon device while some of them are typical of MEMS devices. Another issue limiting applicability of MEMS accelerometers is that most applications of accelerometers involve calculating the parameters of motion such as position, velocity, and orientation from acceleration measurements. This is achieved by single/double integration of the accelerometer output. However, this results in inaccuracies due to the accumulation of errors.

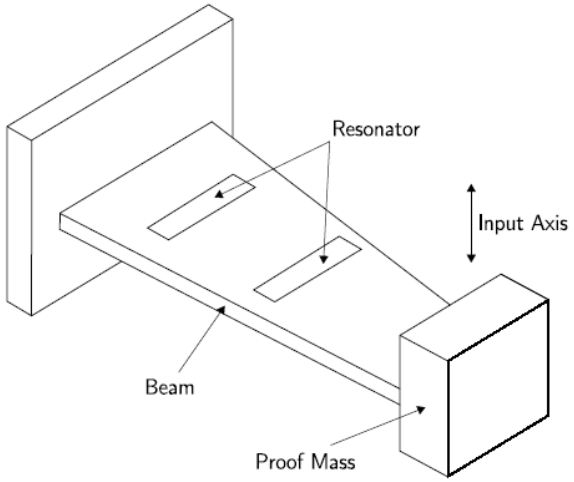

Fig. 2.2 Surface Acoustic Wave Accelerometer [53]

2.2.1.1 Constant Bias

The bias of an accelerometer is the offset of its output signal from the true value, in m/s^2. The bias is dependent on battery supply variations. Usually, this value stays same over an extended duration. A constant bias error of ε, when double integrated, causes an error in position, which grows quadratically with time. The accumulated error in position is

$$s(t) = \varepsilon . \frac{t^2}{2} \tag{2.1}$$

where *t* is the time of integration [53].

2.2.1.2 Thermo-Mechanical White Noise / Velocity Random Walk

All semiconductor integrated circuits experience temperature dependent white noise. As MEMS devices are manufactured using similar technology and they have mechanical moving parts, thermo-mechanical white noise corrupts the sensor output. On integrating, this white noise will produce a random walk in velocity. A random walk is defined as a process consisting of a series of steps, in which the direction and size of each step is randomly determined. To find out the effect, on the calculated position, of the white noise on accelerometer output we need to double integrate the samples from accelerometer output. The analysis presented in [53] follows next. Let N_i be the *i*-th random variable in the white noise sequence, with $E(N_i) = E(N) = 0$ and

Var(N_i) = Var(N) = σ^2. The result of double integrating the white noise signal $\varepsilon(t)$ over a time span t = n·δt is

$$\iint_0^t \varepsilon(\tau)d\tau\, d\tau = \delta t \sum_{i=1}^{n} \delta t \sum_{j=1}^{n} N_j = \delta t^2 \sum_{i=1}^{n}(n-i+1)N_i \tag{2.2}$$

where n is the number of samples received from the device during the period and δt is the time between successive samples. The expected error in position is

$$E\left(\iint_0^t \varepsilon(\tau)d\tau\, d\tau\right) = \delta t^2 \sum_{i=1}^{n}(n-i+1)E(N_i) = 0 \tag{2.3}$$

and the variance is

$$\begin{aligned} Var\left(\iint_0^t \varepsilon(\tau)d\tau\, d\tau\right) &= \delta t^4 \sum_{i=1}^{n}(n-i+1)^2\, Var(N_i) \\ &= \frac{\delta t^4 n(n+1)(2n+1)}{6} Var(N) \\ &\approx \frac{\delta t \cdot t^3 \cdot \sigma^2}{3} \end{aligned} \tag{2.4}$$

under the assumption that δt is small, which is usually valid for modern MEMS accelerometers. This analysis shows that accelerometer white noise creates a second order random walk in position, with zero mean and a standard deviation

$$\sigma_s(t) \approx \sigma \cdot t^{3/2} \cdot \sqrt{\frac{\delta t}{3}} \tag{2.5}$$

2.2.1.3 Flicker Noise / Bias Stability

Flicker noise in MEMS accelerometers causes the bias to wander over time. This noise is caused by electronics and in other components susceptible to random flickering. Flicker noise has *1/f* spectrum, the effects of which are usually observed at low frequencies in electronic components. At high frequencies this noise is overshadowed by white noise. Such fluctuations are usually modeled as bias random walk. Flicker noise creates a second order random walk in velocity where uncertainty grows proportionally to $t^{3/2}$, and a third order random walk in position which grows proportionally to $t^{5/2}$ [53]. In reality, bias fluctuations do not really behave as a

random walk. If they did, then the uncertainty in the bias of a device would grow without bound as the time span increase. In practice, the bias is constrained to be within some range. Therefore, the random walk model is only a good approximation to the true process for short periods [53].

2.2.1.4 Temperature Effects

Temperature changes cause fluctuations in the bias of output signal. The relationship between bias and temperature depends on the specific device; however, it is often highly nonlinear [53]. If the sensor board contains a temperature sensor then it is possible to apply corrections to the output signals in order to compensate for temperature dependent effects.

2.2.1.5 Calibration Errors

Calibration errors (errors in scale factors, alignments and output linearities) appear as bias errors that are only visible whilst the device is undergoing acceleration.

2.2.1.6 Summary

Table 2.1 summarizes the error sources presented in this section. The relative importance of each error source depends on the specific device being used. In case of accelerometers, the errors have significant impact on the accuracy and reliability of the performance as these errors get accumulated in the double integration process.

Table 2.1 Summary of Accelerometer Error Sources [53]

Error Type	Description	Result of Double Integration
Bias	A constant bias ϵ in the accelerometer's output signal	A quadratically growing position error $$s(t) = \epsilon \cdot \frac{t^2}{2}$$
White Noise	White noise with some standard deviation σ	A second-order random walk. The standard deviation of the position error grows as $$\sigma_s(t) = \sigma \cdot t^{3/2} \cdot \sqrt{\frac{\delta t}{3}}$$
Temperature Effects	Temperature dependent residual bias	Any residual bias causes an error in position which grows quadratically with time
Calibration	Deterministic errors in scale factors, alignments and accelerometer linearities	Position drift proportional to the squared rate and duration of acceleration
Bias Instability	Bias fluctuations, usually modelled as a bias random walk	A third-order random walk in position

2.2.2 Gyroscopes

Gyroscopes are also available in almost all categories as that of accelerometers. Two of those common types with impressive accuracy are mechanical and optical gyroscopes. A mechanical gyroscope consists of a spinning wheel mounted on two gimbals, which allow it to rotate in all three axes Fig. 2.3. The fallout from the conservation of angular momentum is that the spinning wheel resists changes in orientation. Hence, when a mechanical gyroscope is subjected to a rotation the wheel will remain at a constant global orientation and the angles between adjacent gimbals will change. To measure the orientation of the device, the angles between adjacent gimbals can be read using angle pick-offs. Note that a conventional gyroscope measures orientation. In contrast, nearly all modern gyroscopes (including the optical and MEMS types) are rate-gyros, which measure angular velocity.

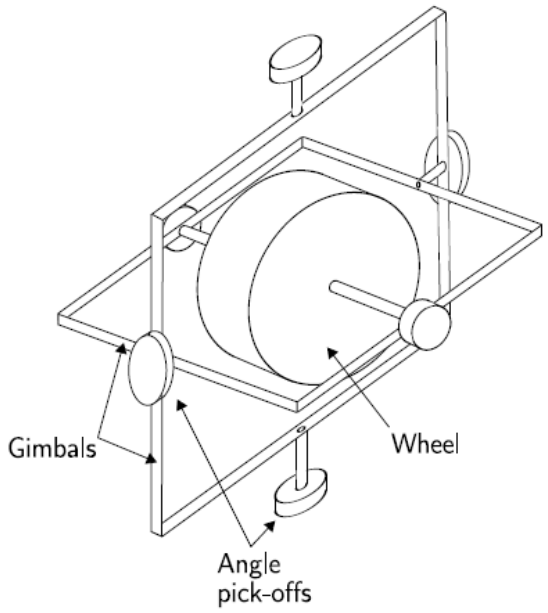

Fig. 2.3 A conventional mechanical Gyroscope [53]

A fiber optic gyroscope (FOG) uses the interference of light to measure angular velocity. A FOG consists of a large coil of optical fiber. To measure rotation two light beams are fired into the coil in opposite directions. If the sensor is undergoing a rotation then the beam travelling in the direction of rotation will experience a longer path to the other end of the fiber than the beam

travelling against the rotation, as illustrated in Figure 2.4. This is known as the Sagnac effect. When the beams exit the fiber they are combined. The phase shift introduced due to the Sagnac effect causes the beams to interfere, resulting in a combined beam whose intensity depends on the angular velocity. Therefore, it is possible to measure the angular velocity by measuring the intensity of the combined beam.

Unlike mechanical gyroscopes, optical gyros contain no moving parts and require only a few seconds to start-up. The accuracy of an optical gyro is largely dependent on the length of the light transmission path (larger is better), which is constrained by the size of the device.

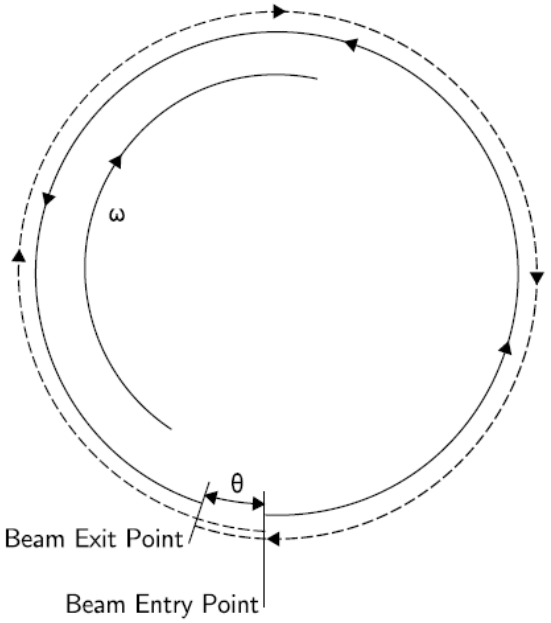

Fig. 2.4 The Sagnac effect. The dashed line is the path taken by the beam travelling in the direction of rotation. The solid line is the beam travelling against the rotation. θ is the angle through which the gyro turns whilst the beams are in flight [53].

Due to large part counts, parts with high-precision tolerances and intricate assembly techniques make mechanical and optical gyroscopes expensive. On the contrary, MEMS gyroscopes have low part counts (as low as three parts) and are cheap to manufacture. MEMS gyroscopes make use of the Coriolis effect [47], which states that in a frame of reference rotating at angular velocity ω, a mass m moving with velocity v experiences a force

$$F_c = -2m(\omega \times v) \tag{2.6}$$

MEMS gyroscopes use vibrating elements of different geometries, e.g., vibrating wheel and tuning fork, to measure the Coriolis effect. The simplest geometry consists of a single mass that is driven to vibrate along a drive axis, as shown in Fig. 2.5. When the gyroscope rotates, a secondary vibration is induced along the perpendicular sense axis due to the Coriolis force. The angular velocity is calculated by measuring this secondary rotation.

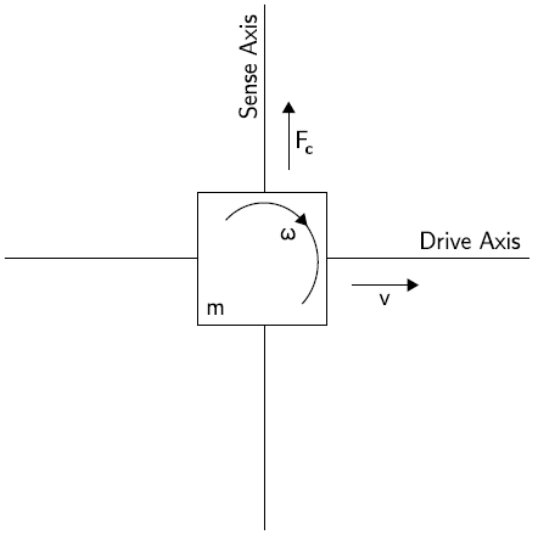

Fig. 2.5 A vibrating mass gyroscope [53]

Similar to MEMS accelerometers, performance of MEMS gyroscopes is also affected by error sources. The only difference being that in case of gyroscope, the parameters of motion of interest are the angles, which are obtained by the single integration of gyroscope output signals. It might seem like the effect of gyroscope error is not as severe as that of accelerometer's output error as they are integrated only once while as those from accelerometer are integrated twice. However, when the gyroscopes are used in navigation systems or motion tracking applications along with the accelerometers, then the gyroscope errors are the most dominant ones because gyroscope is used to calculate the orientation matrix, which is then used to transform the accelerometer outputs. Thus, it is imperative to learn about the error sources in gyroscope and their effect on its accuracy/reliability.

2.2.2.1 Constant Bias

The bias of a rate gyro is the average output from the gyroscope when it is not undergoing any rotation (i.e., the offset of the output from the true value), in °/h. A constant bias error of ε, when integrated, causes an angular error which grows linearly with time

$$\theta(t) = \varepsilon \cdot t \tag{2.7}$$

The constant bias error of a rate gyro can be estimated by taking a long-term average of the gyroscope's output while it is not undergoing any rotation. Once the bias is known, it can be compensated by subtraction [53].

2.2.2.2 Thermo-Mechanical White Noise / Angle Random Walk

The gyroscope output is perturbed by a white noise sequence, which on integration produces a random walk error in the angle output. We can analyze the effect of this white noise sequence on the error in calculated angle by following the analysis given in accelerometer's case. Let N_i be the i-th random variable in the white noise sequence, with $E(N_i) = E(N) = 0$ and $Var(N_i) = Var(N) = \sigma^2$. The result of integrating the white noise signal $\varepsilon(t)$ over a time span $t = n \cdot \delta t$ is

$$\int_0^t \varepsilon(\tau) d\tau = \delta t \sum_{i=1}^{n} N_i \tag{2.8}$$

The error in angle generated as a result of white noise in gyroscope's output has mean and variance as given below

$$E\left(\int_0^t \varepsilon(\tau) d\tau\right) = \delta t \cdot n \cdot E(N) = 0$$

$$Var\left(\int_0^t \varepsilon(\tau) d\tau\right) = \delta t^2 \cdot n \cdot Var(N) = \delta t \cdot t \cdot \sigma^2 \tag{2.9}$$

Hence, the noise introduces a zero-mean random walk error into the integrated signal, whose standard deviation grows proportionally to the square root of time:

$$\sigma_\theta(t) = \sigma \cdot \sqrt{\delta t \cdot t} \tag{2.10}$$

It is common for manufacturers to specify noise using an angle random walk (ARW) measurement (units $°/\sqrt{h}$). Other measurements used to specify noise are power spectral density

(units $(°/h)^2$/Hz) and FFT noise density (units °/h/$\sqrt{Hz}$). It is possible to convert between the various different noise specifications using the equations

$$ARW(°/\sqrt{h}) = \frac{\sqrt{PSD((°/h)^2/Hz)}}{60} \tag{2.11}$$

$$ARW(°/\sqrt{h}) = \frac{FFT(°/h/\sqrt{Hz})}{60} \tag{2.12}$$

For more information about angle random walk and noise specifications see [37].

2.2.2.3 Flicker Noise / Bias Stability

Flicker noise also affects the gyroscope output making the bias of gyroscope output wander over time. Bias fluctuations that arise due to flicker noise are usually modeled as a random walk. A bias stability measurement describes how the bias of a device may change over a specified period, typically around 100 seconds, in fixed conditions (usually including constant temperature). Bias stability is usually specified as a 1σ value with units °/h, or °/s for less accurate devices. Over time, this property creates a random walk in the gyro bias, whose standard deviation grows proportionally to the square root of time. For this reason bias stability is occasionally specified by a bias random walk measurement

$$BRW(°/\sqrt{h}) = \frac{BS(°/h)}{\sqrt{t(h)}} \tag{2.13}$$

where *t* is the period over which bias stability is defined. If we assume the bias random walk model, then the result of integrating the bias fluctuations is a second-order random walk in angle [53].

2.2.2.4 Temperature Effects

Temperature fluctuations due to changes in the environment and sensor self-heating induce movement in the bias. Note that such movements are not included in bias stability measurements, which are carried out under fixed temperature conditions. The relationship between bias and temperature is often highly nonlinear for MEMS sensors.

2.2.2.5 Calibration Errors

Errors in the scale factors, alignments, and linearities of the gyros are collectively termed as calibration errors. They come into play only when the device is under (rotational) motion. Such errors lead to the accumulation of additional drift in the integrated signal, the magnitude of which is proportional to the rate and duration of the motions [11].

2.2.2.6 Summary

For MEMS gyroscopes, angle random walk (noise) errors and uncorrected bias errors either due to uncompensated temperature fluctuations or an error in the initial bias estimation, are usually the most important sources of error. Angle random walk can be used as a lower bound for uncertainty in the orientation obtained from integrating a rate-gyroscope's signal. Table 2.2 summarizes the different error sources present in MEMS gyroscope's output and their effect on its performance.

Table 2.2 Summary of Gyro Error Sources [53]

Error Type	Description	Result of Integration
Bias	A constant bias ϵ	A steadily growing angular error $\theta(t) = \epsilon \cdot t$
White Noise	White noise with some standard deviation σ	An angle random walk, whose standard deviation $\sigma_\theta(t) = \sigma \cdot \sqrt{\delta t \cdot t}$ grows with the square root of time
Temperature Effects	Temperature dependent residual bias	Any residual bias is integrated into the orientation, causing an orientation error which grows linearly with time
Calibration	Deterministic errors in scale factors, alignments and gyro linearities	Orientation drift proportional to the rate and duration of motion
Bias Instability	Bias fluctuations, usually modelled as a bias random walk	A second-order random walk

Chapter 3

Problem Statement

In recent years, research interest in the area of telemonitoring for health and wellness purposes has increased with the advances in sensor technologies. Considerable amount of work has been done concerning remote health monitoring for elderly and disabled persons. IMUs have been widely used for monitoring the mobility and posture with some work done on motion tracking using IMUs. This chapter reviews the earlier work that influenced the work presented in this thesis.

We can categorize the systems developed so far in two broad categories –

- Systems that use inertial sensors as secondary sensors to improve/enhance the functionality of primary sensor
- Systems that use inertial sensors as primary sensors.

References [36] and [17] belong to the first category where in RFID sensors are used as primary sensors to identify/classify ADLs. ADLs involving interaction with the physical and social environment such as using telephone, doing laundry, preparing food, and housekeeping are known as Instrumental ADLs (IADL). In identifying ADLs using RFID sensors, various objects are tagged with RFID sensors and a RFID reader glove is worn to carry out ADLs. In cases where RFID tag reading cannot identify the ADL accurately, accelerometer data is used to identify ADLs. This is achieved by calculating various features of the accelerometer data such as mean, variance, energy, spectral entropy, FFT coefficients etc. and using either Naïve Bayes or Hidden Markov Models (HMMs) or Joint Boosting algorithms for activity classification.

References [28], [25], [44], and [32] belong to the second category where in IMUs serve as the primary sensor. [28] uses a 2D accelerometer and a gyroscope orthogonal to the

accelerometers, attached to the chest, to monitor physical activities and postural transitions in elderly. The wavelet transform, in conjunction with a kinematics model, detects transitions among different postures like standing, sitting and lying as well as walking periods during daily life with an accuracy of about 99%.

Luinge et al. [25] use a chest mounted 3D accelerometer to determine the trunk and pelvis inclination during the functional 3D activity of stacking crates. They start out with a detailed model for the sensor signal based on assumptions concerning the frequency content of the acceleration of the movement that is measured, the knowledge that the magnitude of the gravity is $1g$ and taking into account a fluctuating sensor offset. A complementary extended Kalman filter is used to estimate the various components of the sensor output, which are then used to calculate inclination. The root-mean square error in inclination estimate is found out to be 2° using an optical position tracking system.

Uswatte et. al. [44] used a 2D accelerometer mounted on each wrist to identify the period and duration of movement in both arms. This was achieved using a 'threshold filter' (low pass filter). The outcome of the study was the ratio of impaired-to-unimpaired arm threshold filtered data and its correlation with 2 real-world measures of arm use - the MAL (motor activity log) and AAUT (Actual Amount of Use Test). The ratio summary output was verified against another real-world measure of overall physical activity, the stroke impact scale (SIS).

Patel et al. [32] is another example of inertial sensors being used in CIMT for stroke rehabilitation. Here, the authors use multiple accelerometers on fingers, palm of the hand, forearm, upper arm of the affected side and one more on sternum to analyze a movement and its components such as reaching, manipulation, release/return etc. Data features are extracted and selected so as to maximize the separation among classes associated with different clinical scores. The movements were classified based on these features using Random Forest to estimate clinical score of each movement. This process was carried out for a subset of motor tasks used in measuring Functional Ability Scale (FAS) and all the estimated clinical scores were combined

through a linear equation to calculate FAS. This approach validates usage of accelerometer data for calculating FAS score reliably.

With advances in manufacturing technology of MEMS devices, their accuracy has started improving along with reduction in size and power consumption. This has given rise to a new era of wearable computing and other exciting opportunities in virtual reality. Various medical applications have benefitted from these emerging technologies. People have started using inertial sensors for motion tacking purposes. Guillemaud et. al. [12] describes a motion capture device for the purposes of activity classification or monitoring. The IMU consisted of a 3D accelerometer and 3D magnetometer. Data from these sensors was fused to estimate the body orientation. This method works fine for tracking slow movements but tracking fast movement becomes difficult without gyroscope signal giving rise to very large orientation errors. Sabatini et. al. [34] presents a model for quaternion-based strap-down integration for application to gait analysis. It uses an IMU consisting a 3D accelerometer and 3D gyroscope. The model is validated with simulations based on synthetic trajectories that had been derived from the author's earlier work on foot inertial sensing.

Torres et al. present a mathematical algorithm for 3D tracking using IMU in [43]. Their IMU consists of 3D accelerometers, 3D gyroscopes and 3D magnetometers. They use Kalman filter based sensor fusion for tracking purposes. Without proper model for the sensor signal, the algorithm fails to track position not only in 3D but also in 2D due to the drift present in MEMS sensor outputs. Bachmann presents a real time human limb tracking virtual reality application using 3D accelerometer, 3D gyroscope and 3D magnetometer in [4]. Accelerometer and magnetometer are used to track low frequency components of the motion while as gyroscope is used to track high frequency components of the motion. A quaternion-based Kalman filter allows continuous correction for drift and tracking of the movement through all orientations.

3.1 Problem Statement

As explained in Section 2.1, current form of CIMT uses a mitt and a touch sensor on wrist along with motor activity log maintained by patient to monitor affected arm usage indirectly. This method is not efficient and prone to errors. This inefficiency translates into longer time for recovering from disabilities and more visits to therapist. This increases the cost of stroke related health care. Thus, there is a need to develop cost-effective and reliable technique, which can monitor affected arm use accurately. This will reduce the number of visits to therapist and accurate detection of arm use will make CIMT more effective resulting in reduction of total time to recover functional abilities in stroke-affected arm.

Although, accurately sensing the affected arm movement will be a significant improvement over the technique being followed currently, the affected arm movement by itself does not give much information about rehabilitation progress. The objective tests such as BBT that are being used to assess the rehabilitation progress require manual intervention, which again introduces inefficiency. Advances in MEMS sensor technology have made wearable sensor systems a reality. This thesis will investigate a wearable sensor based technique, which can objectively assess the rehabilitation progress with minimal user intervention.

Thus, the combination of solutions to both issues described thus far will enable therapists to track rehabilitation progress remotely.

It is clear, from the literature review presented earlier in this chapter, that inertial sensors have the potential to track human motions but their performance is limited by the error sources. These error sources in MEMS sensors arise due to limitations of manufacturing process. With advances in manufacturing technology of MEMS sensors, performance of MEMS sensors has been improving. A human arm motion tracking system based on MEMS sensors will enable therapists to observe the affected arm motion as it was performed in real life. This ability will negate any need of an objective metric to track rehabilitation progress. This thesis will investigate development of such motion tracking system based entirely on MEMS sensors. Taking note of

the issues plaguing the applicability of inertial sensors in reliable and accurate human motion tracking, as discussed earlier, this thesis will start exploring the solution space with minimum complexities such as 1D motion with no orientation change. Then, the thesis will build upon 1D solution to determine its usability for 2D motions. Various error sources limiting the performance of motion tracking algorithm and methods that can correct those error sources will also need to be investigated.

With all of the above solutions, we will be a step closer to a rehabilitation therapy with minimum manual intervention and more accurate objective performance metrics. Designing a wearable sensor system will enable us to analyze the arm motions on-board. An efficient data processing algorithm can help make this data analysis in real time providing invaluable real time feedback to the patient.

Chapter 4

Quantifying Impaired Arm Usage

As described in Chapter 2, the current state of stroke rehabilitation therapy (CIMT) administration has some critical inefficiencies/inaccuracies associated. Improvements in MEMS inertial sensors (MIS) have enabled us to sense human motions more reliably. MIS based technology prevents subjective and error prone aspects of CIMT. Therefore, the improvement deliverable by MIS based technology is invaluable.

The primary source of inaccuracy within CIMT lies in determining the duration the patient underwent CIMT. Therapists have used this duration as a direct indication of effectiveness of CIMT, especially when patient is not under direct observation. When patient is away, therapists cannot take manual observations. Therefore, they rely on duration as an indicator of effectiveness though there is no definite method available to measure that duration. Currently, most therapists use a constraining mitt (glove) on unaffected arm with touch/proximity sensor embedded in it. Whenever patient wears the mitt, touch/proximity sensor will indicate so with change in its output. As unaffected arm is constrained, therapists assume that patient is using affected arm and hence a CIMT session is underway. If someone wears the constraining glove but does not move the affected arm then therapist would not know it from touch/proximity sensor reading and this introduces inefficiency in the CIMT. In addition, if someone wears the constraining glove but uses the affected arm just 20% of the time the glove was on, the sensor output will not be able to quantify the affected arm usage, introducing inaccuracy in the CIMT.

This chapter describes a new approach towards identifying CIMT periods using MEMS inertial sensors. The first section describes the sensor used for this purpose. Second section presents an algorithm to find out the durations when the affected arm is in use. We extend this

algorithm further to calculate the activity count in the case when repetitions of a single activity are performed. We summarize the contributions of this chapter in the last section.

4.1 WiTilt v2.5

This chapter uses a 3D accelerometer sensor board (now discontinued) from Sparkfun electronics. The product name for this board is WiTilt v2.5. It has a Freescale MMA7260Q triple axis accelerometer and a class 1 Bluetooth link for robust communications. Some other features of the WiTilt v2.5 are - adjustable sensor range from 1.5*g* to 6*g*, output in raw ADC or calculated gravity format, adjustable output frequency from a minimum of 10Hz to a maximum of 610Hz in selected modes of operation. Figure 4.1 below shows the sensor board against a U.S. quarter dollar coin.

Fig. 4.1 Sensor Board - WiTilt v2.5

Using Lithium-ion battery and some support materials the sensor board was assembled into a wearable device as shown in Fig. 4.2.

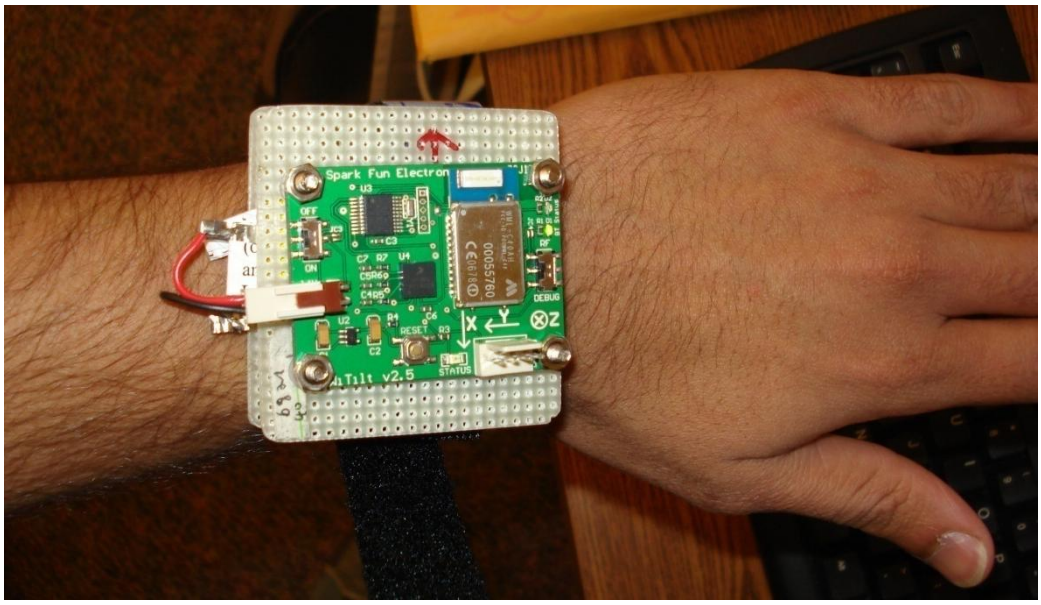

Fig. 4.2 The Wearable Device used in this work

The three principal axes of the sensor can be clearly seen marked in bottom right corner of the sensor board. The Y-axis is oriented along the length of forearm while as X-axis is perpendicular to the forearm length. Z-axis is oriented such that it goes into the plane of the paper (image).

The output of any axis of an accelerometer is –

$$z = g \cdot cos\ \theta + a + o + w \tag{4.1}$$

where, z – Accelerometer axis output,

g – Acceleration due to gravity,

θ – Angle made by the particular axis of accelerometer with vertical axis of g

a – Linear acceleration in the direction of accelerometer's axis

o – Offset present in the accelerometer axis

w – Measurement noise

a represents the component of linear acceleration present along the concerned axis of accelerometer when a motion is being performed. The offset in accelerometer output, o is a very low frequency signal, and we can assume it constant for in-lab sessions of CIMT, which usually will not be longer than an hour. The measurement noise term w represents the white noise present

in the sensor output due to thermo-mechanical vibrations as well as quantization noise of ADC on-board. Though accelerometer output responds to motions in each axis, because of measurement noise *w*, the output will not be stationary when there is no motion.

4.2 Impaired Arm Usage

To improve upon the touch/proximity sensor based method for finding impaired arm use during a CIMT session, we need to find the durations when the impaired arm is -"actually" being used rather than the total time of CIMT session.

4.2.1 Converting data to '*g*'-scale

WiTilt has a 10-bit ADC on board giving a range of 0 to 1023 counts on all '*g*' ranges. In this work, the sensor was set up with a ±2*g* range. Thus, to convert raw ADC output into '*g*'-scale, data requires centering on 512 and scaling according to the '*g*' range selected. For ±2*g* range, the centered data is scaled (divided) by 256 to get the sensor output in '*g*'-scale.

4.2.2 Threshold

As described earlier, the accelerometer output changes in response to the motions carried out in the direction of any of its sensitive axes. Nevertheless, when there is no motion, accelerometer output might still show some variations due to the measurement noise *w*. Thus, to identify durations when the impaired arm is not in use, we should find the durations of no signal variations after we remove the noise *w*. We can characterize noise *w* by various ways. One approach is to find the statistical parameters of the noise component by keeping the sensor stationary for about 30 minutes and sampling the data at minimum of 10Hz. We considered two parameters to characterize the noise *w* – standard deviation and peak amplitude. Both of these values are used to threshold the accelerometer signal variations among consecutive samples. If the signal change between two samples is less than the parameter under consideration, then that signal change is considered to be due to measurement noise *w* and not due to the actual movement of impaired arm. We attribute remaining all signal variations to the impaired arm

movement. Thus, we can identify the periods of activity and inactivity from observing the accelerometer output signal of a particular axis.

The impaired arm movements of stroke survivors are not very smooth as they lack fine motor skills. Unintended jerks are characteristic of stroke affected arm movements. Thus, choosing peak amplitude as threshold parameter over the standard deviation helps improve the accuracy of this method. However, this reliance on presence of unintended jerks will make this method unusable for someone carrying out smooth movements (or not exhibiting unintended jerks). In the case of smooth movements, the signal change between consecutive samples might be well below the threshold and over time the signal will change sufficiently but this algorithm will not be able to identify this change.

The data collected for an activity of throwing a ball illustrates the algorithm. Figure 4.3 shows the '*g*'-scale converted raw data along with mean of the data for individual axis. As can be seen from the Fig. 4.3, the activity is essentially carried out in XY plane with a minimal movement along Z direction.

4.2.3 Calculating the arm usage

As the arm moves in 3D space, at least one of the accelerometer axes will experience the motion. Whenever a sample represents impaired arm motion, we increment a counter. To calculate the total impaired arm usage, we normalize the activity period over the complete observation period.

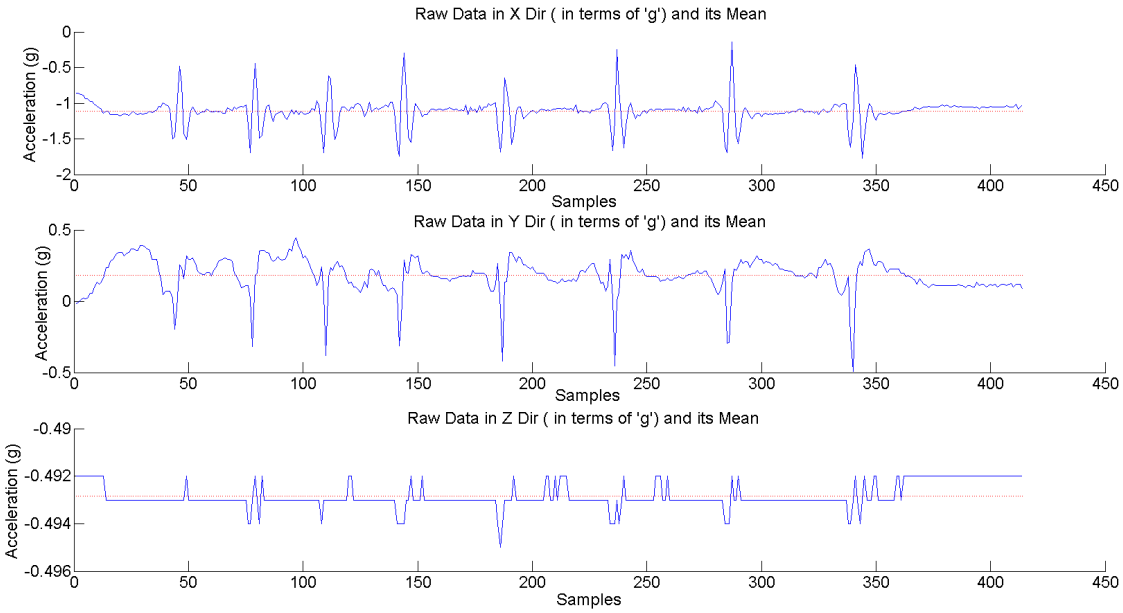

Fig. 4.3 Raw Data in '*g*'-scale with it mean along 3 axes

We also have counters for each of the three axes. This information provides an insight into the impaired arm usage along the three axes of accelerometer whose alignment with respect to the affected arm does not change.

Figure 4.4 shows the data after it's subjected to a threshold of 0.022*g*. The samples that are above the threshold from previous sample are marked as '1' indicating detection of activity while as all other samples are marked '0' indicating no activity.

The work presented in this chapter is coded into a command line operated - menu driven software, that runs on any Microsoft Windows® machine as a standalone program. The software has the facility of storing the results of each processed data file into a text file. Table 4.1 shows the results calculated by this algorithm for the above data file.

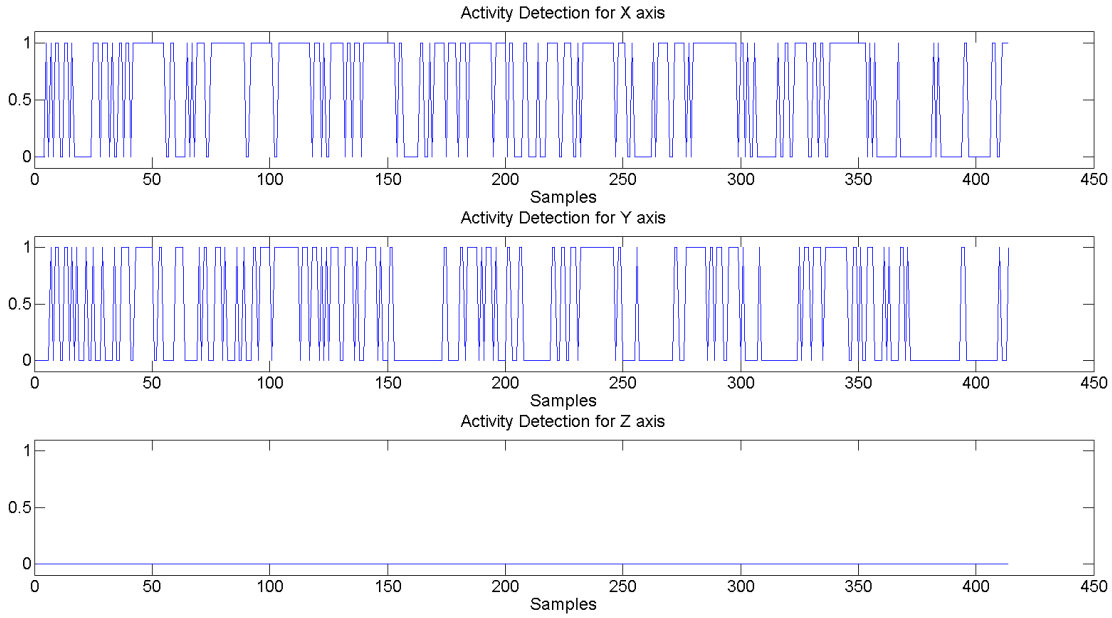

Fig. 4.4 Activity Markers along 3 axes after threshold filtering raw data

Table 4.1 Result of Impaired Arm Usage Algorithm

Usage_X	Usage_Y	Usage_Z	Usage
58.741%	42.191%	0.2331%	68.765%

J. C. Lötters et. al. present another approach of determining activity periods using 3D accelerometers in [23]. They make use of the fact that when under no motion the vector sum of accelerations experienced by three axes of accelerometer equates to 1*g*. Thus, to determine activity durations they apply threshold to the rectified vector sum of accelerometer output. Although, this method serves the purpose, our algorithm gives out more information than just activity periods. As we detect activities individually in each axis and calculate the total arm usage, a physician gets to know the rehabilitation progress in each axis as well as the overall therapy progress from these values.

4.2.4 Counting Activities

We improved the way CIMT is delivered and the rehabilitation progress is monitored by improving the accuracy of detecting impaired arm use. We will go even further to get more meaningful information using 3D accelerometer. In CIMT, a stroke survivor is encouraged to use his affected arm more than the unaffected one. In addition, for people with severe stroke and for

those who are in early stages of rehabilitation, the easier movements are not ADLs but rather some basic arm movements usually done repetitively. For this scenario, we present an algorithm to find the number of repetitions done using 3D accelerometer signal.

For this purpose, we consider the signal associated with the impaired arm movement with some basic repetitious movement. This is necessary to avoid any drift in the average value of the signal due to orientation change. Data is converted into a stream of '1's and '-1's by comparing them with respect to the average value. Values higher than the average value will get mapped to '1' and values lower than the average value will get mapped to '-1'. Whenever the signal changes its mapped value, the activity counts increment. Figure 4.5 shows the results obtained using this method.

As seen in the Fig. 4.5, this method does not perform well and overestimates the activity count. The primary reason behind this degraded performance is measurement noise. The algorithm compares the signal value directly with its average without considering the effect of *w*. In addition, the algorithm assumes that starting and final position of the impaired arm remains same to ensure the data average is midway between the two extremes that any activity might experience. The algorithm's inability to distinguish among signal changes due to activity and those due to noise, limits its performance. One way to improve the algorithm's performance will be using some threshold. The threshold should be large enough to reject the noise but should be small enough to detect the activities. Thus, a modified algorithm is presented to calculate activity count.

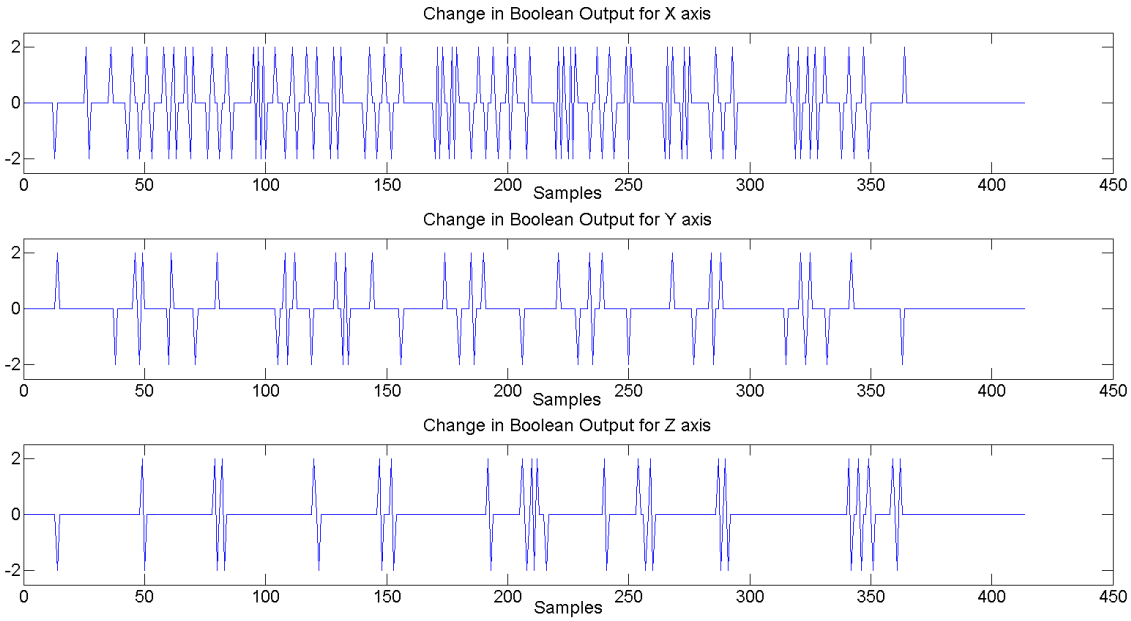

Fig. 4.5 Activity count measurement output along 3 axes

In the second algorithm, weighted moving average is calculated over a window of *m* samples on either side of the sample. The value of parameter *m* depends on the activities being carried out. Too large value of *m* will merge multiple activities while as too small value of '*m*' will overestimate the activity count. This moving average is subtracted from the raw data to get high frequency content of the data centered about zero. The high frequency contents will also have high frequency part of the measurement noise. We remove the measurement noise by threshold filtering high frequency contents using the peak amplitude of noise. All the samples with values less than the peak amplitude of measurement noise are forced to zero.

To determine the activity count from the noise removed high frequency content of data, we search for 'dead (no-motion) periods'. A dead period is defined as the period where consecutive samples with value 0 exceed or equal to a user parameter *dead*. These periods represent durations when there was no movement. Remaining intervals are noted as activity intervals and thus we not only find the activity count but also the duration of each activity. Results obtained after processing the data file considered in Fig. 4.5, are shown in Fig.4.6 through Fig. 4.8. Figure 4.6 shows raw data along with output of moving average filter with *m*=5. Figure

4.7 shows result obtained after subtracting moving average filter output from the raw data. The output of noise removal step is shown in Fig. 4.8.

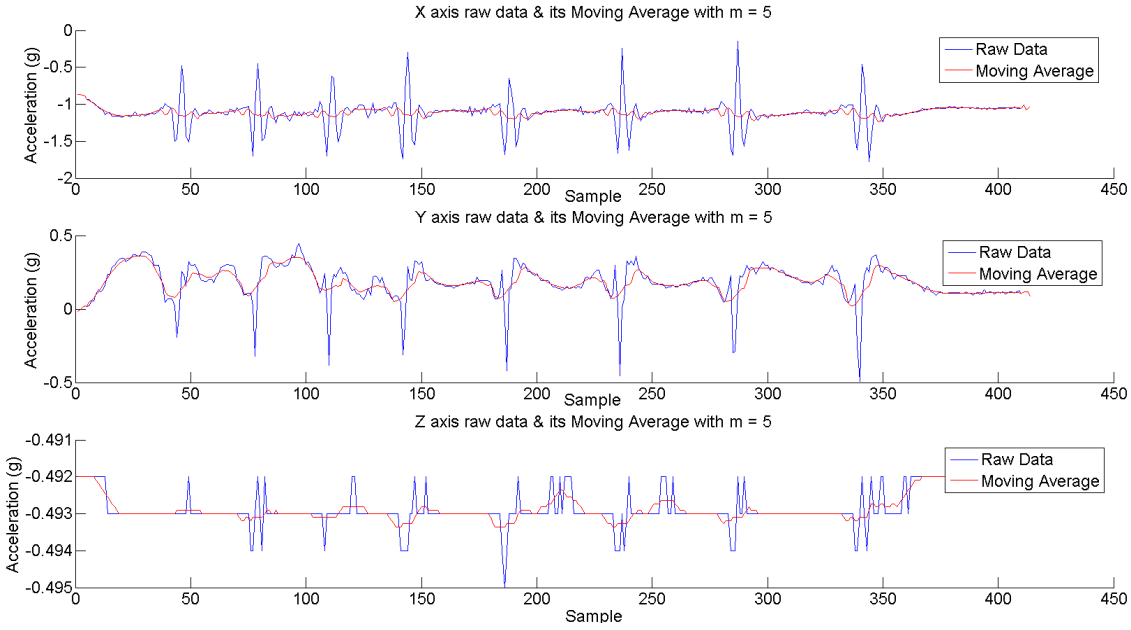

Fig. 4.6 Raw data and moving average filter output with *m*=5

As can be seen from Fig. 4.6 – Fig. 4.8, the second algorithm does a better job of removing noise and isolating activities. With *m*=5 and *dead*=10, this algorithm gives out number of activities carried out as 7 instead of original number of 8. This error is caused by the noisy signal observed in X axis between activity 2 and 3 (around sample number 100). Making *dead*=9 gives us equivalent number of activities as 8. For *dead*=8 the algorithm overestimates the activity number to 9.

The activity of throwing a ball usually has rapid changes in acceleration values as compared with a typical ADL. Thus, another activity of putting plastic chips in a checker box was carried out. Figure 4.9 shows the raw data and moving average filter output for this particular trial. As can be seen in Fig. 4.9, the acceleration values do not change that rapidly. However, even for this kind of data the algorithm presented in this section correctly identifies the number of activities performed.

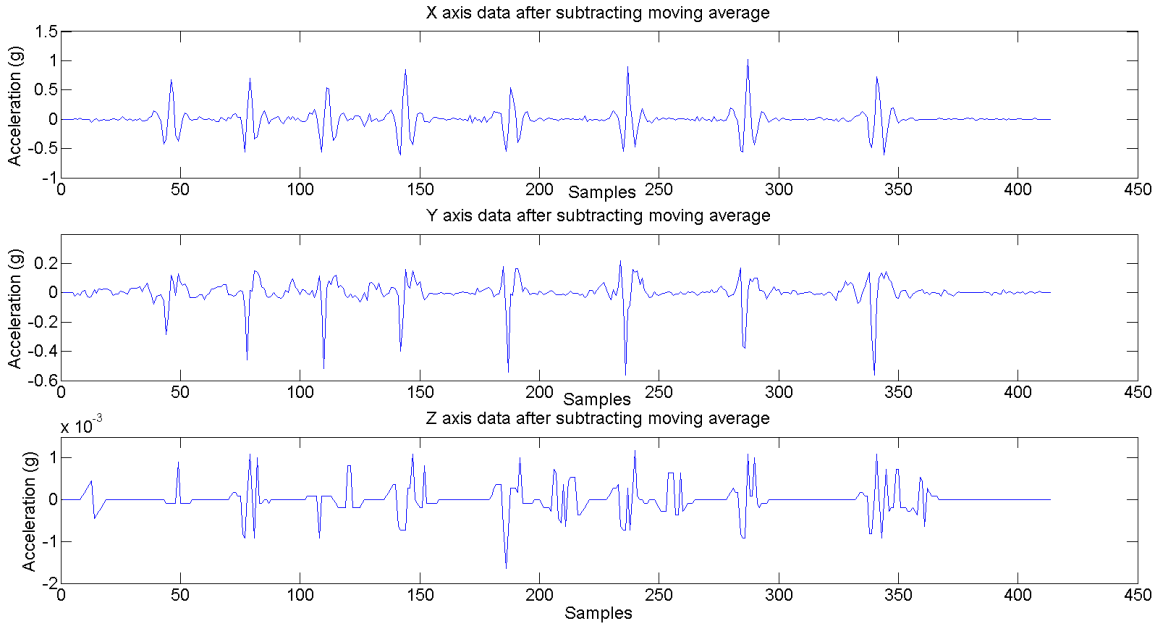

Fig. 4.7 Subtraction of moving average from raw data

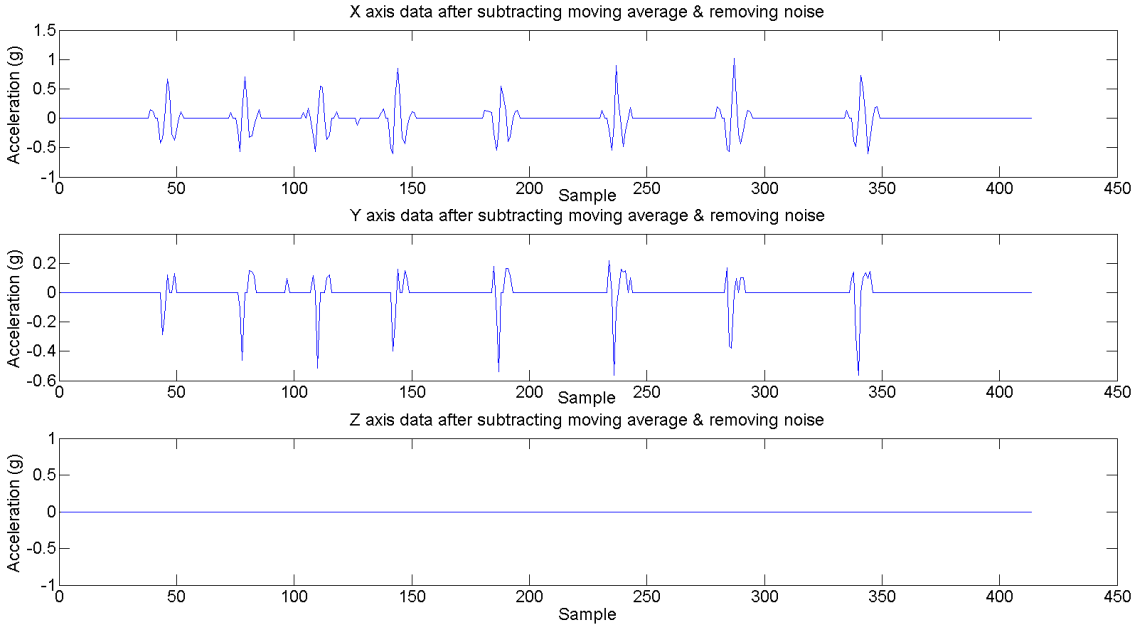

Fig. 4.8 Noise removal output

Thus, identifying individual activity still depends on the selection of '*dead*' parameter value. The dependence of *dead* on the activity performed by patients and the movement characteristics of the subject under consideration need to be investigated. Also, low pass filtering noise removed data of Fig. 4.8, might lead us to a more robust algorithm and much better results.

Please refer appendix A.1 for the source code of first activity detection method and appendix A.2 for the source code of second activity detection algorithm.

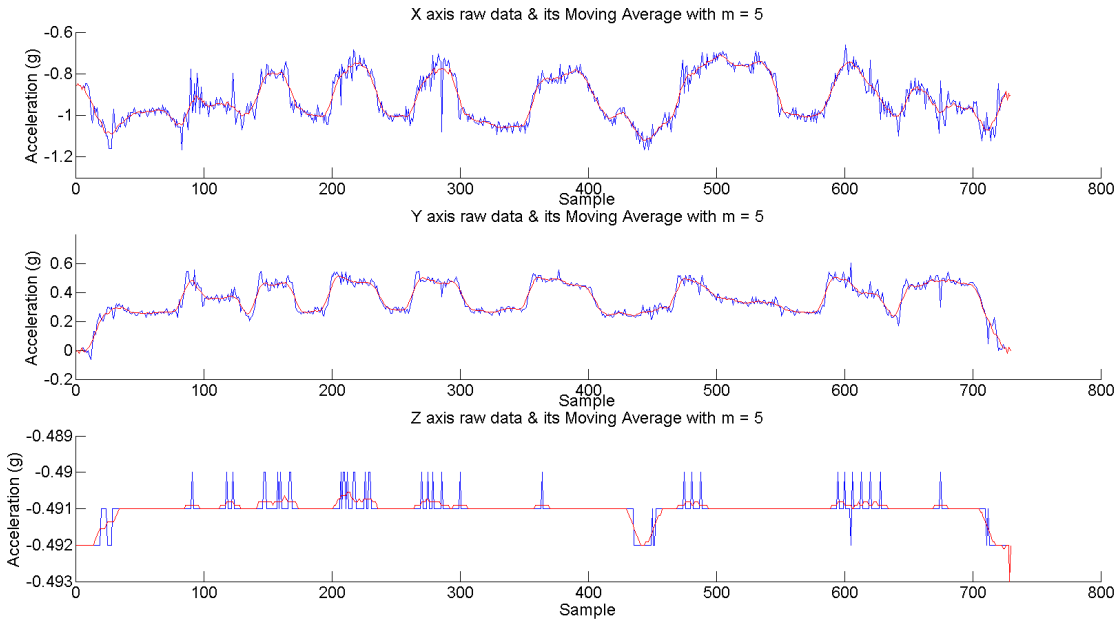

Fig. 4.9 Raw data and moving average filter output with *m*=5 for checker box ADL

4.3 Summary

This chapter presented an algorithm to determine impaired arm usage in a CIMT session. The algorithm makes CIMT more efficient and helps remove some inaccuracies associated with subjective observation of impaired arm usage. It gives more information regarding the impaired arm usage along the three axes of accelerometer, which might be used to improve CIMT in general. An algorithm for finding the activity count using 3D accelerometer is also presented. The next chapter will delve into a novel approach to track rehabilitation progress in CIMT by using 3D inertial sensors.

Chapter 5

Tracking Rehabilitation Progress through Histogram of Distance Traversed

In Chapter 4, we developed an algorithm to determine periods of affected arm usage. The algorithm improved the efficiency of identifying affected arm use as compared with current touch sensor based methodology and provided us with an objective method enhancing the way CIMT is delivered. Another inefficiency associated with the current form of CIMT is the lack of an automated process to determine rehabilitation progress objectively. This chapter looks at developing such an objective test with minimal user intervention with the intention of improving the accuracy, efficiency, and efficacy of CIMT. The primary goal in developing this new objective test is to develop it such that there will be minimum user intervention to carry out this test. This will enable us to use this test for remote monitoring of stroke rehabilitation.

This chapter first explains the functional tests used in current form of CIMT. Then the discussion focuses on the development of accelerometer based functional test. It is then followed by analysis of performance of two parameters namely – the window width and temporal resolution. Results from patient trials and their analysis follow next. The application of this algorithm to classify/differentiate patients based on their rehabilitation progress is discussed in next section. We conclude by summarizing the work done in this chapter.

5.1 Review of CIMT Progress Tracking

As described in Chapter 2, disability experienced by stroke survivors is a result of learned non-use. References [51], [50], [31], and [39] verify the usefulness of rehabilitation therapy methods such as CIMT. For tracking rehabilitation progress of the stroke survivors, multiple clinically proven tests are available such as MAL, FAS, Box and Block Test (BBT) etc. These tests offer a realistic view of the effectiveness of the therapy and hence are used by therapists to

administer the therapy more effectively. Thus, these tests form a critical part of the rehabilitation therapy and thus need to be accurate. Most of these tests are categorized as subjective or objective. Tests like MAL, which is a questionnaire/interview based test, fall under the subjective category because these tests rely on a person's opinion about the progress. On the other hand, tests like FAS and BBT fall under objective category as they try to measure the motor/functional ability of the affected arm and compare that functional ability against non-affected arm through well thought out experiments. Thus, objective tests represent rehabilitation progress more accurately than subjective tests.

The work presented here has been done in collaboration with the NeuroRehabilitation Research Laboratory (NRRL) at Colorado State University's Department of Occupational Therapy. NRRL uses BBT as one of the functional tests to evaluate CIMT and stroke rehabilitation progress in general. The next section discusses the BBT in detail.

5.1.1 Box and Block Test (BBT)

BBT is an outcome designed to measure functional changes in reach, grasp, and release when using the more-affected arm to transport blocks [14]. Image in Fig. 5.1 shows the typical setup to measure BBT performance of right hand. To measure the performance, subject is asked to move blocks from one section to other in prescribed time (1 minute). The same task is then repeated for the other hand. The number of blocks moved by each hand is the outcome of the test. The number of blocks moved by non-affected arm serves as a reference and those moved by the affected arm are compared against it. As rehabilitation progresses, it is expected that affected arm will tend to perform as good as the non-affected arm; implying functional abilities of affected arm are being restored.

Fig. 5.1 Box and Block Test [14]

This test requires minimal human intervention/interaction. The only interaction required is monitoring of elapsed time. Nevertheless, it requires the wooden blocks and the box setup to carry out the test. With this understanding of BBT, we will develop a MIS based functional test that will track the rehabilitation progress objectively.

5.2 Histogram of Distance Traversed Method

BBT is a time constrained performance test where in affected arm's performance is judged based on unaffected arm's performance in a given time. To develop an objective performance based test using accelerometer, we should calculate some numerical quantity from accelerometer output, representative of the performance of arm movements. We will use this numerical measure to determine the effectiveness of rehabilitation therapy.

As described in Section 4.1 earlier, the accelerometer output is the sum of gravity and linear acceleration due to arm motion with some additive noise (Equation 4.1). Thus, accelerometer output is linearly dependent on the arm motion. As the acceleration is second

derivative of displacement with respect to time, integrating acceleration twice with respect to time will give us displacement. However, during repetitive tasks, displacement, being a vector quantity, can never be larger than the range of human arm movements and thus may not be of any use for tracking rehabilitation progress using repetitive activities. Instead, we can consider only the magnitude of acceleration values and calculate a scalar quantity that will be representative of the movements carried out.

BBT is a time-constrained test and monitoring of elapsed time requires some manual intervention. To get rid of this intervention, the accelerometer-based test will be insensitive to time spent on activities. Instead, accelerometer test will be based on some functional objectives to be completed with no regards to time spent on achieving those objectives. To account for the variation in time spent on accomplishing those functional objectives, we will normalize the scalar quantity by total time spent. Thus, this normalized scalar value will be used to judge effectiveness of rehabilitation therapy.

In the case of repeating activities, we can never guarantee that the interval between any two repetitions is same. Thus, these varying 'no-activity' periods will not contribute to the scalar quantity but will affect the normalized answer. Thus, instead of considering all repetitions at once for analyzing the performance, we will split the data into fixed width windows (based on number of samples and effectively time).

5.2.1 Effect of Window Width

Window width should be chosen such that it has at least one complete repetition of the activity along with some non-activity period. We need this constraint as too small of a window width will give us windows with either no motion or part of the motion. In this case, the accumulated distances will not go beyond some maximum value as human arm motion has limits on its acceleration profile. If window width is too small, this maximum value of accumulated distance will be too low (and thus with very less resolution) making it hard to get meaningful information. Making window width too large is as good as analyzing all of the data at one go

which will make accumulated distance grow without any bound and thus making it hard to get meaningful information from the results.

The procedure explained above is discussed with a single axis in mind. Nevertheless, during ADL, it is hard to constrain arm movements to only a single axis. Assuming the generic case of arm movement in 3D, this procedure will be extended to take into considerations movements along 3 axes of accelerometer. This is achieved by applying above procedure to each axis individually. The final equivalent 3D performance metric will be calculated by taking square root of addition of squares of distances traversed in individual axes.

A trial is designed to observe the effects of varying window widths. The trial consisted of an activity repeated multiple times. The trial is conducted each day for a period of 8 days for both the unaffected and affected arm. Another trial is carried out with non-patient participation to see how the algorithm works on non-patient data set. Then, the data from these trials is processed using the algorithm explained above with window widths of 1*s*, 5*s*, and 10*s* each. Minimum window width of 1*s* is chosen in accordance to criterions explained earlier, namely minimum width to be greater than the time required to perform at least one repetition. Figures 5.2 to 5.4 show the histograms of distance covered in three cases under consideration.

Figure 5.2 shows normalized histogram of distance covered with a window width of 1*s*. The figure plots eight traces for affected arm data with each trace representing individual days. Figure 5.2 also plots an average normalized histogram of distance traversed by unaffected arm for 8 days. As can be seen, in Fig. 5.2, with window width of 1*s* the maximum distance covered is too small and very low distances are more frequent. In addition, over a period of 8 days there is no significant change in affected arm's performance compared to unaffected arm average performance. We are expecting the change in affected arm's performance as the clinical staff at NRRL has observed/documented that change using other functional tests over the same period. The results shown in Fig. 5.2 are so because with smaller window widths accumulated distance

depends largely on noise in accelerometer. Thus, for such small window widths the algorithm does not perform satisfactorily.

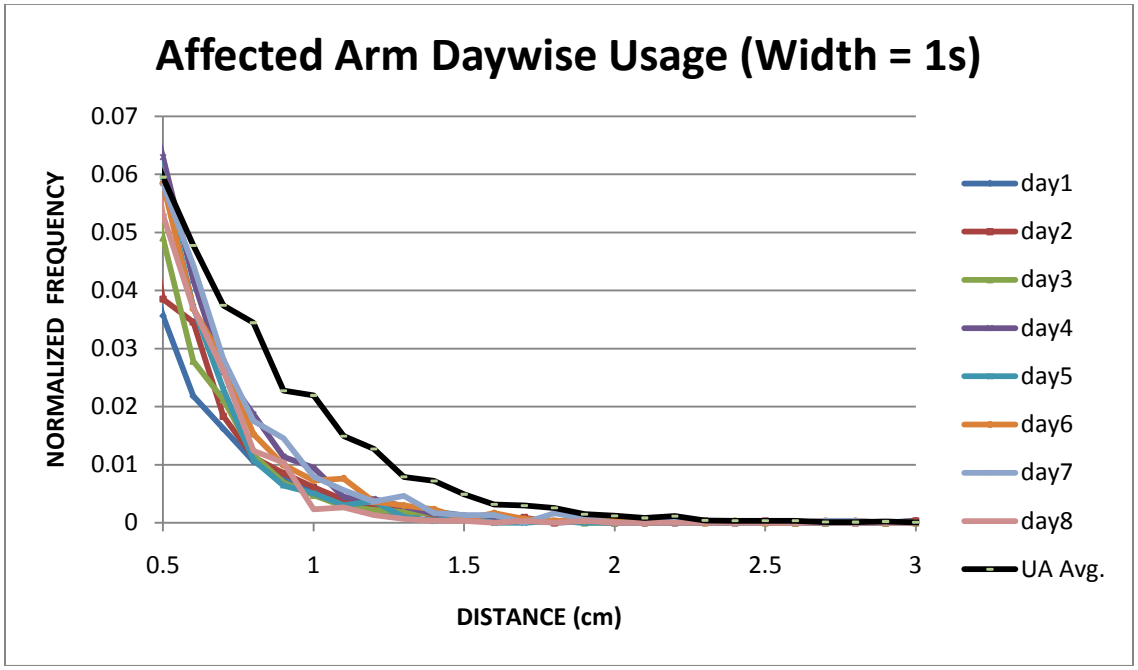

Fig. 5.2 Normalized Histogram of Distance Covered with width = 1*s*

Figure 5.3 shows the comparison of affected and unaffected arm performance over a period of 8 days with a window width of 5*s*. This figure illustrates that with a window width of 5*s* we can see incremental improvement in each day's performance. The unaffected arm average performance for 8 days is significant over the available range. It stands out from the affected arm traces. From the unaffected arm average performance, we infer that non-affected arm will cover large distances more often as compared with affected arm. Thus, we will focus on the frequency of large distances to track rehabilitation progress.

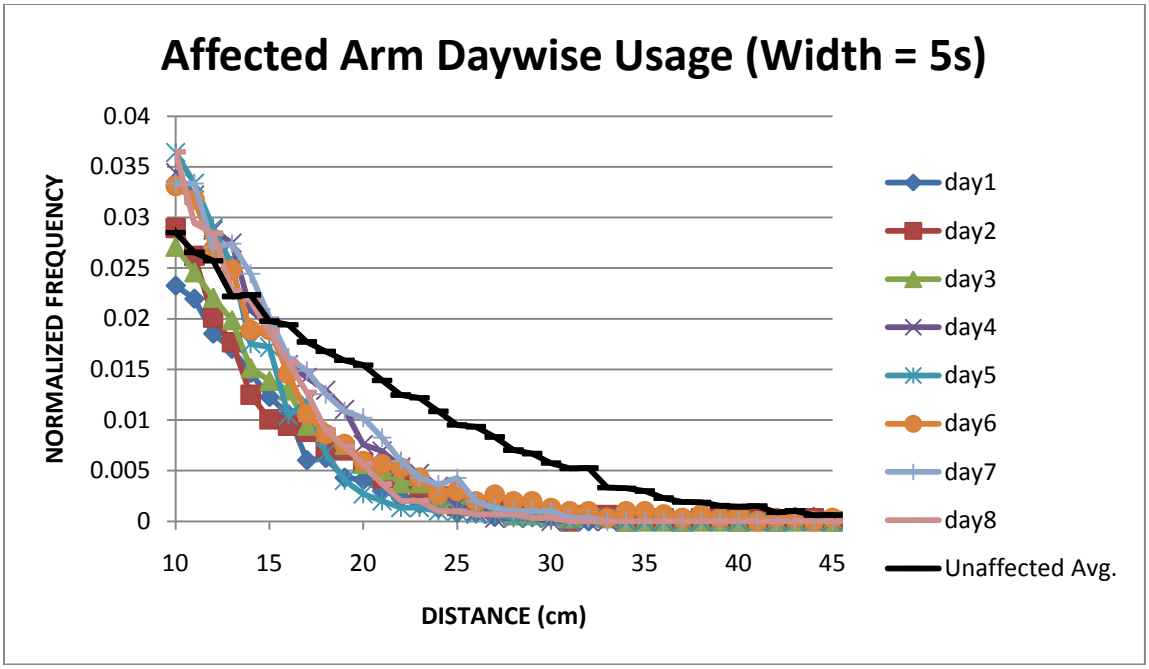

Fig. 5.3 Normalized Histogram of Distance Covered with width = 5*s*

Figure 5.4 shows the same comparison as in Fig. 5.3 but with a window width of 10*s*. This figure tells us the maximum value of distance covered over a window period is bounded. This upper bound will be determined by the maximum rate at which subjects can repeat the activities. Thus, for the subject of this trial, distance accumulated over a window of 10*s* is close to 120 and all other higher distances are too sporadic to be considered for analysis. This study is applicable to those people who are chronic stroke survivors and have some functional ability (like movement of elbow, stretching of fingers) in their affected arm. Maybe later on, once the subject has recovered some abilities partially a wider window width of 10*s* or more will be more useful. However, as of now, window width of 5*s* is the most promising choice.

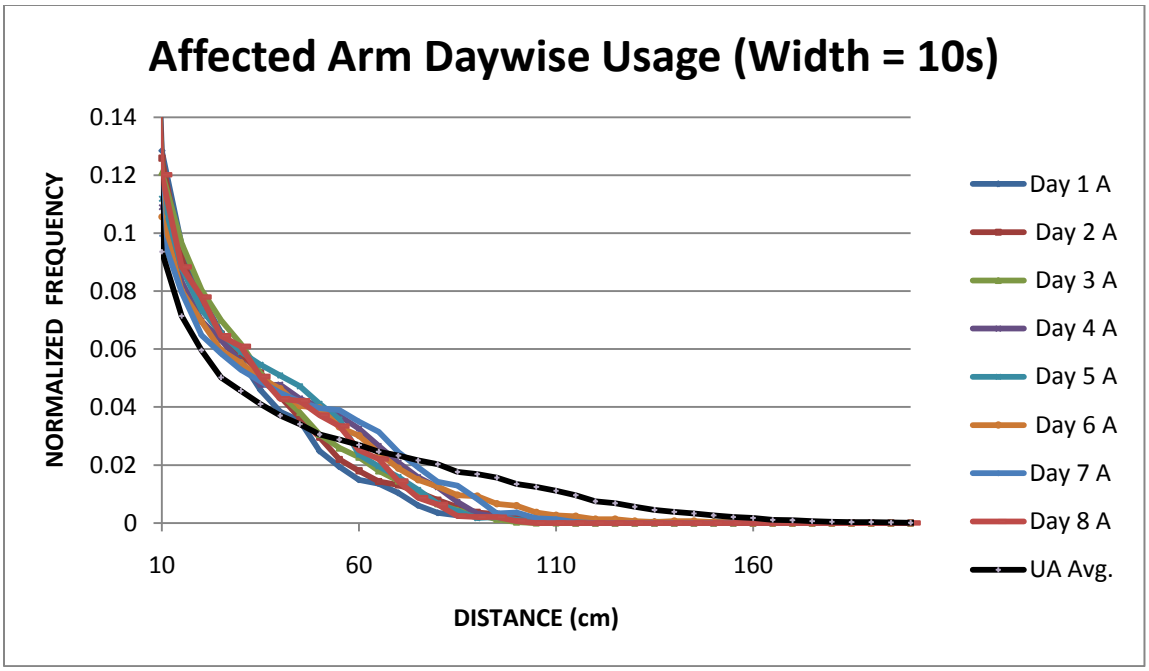

Fig. 5.4 Normalized Histogram of Distance Covered with width = 10*s*

Figures 5.2 through 5.4 represent a single subject's data. Data from other 2 subjects' trials also shows similar behavior as seen in Fig. 5.2 to Fig. 5.4 respectively. This reinforces our choice of window width of 5*s* being generic for the class of subjects under consideration. The results obtained from remaining two subjects' data trials can be found in Appendix B.1.

5.2.2 Performance Metric Observability/Resolution

Current version of CIMT uses subjective performance tracking methods, which suffer from very low temporal resolution. These methods take longer to notice any change in performance metric making the whole CIMT inefficient and slow. This in turn results in rising cost of health care for stroke patients. With accelerometers, we will develop an objective performance tracking method with low temporal resolution (i.e. it will be able to notice changes in performance metric sooner than current available methods).

Figure 5.5 shows zoomed-in view of Fig. 5.3. From this figure, we can see that over a period of 8 days the overall trend of movement of histogram of distance covered by affected arm is towards the average histogram of distance traversed by unaffected arm over the same period. But, this progression is not monotonous enough causing day 5 to be the worst for higher distance

values (instead of day 1) and day 6 & 7 to the best for higher distances (instead of day 8). These inter-day variations are attributed to changes in physical and/or psychological state (fatigue) of subjects.

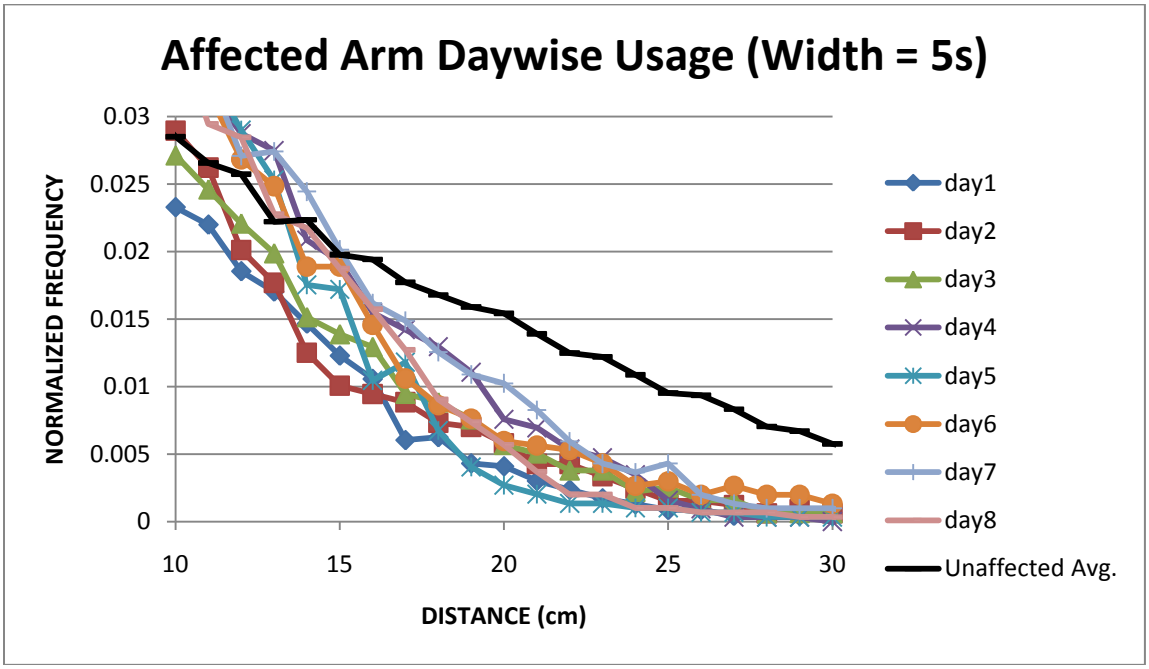

Fig. 5.5 Zoomed-in version of Fig. 5.3

To remove the effect of inter-day variations and emphasize the overall improvement, average histograms of distance covered over a period of some consecutive days are plotted. Figures 5.6 and Fig. 5.7 show 2-day average histograms and 4-day average histograms respectively. As shown in Fig. 5.6, with 2-day average histograms, we get rid of most of the variations and initial 2-day period turns out to be the worst in performance at higher distances while as last 2-day period turns out to be the best. Two 2-day plots in the middle are dominated by considerable noise and we do not see any clear indication of incremental improvement among those two.

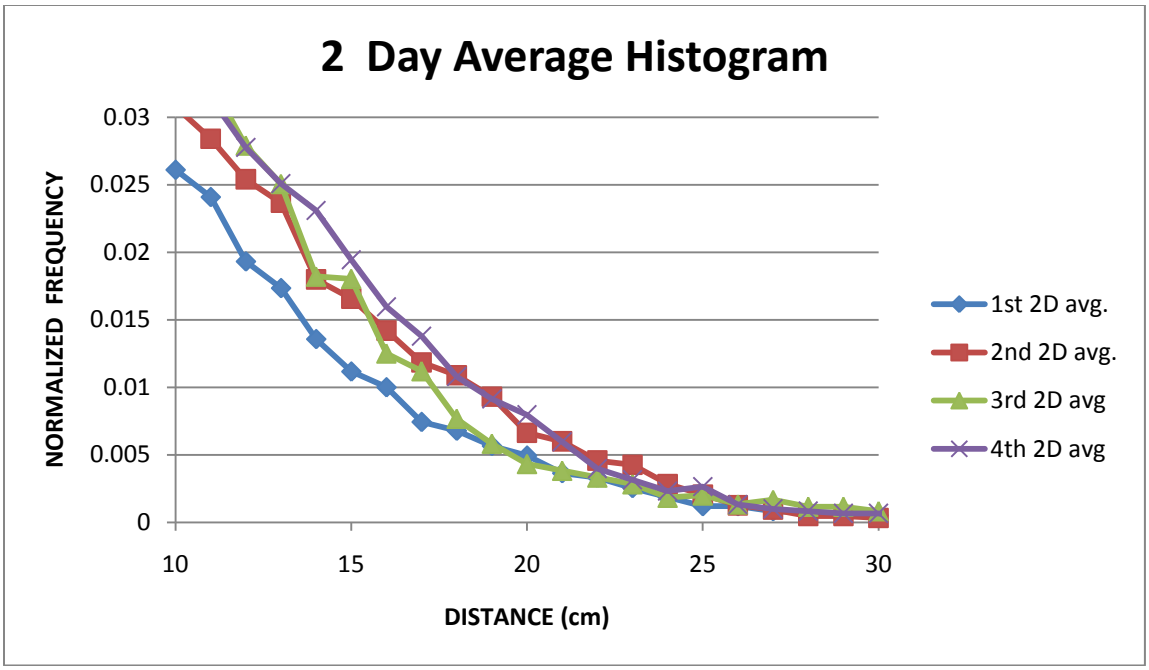

Fig. 5.6 2-day Average Histograms of Distance Covered by Affected Arm

As can be seen from Fig. 5.7, with 4-day average histograms, we see some remarkable improvement at low distance values but at high distance values, it is not that remarkable.

Thus, to observe gross performance improvement average histogram over some consecutive days should be looked at. This new performance metric obtained using accelerometer data does show better temporal resolution. In addition, this resolution can be tweaked around to compensate noise (day-to-day variations).

5.3 Histogram of Distance Traversed Based Comparison

To verify the algorithm described in earlier section, limited clinical trial was carried with the help from NRRL staff members. The number of subjects available limited this clinical trial. The trial consisted of five subjects with chronic stroke and each one having differing movement capability in their affected arm.

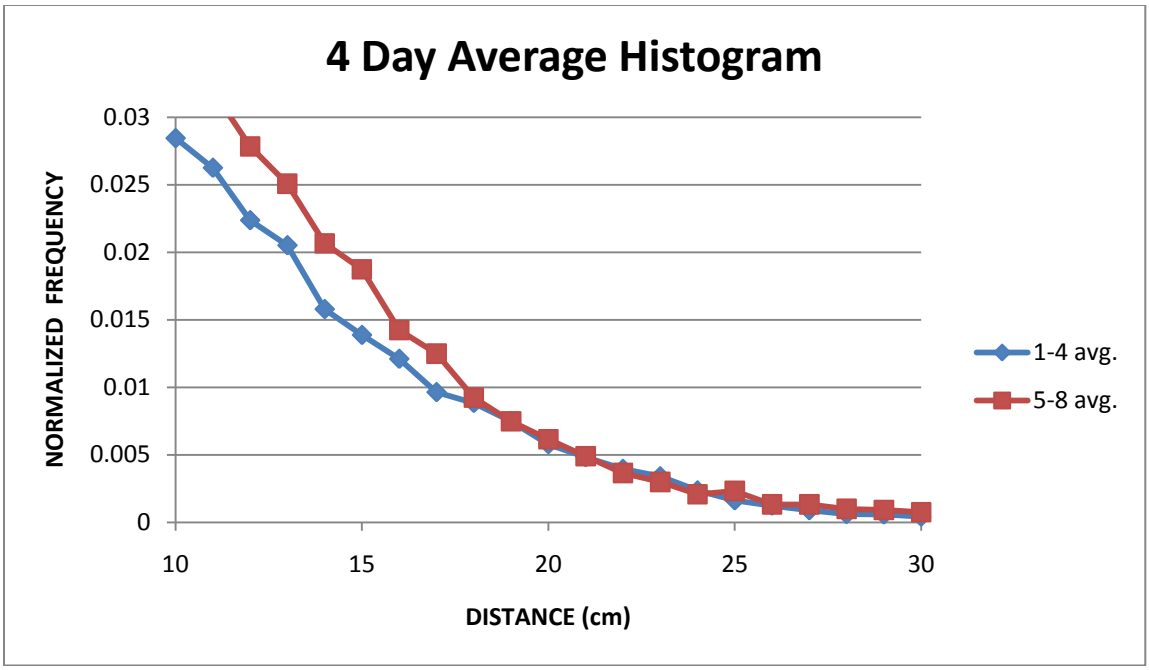

Fig. 5.7 4-day Average Histograms of Distance Covered by Affected Arm

Each subject participating in this trial completed MAL and BBT tests. As mentioned earlier, MAL is a subjective test while as BBT is an objective test developed at NRRL, CSU. After these tests, subjects completed four activities with repetition. The activities were designed such that two of them are considered to be uniaxial, one is considered to be biaxial while as the remaining one is considered to be 3D activity. Fourth activity is a more realistic representation of general ADL.

First activity (‘*activity 1*’) consisted of sliding in 10 small gunny bags placed at some predefined distance (about one foot). This activity is designed to be predominantly along Y-axis of 3D accelerometer. Second one (‘*activity 2*’) consisted of filling in an empty egg crate of 12 eggs with woolen/cotton balls. This activity is representative of 3D ADL and involves fine motor skills along with gross motor skills. The proposed algorithm is designed for gross motor skills only and will not be of much help for analyzing fine motor skills. Third activity (‘*activity 3*’) consisted of lifting an empty plastic cup 10 times as if working out bicep muscles. This activity is designed to be dominantly along X and Y-axis of 3D accelerometer. This activity is a 2D activity. The last activity (‘*activity 4*’) consisted of sliding out 10 shower curtain rings from the shower

curtain bar. This activity is designed to be along either X or Z-axis of 3D accelerometer depending on each individual's convenience during the activity. Table 5.1 summarizes the activities used in data trials along with their characteristics.

Table 5.1 Summary of Activities used in 8-day trial

Activity	Motor Skills	Dimensionality	Description
activity 1	Gross	1D (Y)	Slide in 10 gunny bags
activity 2	Fine	3D	Fill in egg crate with cotton balls
activity 3	Gross	2D (X and Y)	Lifting an empty plastic cup
activity 4	Gross	1D (X or Z)	Sliding out curtain rings from shower curtain bar

Table 5.2 shows the outcomes of MAL and BBT tests arranged in descending order of BBT scores.

Table 5.2 Scores of MAL and BBT

Subject	BBT Score	MAL Score (For completed items)
Subject 2	45	4.33
Subject 4	19	2.79
Subject 3	17	3.07
Subject 5	11	2.64
Subject 1	3	2.19

As expected, BBT and MAL score based performance metrics do not agree with each other because of some obvious differences between their methodologies as described earlier. For histogram algorithm's verification purposes, we should only consider BBT test scores as it is an objective test.

Figure 5.8 shows the histogram of distance traversed by all subjects for activity 1. From this histogram we conclude that subject 1 is the best performing, followed by subject 2, subject 3, and subject 4. Subject 5 appears to be the worst performing of all. The histogram is plotted from a minimum distance of 60 cm to a maximum distance of 120 cm. Maximum distance of 120 cm was decided based on earlier trials of 3 subjects where it was found that any subject would rarely cover more than 120 cm in selected window width of 5 seconds. Minimum distance of 60 cm is chosen, as we do not want to look at low distance histograms.

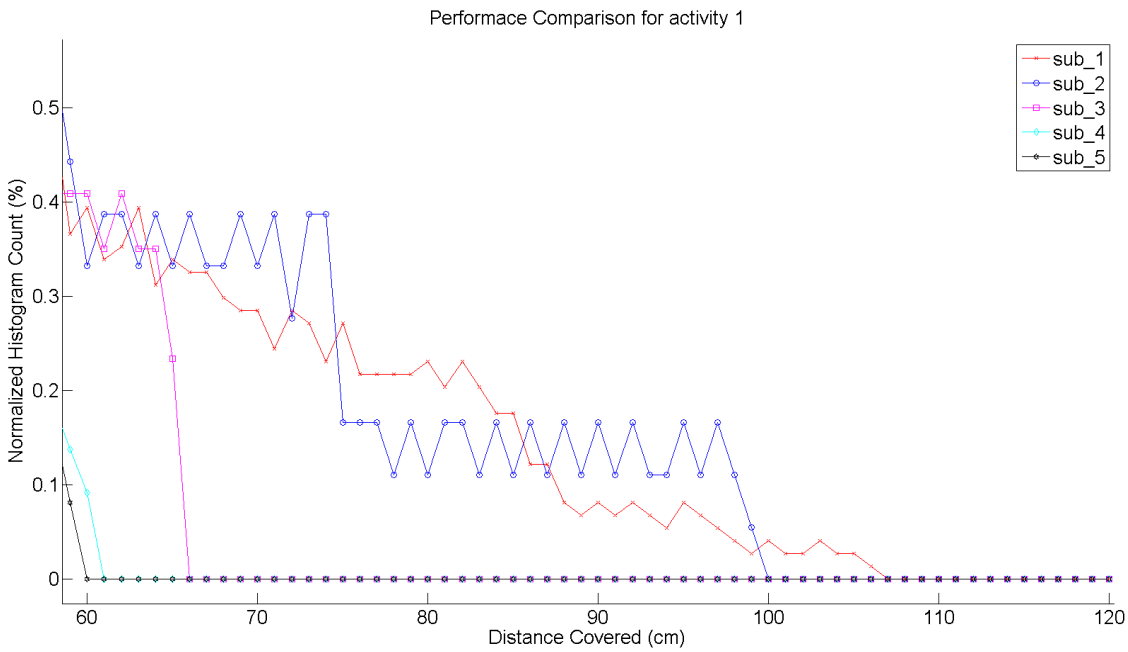

Fig. 5.8 Histogram of Distance Traversed by all subjects for activity 1

Observing Fig. 5.8, we can see that subject 1 covers higher distance more often than any other subject and hence is ranked first. Subject 5 covers maximum distance least frequently than any other subject and thus, subject 5 has the worst performance. Similar procedure is followed for all other activities and each subject's performance was ranked based on histogram method. All results are compiled into a table and arranged by BBT score ranks.

Column 7 in the Table 5.3 gives the average rank over all 4 activities by histogram method while as column 8 gives the average rank over 3 activities, which are performed either in 1D or 2D (i.e. activity 1, activity 3 and activity 4), based on histogram method.

Table 5.3 Histogram based comparison summary

Participant	BBT Rank	*activity 1*	*activity 2*	*activity 3*	*activity 4*	Average (4 activities)	Average (3 uniaxial activities)
Subject 2	1 (45)	2	2	1	2	1.75	1.67
Subject 4	2 (19)	4	5	2	3	3.5	3
Subject 3	3 (17)	3	3	5	4	3.75	4
Subject 5	4 (11)	5	4	4	5	4.5	4.67
Subject 1	5 (03)	1	1	3	1	1.5	1.67

The unexpected result from the Table 5.3 is, disagreement between BBT and histogram method over subject 1's performance. To check what might have gone wrong, subject 1's

videotaped trial was reviewed in collaboration with NRRL staff. Video review concluded that though subject 1 has had chronic stroke, the subject had a severe stroke resulting in very high motor deficits. Thus, the subject lacks even basic gross movements and experiences frequent involuntary sudden movements also known as 'jerks'. The magnitude of acceleration of these involuntary movements is close to that of normal ADLs. Only difference between jerks and ADLs is the time duration for which they last. Jerks are short time phenomenon while as ADLs last longer. The current version of algorithm cannot differentiate among these two as it considers amplitude of accelerations only and not the duration.

On discarding subject 1 data, histogram method based ranking for each individual activity varies greatly. This implies that each subject cannot perform each activity with same proficiency. Columns 7 and 8 of Table 5.3 should be looked at as comprehensive rank for all 4 activities and all but 3D activities respectively, using histogram method. Again, if we discard subject 1, then these comprehensive rankings follow the same order as BBT score based rankings (sans subject 1).

This method of visually observing histogram to rank subjects' performance is tedious and will be prone to observational errors of the ranker, as we start comparing more subjects. Thus, there is a need to have a numeric value as the output of histogram algorithm so that it would be easier for the therapist/physician to comprehend/assess the rehabilitation progress. The area under the histogram curve can be used as an indicator to compare performances. Though normal area calculation might seem promising, it might not be the best method, as it does not differentiate among different distances. The ideal candidate to get this numerical quantity should assign more importance to higher distances over lower distances as higher distance traversal is a good indicator of better functional ability. This can be achieved by assigning different weights to different distances.

Observing all of the histograms presented so far, we infer following - as distance traversed increases, the frequency with which that distance can be traversed decreases. Thus, the

weighting function should increase weights as distance traversed goes on increasing. The rate of increase in weights needs to be determined. The ideal weighting scheme should be such that, it maintains the product of weighted distance traversed and its normalized frequency a constant. This ensures that area contributed by each segment of the histogram plot is same and when any subject who is able to traverse more distance than other subjects will have more total area under the histogram as compared with others.

Generally, the histogram decays exponentially as distance traversed increases. We can assume the tail of exponential decay to be linear for larger distances and thus assign weights that increase linearly as distance traversed goes on increasing. As we are interested in larger distances only, we will ignore all distances below 60cm for this area calculation method. Distance of 60cm is assigned a weight of 0, 61cm gets a weight of 1, 62cm gets a weight of 2, so on and so forth. Such linearly weighted area calculation is done on 5 subject data presented earlier. Each subject's performance is ranked according to this weighted area under the histogram and the results are presented in Table 5.4.

Table 5.4 Linearly Weighted Area under Histogram based comparison summary

Participant	BBT Rank	*activity 1*	*activity 2*	*activity 3*	*activity 4*	Average (4)	Average (3 activities except 3D activity)
Subject 2	1	1	2	1	1	1.25	1
Subject 4	2	4	5	2	3	3.5	3
Subject 3	3	3	3	4	4	3.5	3.67
Subject 5	4	5	4	5	5	4.75	5
Subject 1	5	2	1	3	2	2	2.33

Table 5.4 shows that linearly weighted area calculation method gives different performances for all subjects for each activity. Average performance for 3 activities designed along the principal axes of 3D accelerometer is in agreement with BBT rank if subject 1's data is ignored. However, when average performance over all activities is concerned this method fails to differentiate among 2nd and 3rd ranked subjects i.e. subject 4 and subject 3 respectively. These results are obtained under the assumption of linear weighting. The inconsistency of results

suggests that linear weighting might not be the best (or the most accurate) weighting scheme and other schemes need to be investigated.

Source code for the histogram of distance traversed algorithm is given in appendix A.2. The source code for calculating weighted area under the histogram curve is given in appendix A.3.

5.4 Summary

This chapter has introduced a novel algorithm to assess rehabilitation performance using 3D accelerometer. The algorithm uses statistics generated from accelerometer data within fixed time duration. The algorithm's resolution is obtained from consecutive 8-day trials for three subjects. The algorithm shows noticeable improvements in performance over a period of 8-day when two consecutive day's data is analyzed together. The algorithm's performance for single day analysis is limited by its susceptibility to sensor noise. Thus, the algorithm is extended to work on multiple days' data. This extension is more robust and accurate when compared with single day analysis algorithm. The algorithm is used to differentiate different stroke survivors based on their functional ability of affected arm. We also present a method to automate this algorithm to get rid of observational errors that might arise. The proposed method is based on weighted area calculation. Data analysis presented in this chapter leads us to conclude that linear weighting scheme works well for 1D and 2D activities.

Chapter 6

Tracking Arm Motion Using IMU

Using the algorithm to detect affected arm use, a method to determine the number of times a particular activity is repeated during a CIMT lab session was developed and presented in Chapter 4. Chapter 5 took the developments in Chapter 4 a step further by presenting a novel algorithm to track rehabilitation performance using 3D accelerometer. The algorithm presented in Chapter 5 uses statistical properties of data generated by 3D accelerometer during a specified time window to calculate an objective metric to assess rehabilitation progress. The analysis of resolution of Chapter 5 algorithm proved that the objective metric developed is more responsive than BBT scores. These objective metrics are necessary to observe rehabilitation progress as there is no method available to monitor arm movements remotely. However, recent advances in MEMS sensors enable us to design a wearable sensory system, which can track human motions reliably. With availability of such a system, therapists would be able to track affected arm motions remotely and recreate it whenever necessary to observe performance improvement over time. This chapter will present a Kalman filter based motion tracking system that can track linear motions performed along the primary sensing axes of 3D accelerometer. The algorithm that will be presented in this chapter assumes that there will not be any change in the orientation of the sensor and the linear motion will happen along any one of the primary axes at any point of time. Even with these assumptions, we will show that this algorithm will be useful in tracking stroke rehabilitation progress. A motion tracking system based exclusively on IMUs for human arm motion tracking purposes is the primary contribution of this chapter.

The rest of the chapter is organized as follows. Section 6.1 discusses the motivation behind this work followed by literature review of IMU bases motion tracking systems in Section

6.2. Section 6.3 presents an overview of the sensor used in this chapter followed by a brief discussion on sensor calibration. Section 6.4 discusses the simplified case of 1D motion-tracking algorithm. Section 6.5 presents the Kalman filter, which forms the core part of motion tracking algorithm. For the analysis of Kalman filter and determination of its optimal parameters, Section 6.6 presents a human arm motion simulator. Section 6.7 analyzes the Kalman filter and documents the impact of its parameters on the algorithm's performance. As mentioned earlier this algorithm is designed for linear motions in 2D. A discrete band pass filter is presented in Section 6.8 to minimize the impact of orientation change, which is a result of rotational motion, on algorithm's performance. Section 6.9 documents the results from sensor field trials consisting various representative set of motions from simple 1D motions to complex 2D motions. Section 6.10 presents an analysis of cross-axis sensitivity, which limits the algorithm's performance in complex 2D motions. Section 6.11 presents a method to identify components of 2D motion and to remove effects of cross-axis sensitivity to improve the algorithm's performance. Section 6.12 discusses applications of this algorithm for stroke rehabilitation in general and CIMT in particular followed by the concluding remarks summarizing the contributions of this chapter.

6.1 Motivation

Motion tracking has been a very well researched topic with the advancement of aeronautical field for military and civilian applications. Over the past decade or so, human motion tracking has gained interest among many researchers due to its perceived applications in recreation, athletic training, and consumer electronics.

Very recently, human motion tracking is being considered for some medical applications such as healthy ageing, recovery and rehabilitation from disabilities, and telemonitoring via ambulatory systems. These systems can be classified according to the position of the sensors and sources, or according to the motion-tracking techniques, e.g. electromagnetic position and orientation trackers, acoustic position and orientation trackers, mechanical position and orientation trackers, electrostatic position and orientation trackers, and video and electro-optical

tracking systems [55]. Zhou et. al. [56] suggests following classification for human motion tracking systems and reviews them in detail –

- non-vision based systems (e.g. MT9, G-link, MotionStar, InterSense, Polhemus, HASDMS-1 and glove-based devices),
- vision-based tracking systems with markers (e.g. Qualisys, VICON, PeakMotus, CODA, ReActor2, ELITE bitomech and APAS),
- vision based tracking systems without markers and robotic-guided tracking systems (e.g. Cozens, MANUS, MIME, ARM Guide and robotic arms etc.).

All of the above mentioned systems still pose some challenges for their use in rehabilitation methods. Some of these systems require specially configured instruments such as robots, cameras and software associated with them. This limits the use of such systems to a location where these resources are available. Some of the above systems require a set of skills from the user, such as some medical knowledge of human motion and some technical knowledge about sensors, to operate. Therefore, need a watchful eye of a lab staff/technician. Some of these systems restrict the motions that can be carried out by the subject. Such restrictive systems will produce biased results. VICON system uses set of cameras along with passive markers on subject's body to track subject's motion. Most of these systems are not portable - restricting their use within a limited space either at a physician's lab or at subject's home. Almost all of these systems are costly hence preventing a wide adoption among the health professionals.

The current state of CIMT requires subjects to come to a physician's lab to undergo therapy session. Physician monitors these sessions subjectively to track rehabilitation progress. This method of CIMT delivery not only increases the cost of rehabilitation but also gets limited by a physician's availability, making it harder to deliver CIMT to subjects living beyond some distance. Consequently, the average time for rehabilitation increases, putting economic burden on subjects. Though physician recommends subjects to practice the therapy while in their homes,

there is no method, which enables physician to monitor whether the therapy is being practiced or not, and if it is being practiced, how effective it has been.

IMUs offer a compelling solution to issues mentioned thus far. With advances in technology, MEMS sensors' performance improves whilst their cost reduces. MEMS sensors become smaller and lighter. Most of these MEMS sensors can communicate wirelessly and run off batteries. These are some of the characteristics that make IMUs ideal for portability. Thus, we need to develop a method/algorithm, which enables us to track arm motion using IMU. Developing this technology with off-the-shelf IMU units has a number of economical advantages. We can reduce the cost of the system significantly by giving all subjects a tracking system of their own and then physician can monitor all of them remotely.

6.2 Related Work

Over the last decade, there has been significant amount of research done in the field of human motion tracking using IMUs. Majority of that work has gone into understanding and modeling the MEMS sensors and characterizing them for noise analysis. After that, the focus moved to utilizing these sensors for estimating changes in orientation, caused by either intended or unintended rotational motion. So far, very few researchers have dived into the complex problem of tracking human motion using IMUs. Majority of those who explored this area did so for human gait analysis. Very few have explored the area of tracking human arm motion using IMUs. The aptly titled survey of motion tracking technologies presented by Welch and Foxlin [46], accurately describes the current state of this field of study – highly fragmented approaches and tailor-made solutions incorporating wide array of technologies, designed according to application domain under considerations.

Veltink et. al. [AC, AL] presented a thorough analysis of accelerometer error sources and modeled them to determine the gravity component acting on each axis. Kalman filter was used to predict and filter out the noise present in accelerometer output. Such noise corrected accelerometer output is then used to determine the orientation of the body the IMU is attached to.

For this work, it was assumed that all changes in accelerometer's output are solely due to changes in orientation and there was no linear motion associated with that change. This is very well justified assumption that makes the technique more accurate. However, we cannot use the same assumption for motion tracking purposes.

Najafi et. al. [28] proposed a method to monitor physical activities in elderly using an IMU attached to the subject's chest. Authors considered various postures in this study such as sitting, standing, rolling in/out of bed and physical activity of walking. Authors used discrete Wavelet transform as a signal-processing tool along with a simple kinematics model that ties various postural transitions to IMU signals. It was proved that this method is as good as optical motion tracking. The placement of sensor on chest has some implicit advantages as the axes of accelerometer stay fixed all the time with respect to the torso and human torso has less degree of freedom as compared to human arm. In addition, the orientation changes in torso are directly associated to postural transitions. This study neglected linear motion of any kind, as well.

Sabatini [34] presents a method for gait analysis using inertial sensors. The method uses quaternion based attitude determination using gyroscope signals. Then, accelerometer outputs are gravity compensated and double integrated to calculate distance. The sensors' noise sources are modeled and the errors are estimated using spherical linear interpolation (SLERP) technique. A simulator is developed to verify all of the above techniques. Although, this study presents a robust method for motion tracking, it lacks results from an actual sensor and field trial.

Torres et. al. [43] presents a motion-tracking algorithm using an IMU comprising of 3D accelerometers, 3D gyroscopes and magnetometer. The authors provide a thorough and in-depth solution for tracking motion in an Earth reference frame. They use magnetometer to determine true north and gyroscopes are used to determine orientation with respect to Earth reference frame. Once the orientation is determined, it is used to correct accelerometer output for gravity components and calculate position using Kalman filter as a data fusion tool. The algorithm fails to deliver primarily because of lack of sensor noise modeling. Gyroscopes and accelerometers are

accurate over a short period and start accumulating errors when integrated over long periods while as magnetometers are stable over longer period and are affected by environmental factors such as presence of metallic objects over a short period. This study does not model these and other possible error sources such as scale factors and bias, and hence the algorithm starts accumulating noise and produces error, which increases with time.

Thus, as described in this section, human motion tracking using IMU is still being investigated. A complete solution for 3D motion tracking using IMU still eludes us. This chapter will explore a solution involving Kalman filter. Considering the complexities associated with 3D motion tracking, this thesis will propose a solution for 1D motion tracking and will explore the possibility of extending it for 2D tracking purposes.

6.3 Atomic 6DOF

This part of the thesis uses the 'Atomic 6 Degrees of Freedom' IMU unit from SparkFun [16]. It's a stripped down IMU designed for a good performance at lower price. This unit consists of 3D accelerometer and three 1D gyroscopes aligned with each axis of accelerometer. It has Atmel ATMega328 microcontroller running at 10MHz with six dedicated 10-bit ADC channels reading the sensors. It runs wirelessly on XBee protocol at a baud rate of 115200bps. Its dimensions of 47mm x 37mm x 25mm make it an ideal candidate for portable applications. Fig. 6.1 shows the Atomic 6DOF unit pictured with XBee radio mounted on it.

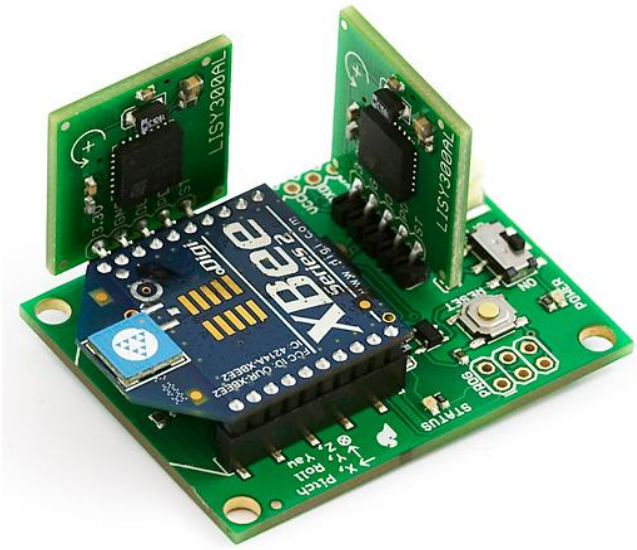

Fig. 6.1 Atomic 6DOF with XBee radio

The product datasheet [15] mentions that the gyroscopes have a resolution of 0.977°/s/tick where one tick is one ADC count. The accelerometers' sensitivity can be adjusted through firmware to one of the following values – 1.5*g*, 2*g*, 4*g*, and 6*g*. Accordingly, the accelerometer's resolution takes on following values respectively - 0.00403*g*/tick, 0.00537*g*/tick, 0.0107*g*/tick and 0.0161*g*/tick. During ADLs, human arm experiences accelerations in the range of about 0.2*g* – 0.3*g*. Thus, to improve the accuracy of results obtained, a sensitivity of 1.5*g* with maximum resolution is selected for this study. The unit samples data at a rate of 100Hz.

For motion tracking purposes, we want the absolute values of accelerations. Atomic 6DOF gives out raw ADC counts as its output. Thus, the accelerometer needs to be calibrated before being used for motion tracking purposes. The calibration of accelerometer is achieved by subjecting it to known acceleration values and mapping its ADC output to those known acceleration values. To subject accelerometer to known accelerations, the unit is mounted on a 5cm sine bar with its X-axis aligned along the length of sine bar and Y-axis aligned along the breadth of sine bar. With an inclination of θ, the X-axis experiences $g \cdot sin(\theta)$ while as Z-axis is subjected to $g \cdot cos(\theta)$. Enough samples are collected for a given inclination so that the results

obtained are statistically stable. Table 6.1 shows the results obtained for X-axis while as Table 6.2 shows the results obtained for Z-axis.

Table 6.1 Atomic 6DOF Calibration for X-axis

θ (Degree)	Samples	Average (ADC Count)	Standard deviation (ADC Count)	Acceleration (m/s^2)	ADC Count for 1*g*	Resolution (m/s^2)
0	2244	482.94	1.904	0	-	-
5	2423	462.18	1.919	0.854	238.210	0.041
10	2233	440.35	1.930	1.702	245.235	0.039
15	2270	419.41	1.912	2.538	245.461	0.040
20	2266	399.68	1.896	3.354	243.443	0.040
25	2443	379.19	1.899	4.144	245.483	0.040
30	2595	360.29	1.902	4.903	245.294	0.040
35	2227	342.42	1.885	5.624	244.992	0.040
40	2308	325.50	1.912	6.303	244.925	0.040
44	2573	312.80	1.919	6.812	244.918	0.040

ADC counts (ticks) required to get 1*g* signal in a particular axis is derived by correlating changes in sensor's ADC count output to the changes in actual acceleration for each θ. One other thing that we can observe from the Tables 6.1 and 6.2 is that for an inclination of 5°, the signal to noise ratio in sensor output is not good and thus the result that is obtained for that inclination does not agree well with other inclinations.

As can be observed from the Tables 6.1 and 6.2, for all axes the accelerometer has the specified resolution of 0.004*g*/tick. In addition, from the average value of each axis for all different θ, we can map different ADC counts to actual acceleration. This calibrates the accelerometer of Atomic 6DOF enabling us to use it for the purposes of human motion tracking. For this calibration and all future calculations, the value of *g* is assumed to be 9.80665 m/s^2. One key observation we can make from the Tables 6.1 and 6.2 is that all axes of accelerometer have a

standard deviation of about 2 ADC counts in their output. This is nothing but the measurement noise of MEMS sensors. Converting this measurement noise to SI units, we get a standard deviation of about 0.0804 m/s^2 or equivalently 8.04 cm/s^2. Usually during ADL the distances covered are of the order of few tens of centimeters. This much measurement noise level will be a limiting factor on the long term performance and accuracy of the system developed using this accelerometer, if not taken care of properly. For short term performance and accuracy there will not be a huge impact as the signal to noise ratio during an activity period will be very high.

Table 6.2 Atomic 6DOF Calibration for Y-axis

θ (Degree)	Samples	Average (ADC Count)	Standard deviation (ADC Count)	Actual acc (m/s^2)	ADC Count for 1*g*	Resolution (m/s^2)
0	2244	798.279	1.734	9.806	-	-
5	2423	796.956	1.702	9.769	347.723	0.028
10	2233	794.627	1.750	9.657	240.385	0.040
15	2270	789.759	1.734	9.472	250.067	0.039
20	2266	783.556	1.685	9.215	244.138	0.040
25	2443	774.895	1.729	8.887	249.585	0.039
30	2595	764.877	1.721	8.492	249.318	0.039
35	2227	753.199	1.700	8.033	249.270	0.039
40	2308	739.801	1.720	7.512	249.954	0.039
44	2573	728.427	1.683	7.054	248.884	0.039

6.4 Motion Tracking in 1D

As described in Section 6.2 earlier, Torres et. al. [43] presented a motion-tracking algorithm involving an IMU consisting of accelerometers, gyroscopes and magnetometers. Fig. 6.2 shows the algorithm flowchart given in [43].

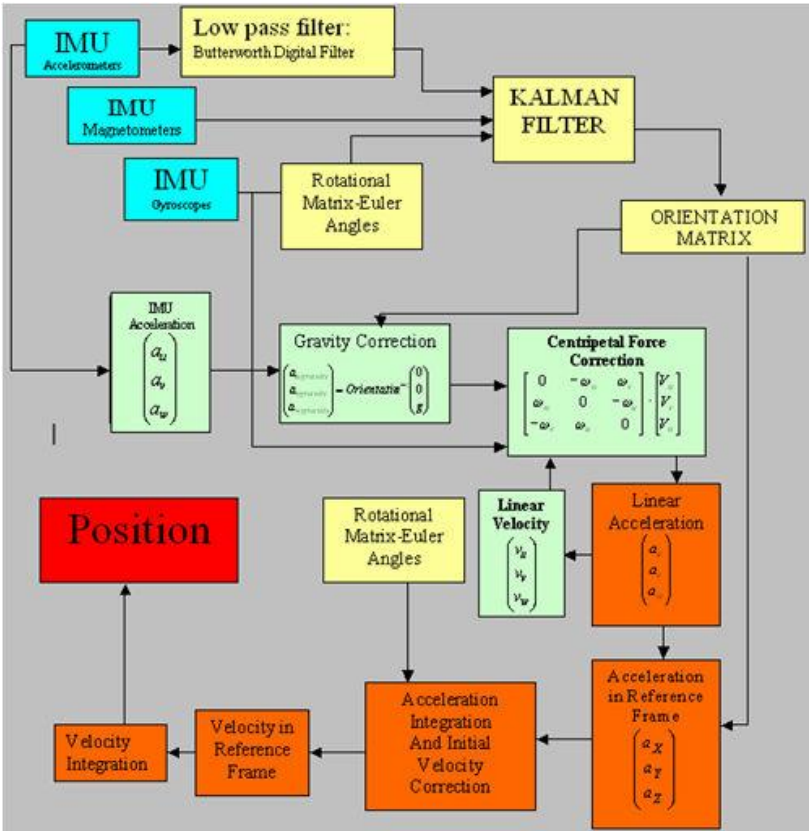

Fig. 6.2 Motion tracking algorithm flowchart presented in [43]

The algorithm uses Kalman filter to calculate orientation matrix. After the orientation matrix is calculated, the accelerometer outputs are compensated for orientation changes to obtain the component associated with linear motion. In 3D motion tracking, orientation estimation is the most critical component as any errors in orientation estimation are propagated much faster and result in significant errors in the outcome. Thus, to evaluate applicability and accuracy of accelerometers for motion tracking purposes, it is imperative to remove all the complexities of orientation changes and just considering pure linear motions. With this constraint in place, a method for tracking motion in 1D will be presented along with an extension, which will allow us to explore whether the method can be used as is for 2D motions (with no orientation change).

The sensor used in [43] has 3D accelerometer, 3D gyroscope, and 3D magnetometer. These sensors have all of their orthogonal axes aligned to each other and they form the sensor's frame of reference. Magnetometer measures the earth magnetic field on three orthogonal axes of

sensor's reference frame. With these sensors, it is possible to track motion in a fixed frame of reference such as Earth's fixed frame. Earth's fixed frame of reference is defined by X-axis being parallel to Earth's surface and pointing toward Earth's magnetic north, Z-axis parallel to the gravity vector and hence perpendicular to Earth's surface and Y-axis orthogonal to both X and Z axes such that they form a right handed co-ordinate frame [43]. Magnetometer is the primary link between Earth's fixed frame and sensor's frame of reference. As the sensor measures all quantities with respect to its frame of reference, an orientation matrix is needed to transpose measured quantities back into Earth's fixed frame of reference for tracking purposes. Atomic 6DOF IMU that we use in this section does not have a magnetometer and thus it cannot track the motion with Earth's fixed frame of reference. Instead, all the tracking that can be achieved using Atomic 6DOF will be with reference to sensor's frame of reference at the start of motion tracking. Considering short-term accuracy of gyroscopes and long-term stability of accelerometers and that in typical ADL, the ratio of duration of arm motion to the duration of no motion is always small; we can either enable or disable arm motion tracking depending on whether the arm is being moved actually or not. Then after each period of inactivity, motion tracking can start tracking the motion with reference to IMU's orientation at that instant and we do not have to worry about Earth's fixed frame of reference. The development of 1D motion-tracking algorithm is based on the assumption of no orientation change and hence this issue of changing frames of reference will not matter in 1D motion-tracking scenario.

6.5 Kalman Filter for Motion Tracking

In 1960, Rudolf E. Kálmán introduced his systematic approach to linear filtering based on the method of least squares [18]. Soon after its introduction, it became popular among engineers as it performed much better than most other techniques and it was easy to implement on digital computer. Its usage proliferated beyond engineering into diverse areas such as economics, vehicle navigation, weather prediction and many more. Along with performance improvements and ease of implementation, Kalman filter also had an advantage of analytical elegance [54] as

the filter design relied on the complete systematic analysis of the problem at hand. Kalman filtering has contributed immensely to the field of motion tracking in general. Kalman filter lends itself quite well for sensor data fusion. Thus, we will use Kalman filter for motion tracking purposes.

Kalman filter's ease of implementation for digital computers is an effect of its recursive nature. The recursive nature of Kalman filter implies that only the estimated state from previous time step and current measurement are needed to calculate current state estimate. Generally, a Kalman filter describes the system dynamics with following two fundamental equations –

$$X_k = F_k X_{k-1} + B_k u_k + w_k \tag{6.1}$$

$$z_k = H_k X_k + v_k \tag{6.2}$$

where X_k is the estimated state of the system at time step *k*. F_k is called as state transition matrix which relates estimated state from previous time step *k-1* to that of current time step *k*. B_k is control input matrix which transforms the control vector u_k into state. w_k is the process noise which is modeled as a zero mean multivariate normal distribution with covariance Q_k. z_k is the observed measurement at time step *k* which is related to system state X_k by measurement matrix H_k. v_k is the process noise which is modeled as a zero mean multivariate normal distribution with covariance R_k. Thus, w_k and v_k are represented as –

$$w_k \sim (0, Q_k) \tag{6.3}$$

$$v_k \sim (0, R_k) \tag{6.4}$$

All matrices that have a subscript can change each time step, but that is not a required condition for all systems and Kalman filter allows them to be constant for a system.

The kinematic equations for 1D motion in discrete time domain are –

$$s_k = s_{k-1} + dt \cdot v_k \tag{6.5}$$

$$v_k = v_{k-1} + dt \cdot a_k \tag{6.6}$$

where s_k denotes the position of the object being tracked at time step k, v_k denotes its velocity at time step k, and a_k denotes its acceleration at time step k. These quantities sufficiently describe the motion of the object at any time and thus they would form the state vector for motion tracking algorithm. Thus, we have –

$$X_k = \begin{bmatrix} s_k \\ v_k \\ a_k \end{bmatrix} \tag{6.7}$$

$$H_k = [0 \quad 0 \quad 1] \tag{6.8}$$

We know that during ADLs, non-motion period is longer than the motion period. When there is motion, the change in accelerometer output depends on not only the activity but also the initial orientation of the arm. There is no statistical model available for modeling human arm accelerations during ADLs. To relate acceleration for two consecutive time steps we model acceleration as first order autoregressive (AR) process. Based on this assumption and equations (6.5) and (6.6) we get –

$$F_k = \begin{bmatrix} 1 & dt & 0 \\ 0 & 1 & dt \\ 0 & 0 & \alpha \end{bmatrix} \tag{6.9}$$

α is AR parameter and we would find its value using a simulator. For the problem at hand, there is no control input vector available. Therefore, we neglect u_k and B_k from Kalman filter equation (6.1).

Equations (6.7), (6.8), and (6.9) represent the Kalman filter model for 1D motion tracking case. For 2D motion tracking case, these equations change as follows –

$$X_k = \begin{bmatrix} s_{xk} \\ s_{yk} \\ v_{xk} \\ v_{yk} \\ a_{xk} \\ a_{yk} \end{bmatrix} \tag{6.10}$$

$$H_k = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.11}$$

$$F_k = \begin{bmatrix} 1 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & dt \\ 0 & 0 & 0 & 0 & \alpha_x & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha_y \end{bmatrix} \tag{6.12}$$

For 2D motion tracking case, we have represented all state variables for both X-axis and Y-axis. The axis name in subscript for all variables indicates this fact. For F_k, we have used α_x and α_y as two different AR parameters just to take care of the case if different AR models exist for both axes. Nevertheless, for our analysis purposes we will assume both X and Y-axis have same AR model and thus we replace α_x and α_y by α.

6.6 Human Arm 1D Motion Simulator

We need to determine the parameters of Kalman filter discussed in earlier section that will work best for this study's purposes. We need to evaluate the filter's performance as well. To do this, we will need to simulate the human arm motion data and pass this controlled input to the filter and analyze the filter performance.

We can decompose any ADL into two primary phases – *reaching movement* and *object manipulation movement*. Reaching movements require gross motor skills while as object manipulation movements require fine motor skills. As we plan to construct a wearable device with IMU that is to be worn on the wrist, we will be unable to sense fine motor skills reliably. Thus, for motion tracking purposes we will focus on reaching movements (such as '*activity 1*' and '*activity 4*' introduced in Chapter 4).

With advancement of robotics, considerable work has been done to understand and analyze human motions. Human arm motion research has benefitted from this phenomenon. In general, human arm motion is a feedback based control system. Human brain processes the information obtained by various sources to plan and modify the trajectory of the motion. Many people have presented various models for human arm's motion control and trajectory planning. Hersch et. al. [13] have presented a modified VITE model [7] which relates the acceleration,

velocity and position of human arm during a reaching movement with that of target position. However, with IMUs, we cannot get any kind of feedback related to either the arm position or the target position and thus we cannot make use of this equation. For our purposes, we need a simpler model that concerns only with arm's position, velocity, and acceleration.

Experimental observations in [19] of planar human arm reaching movements indicated that humans tend to generate straight-line path and uni-modal bell shaped arm velocity profiles. Our brain achieves this by completing an underlying objective of achieving smoothest possible motion. Technically, smoothest movement corresponds to minimizing jerk (rate of change of acceleration) experienced by the arm. This particular model of human arm motion is called as 'minimum jerk theory'. This theory guarantees that the velocity profile generated by minimizing the jerk is always bell shaped. Based on this theory as well as some observations relating to reaching movements from 6DOF, we created a simulator to generate velocity, acceleration and position signals.

The simulator takes in sampling frequency, duration of activity, and amplitude of velocity as parameters. Based on these parameters it generates a bell shaped velocity profile. From this velocity signal, it calculates the acceleration and distance signals. The simulator generates velocities for both X-axis and Y-axis. Therefore, we can use this simulator not only for 1D motion-tracking model but also for 2D motion tracking model. For 2D motion tracking purposes, we have a top-level wrapper file encompassing the simulator. This wrapper file takes in the quadrant in which the user wants to simulate 2D rectangular track, sampling frequency and time duration for tracing one segment of the rectangle. Wrapper file calculates rest of the parameters and passes them on to simulator.

For a sampling frequency of 100Hz, first quadrant, a segment duration of 3 seconds the simulator generates the velocity profile in X-axis as shown in Fig. 6.3. Figure 6.4 shows the acceleration in X-axis generated by the simulator. Lastly, Fig. 6.5 shows the rectangular track generated by the simulator.

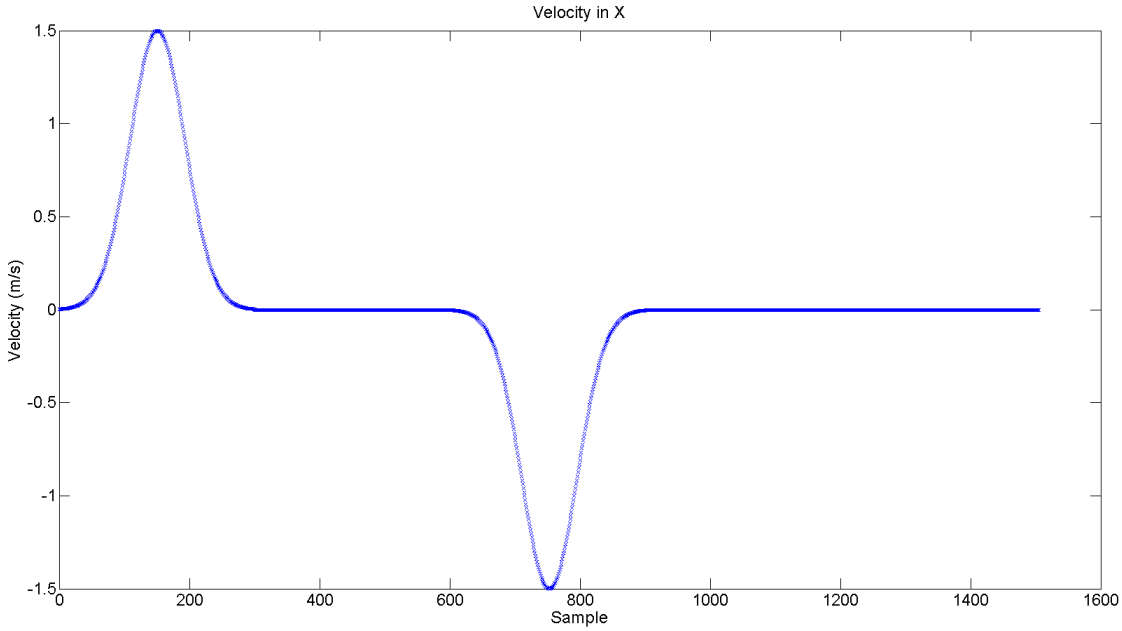

Fig. 6.3 Velocity generated by simulator in X-axis

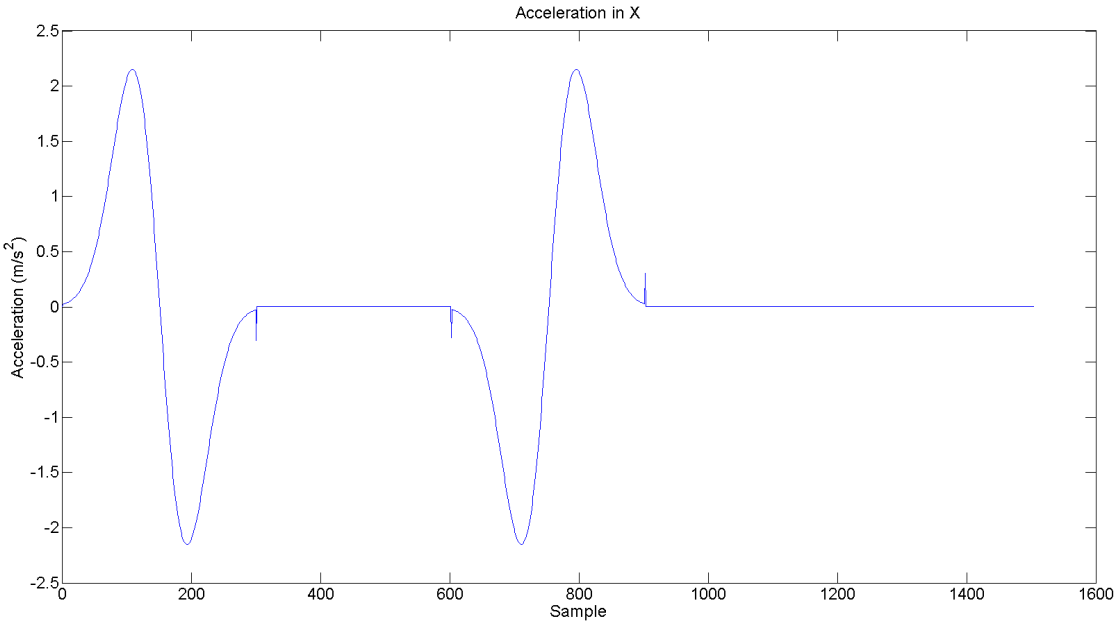

Fig. 6.4 Acceleration derived from velocity in X-axis of Fig. 6.3

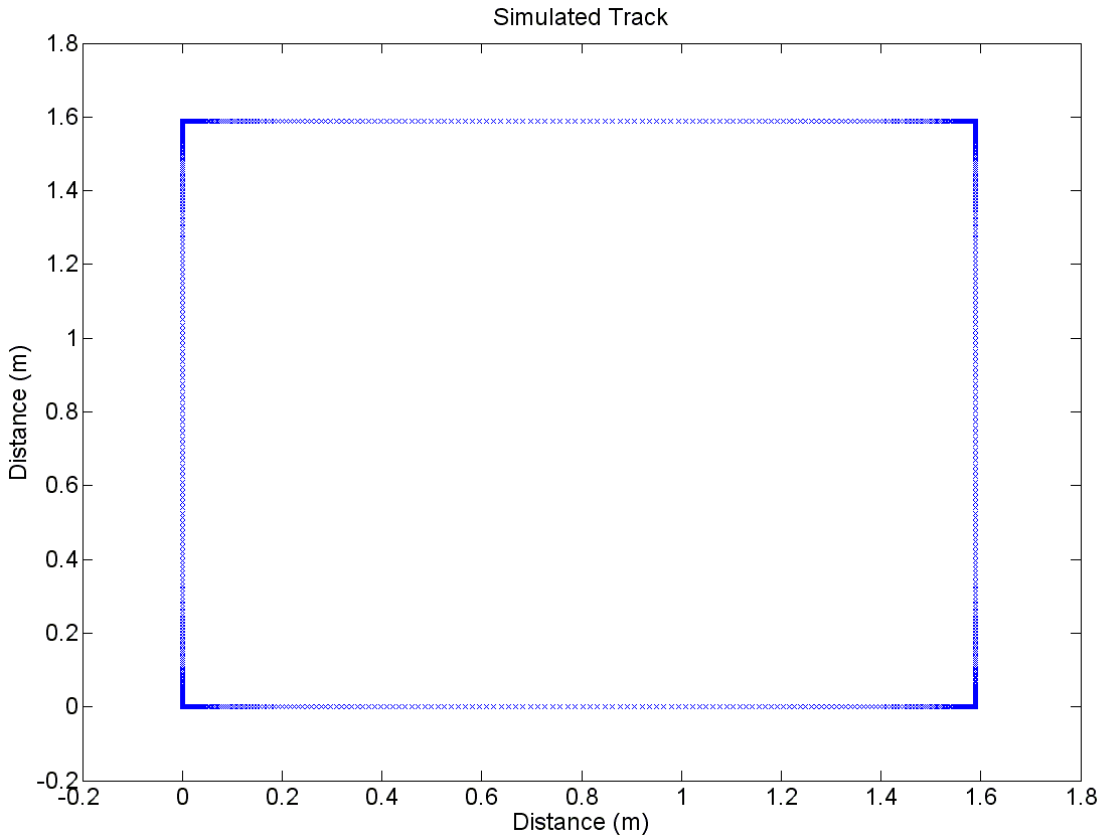

Fig. 6.5 Simulated test track

Appendix A.4 has the code for simulator and its wrapper file.

6.7 Analyzing Kalman Filter Performance

In Section 6.5, we presented the Kalman filter model for 1D motion tracking and extended it so that we can track 2D motion. To analyze Kalman filter's performance and to determine optimal values of its parameters for the best performance possible from the filter, we will use the simulator discussed in Section 6.6. For performance analysis of Kalman filter, we will simulate the arm motion and combine an additive noise representing the measurement noise expected in sensor output. We will then compare the filter's output with the values generated by simulator to see how well the filter filters out additive noise.

6.7.1 Stationary (Non-Motion) Case

When there is no motion, the accelerometer output will be constant with additive measurement noise. The orientation of the accelerometer determines this constant value.

Neglecting orientation, the accelerometer output will effectively be zero with measurement noise. Thus, to simulate this non-motion case, we generate a constant signal of value 0 and add measurement noise to it.

For Kalman filter, α will be set to 1, as we know that the acceleration is a constant signal. As there is no motion, there will not be any uncertainty about state variables namely position, velocity, and acceleration. Therefore, we will keep Q_k at 0.

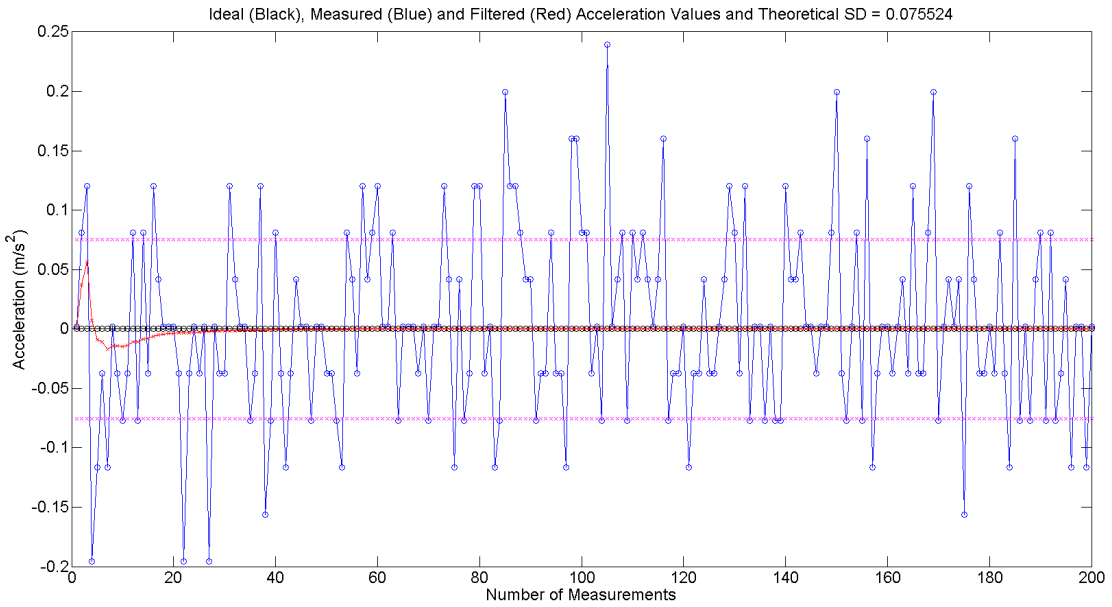

Fig. 6.6 Ideal (Black), Simulated (Blue) and Filtered (Red) acceleration values for stationary case with R_k = 0.075524. Pink trace marks the standard deviation of additive noise (R_k)

Figure 6.6 shows the ideal (black trace), simulated (blue trace), and filtered (red trace) acceleration values. As described earlier, additive measurement noise, representative of noise observed in Atomic 6DOF, corrupts ideal values to produce simulated values. The measurement noise is characterized by keeping Atomic 6DOF stationary for more than 10 minutes (i.e. more than 60,000 samples). The standard deviation obtained from this data is used as R_k for the simulation. Figure 6.7 shows the error in Kalman filter estimates as compared with ideal values. As can be seen in Fig. 6.7, the filter takes about 60 samples or so before settling down to value of 0. When we start off, the error in filter is high but soon that error reduces sharply and we start seeing accurate filtered output. Figure 6.8 depicts the covariance in X-axis acceleration values.

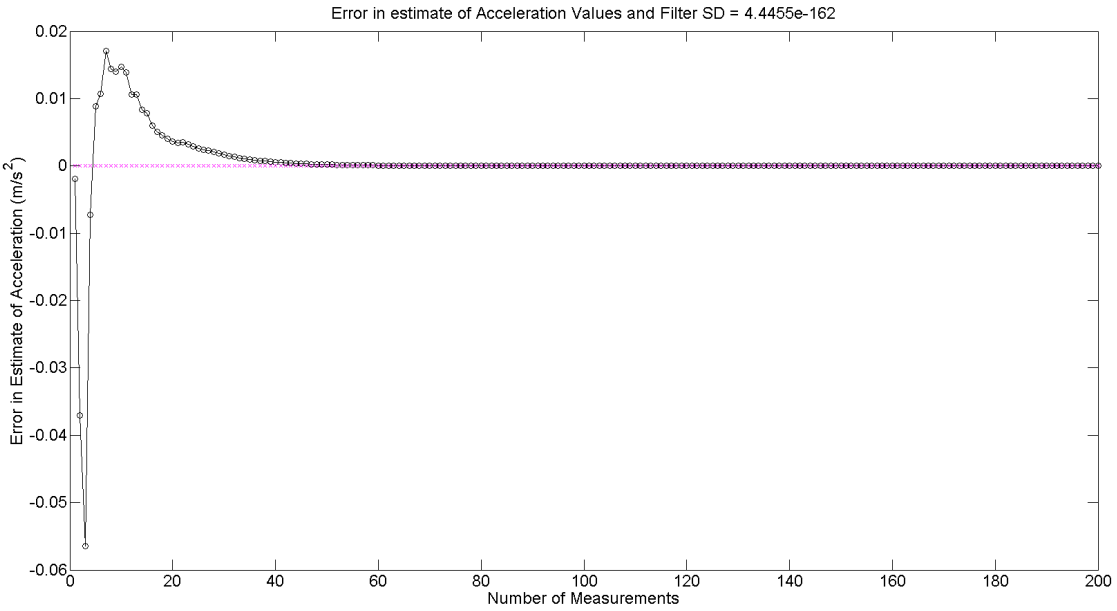

Fig. 6.7 Error in acceleration estimates

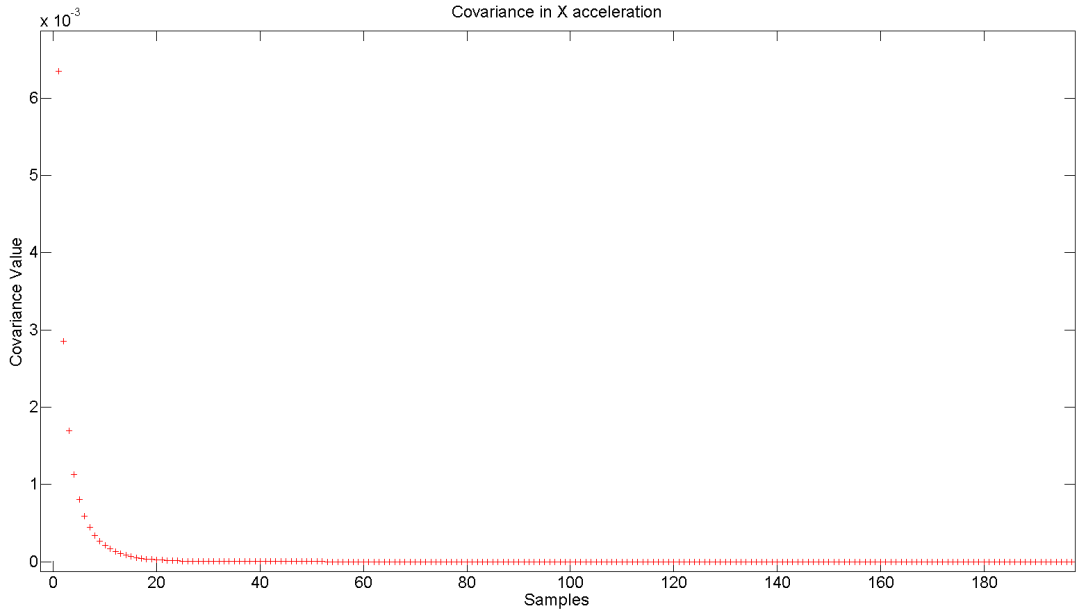

Fig. 6.8 Covariance of X-axis acceleration

As discussed in [54], covariance of state variables is one of the robust indicators of stability and validity of Kalman filter model for the problem at hand. If the filter is modeled properly, this covariance should decrease as more samples start coming in and finally settle down to low value and stay there for the remaining time. In Fig. 6.8, we do see same behavior, which validates accuracy of our model.

6.7.2 Dynamic (Motion) Case

During any motion, the acceleration will change to produce a bell shaped velocity profile. Depending on the direction of motion, the acceleration profile changes. Without any definitive model available to describe this acceleration profile as an AR process, we need to find optimal value of α that will enable us to track the acceleration.

Figure 6.9 shows the ideal, simulated, and filtered acceleration values when there exists some motion. For the purpose of Fig. 6.9, we set α to be 0.7 and Q_k to be zero. Figure 6.10 shows the error in estimating acceleration values for the dynamic case. It is evident that filter is estimating acceleration value to be zero all the time and thus error in estimate is almost the same signal as that of ideal values. We tried different values of α to find out that no matter what value α has; we cannot track acceleration until Q_k is zero.

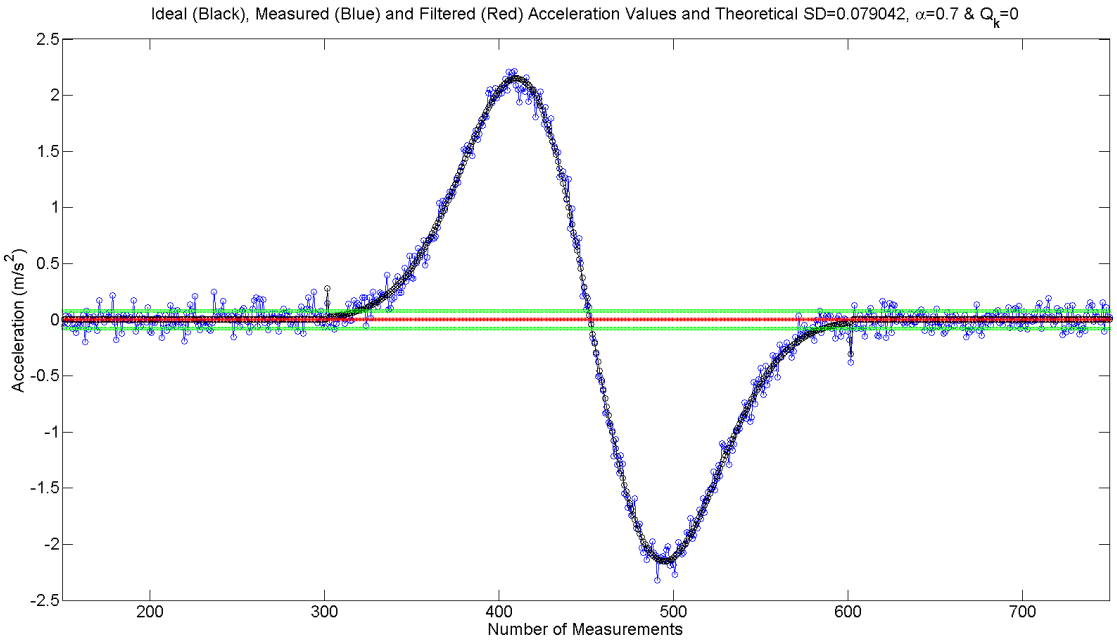

Fig. 6.9 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.075524, α = 0.7, and Q_k = 0

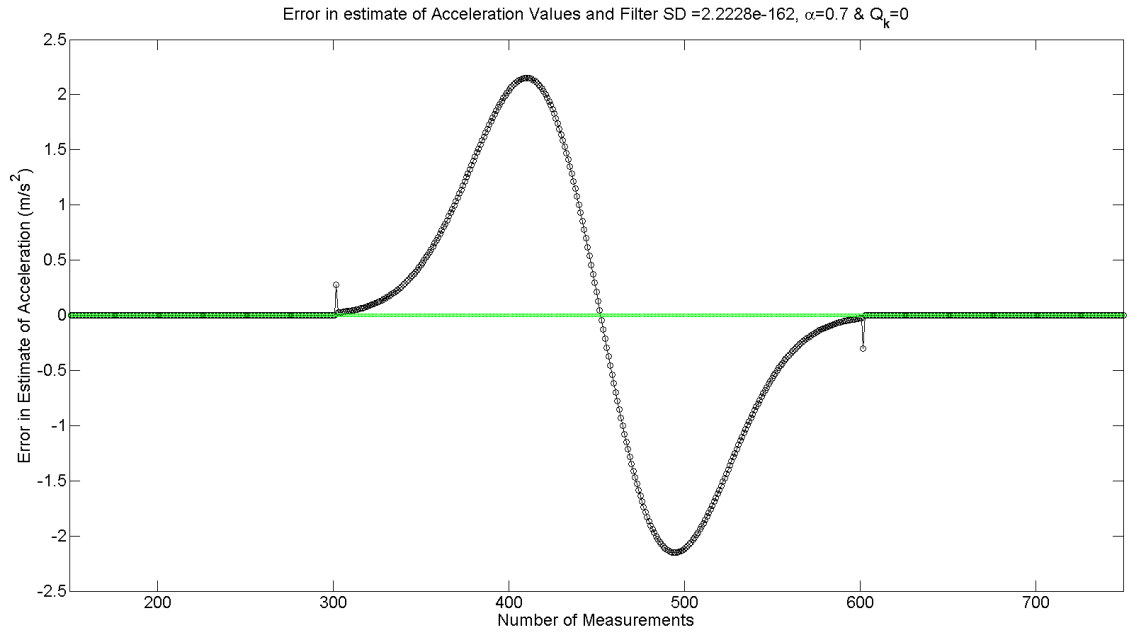

Fig. 6.10 Error in estimating acceleration values for dynamic case with R_k = 0.075524, α = 0.7, and Q_k= 0

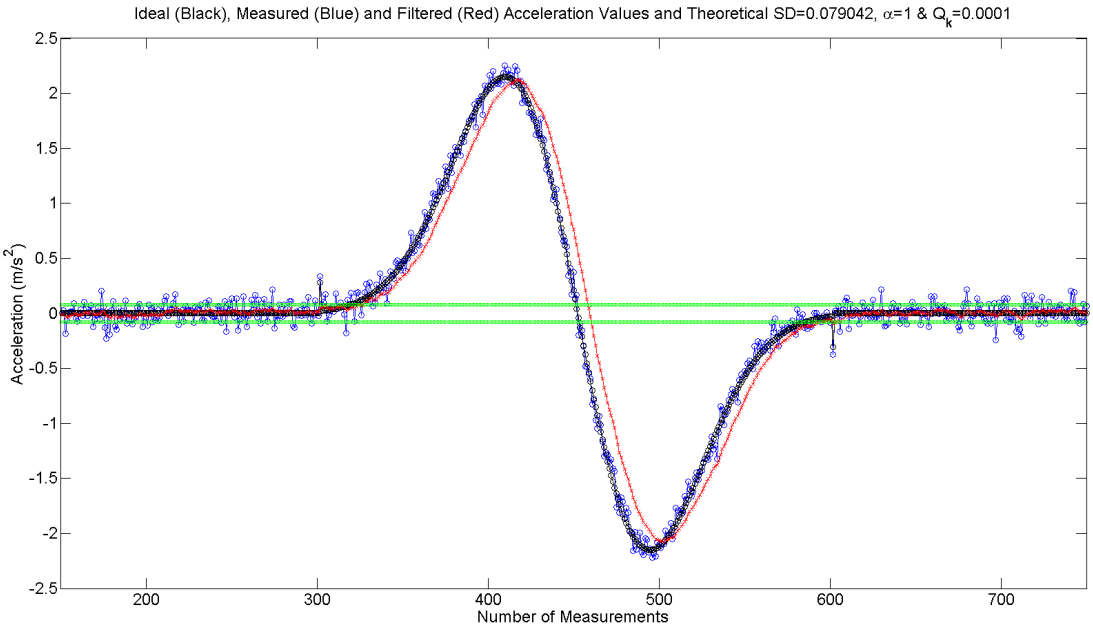

Fig. 6.11 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.0790, α = 1, and Q_k = 0.0001

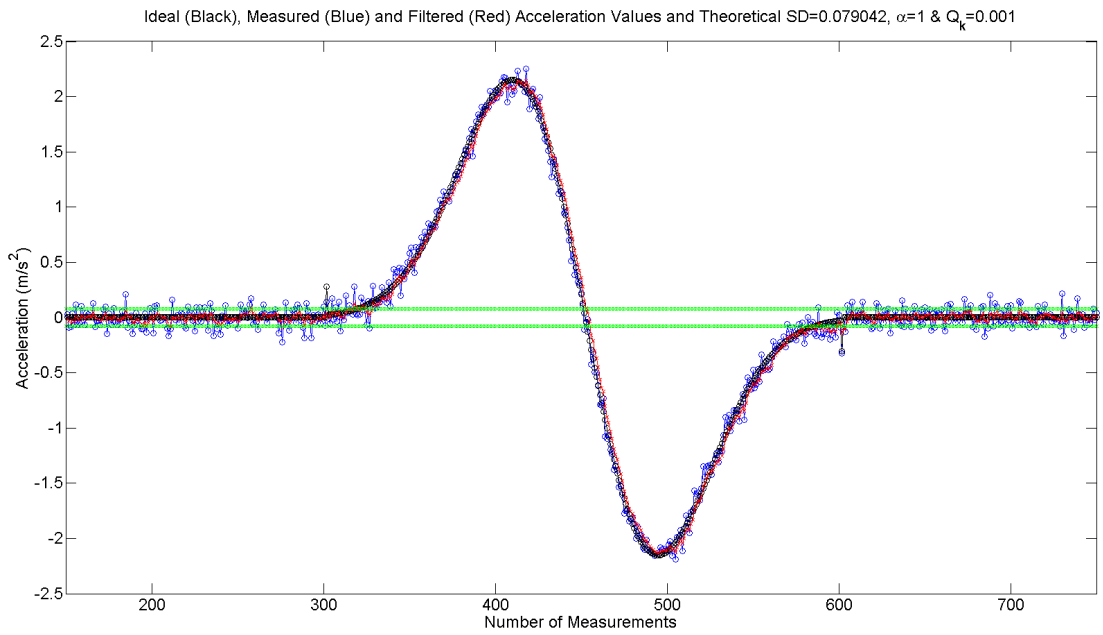

Fig. 6.12 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.0790, α = 1, and Q_k = 0.001

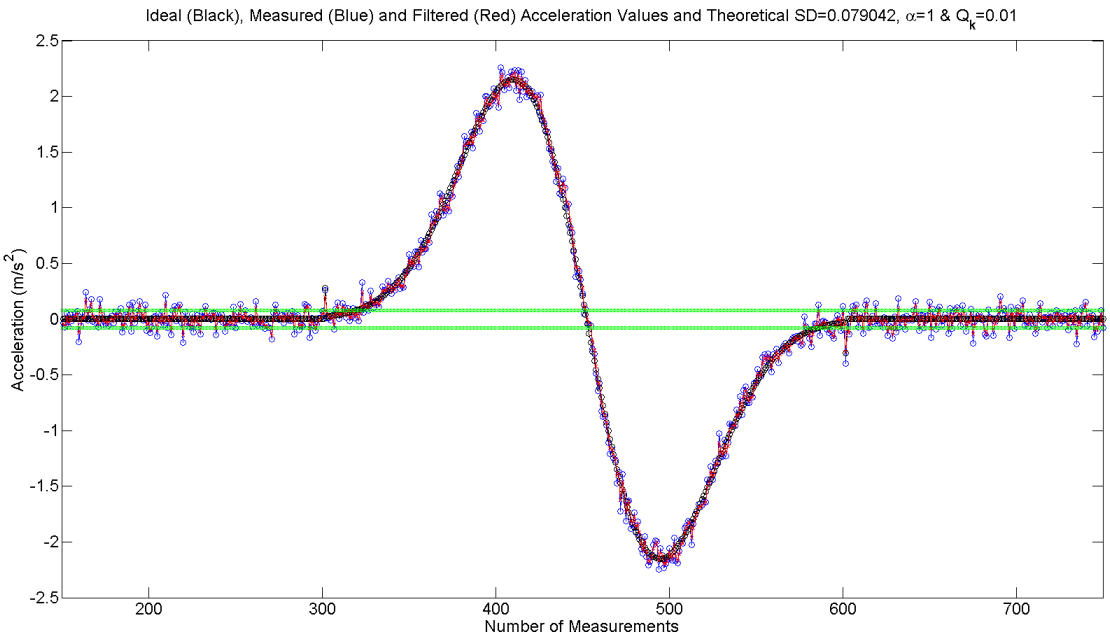

Fig. 6.13 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.0790, α = 1, and Q_k = 0.01

As described in earlier sections, we know that there will not be much uncertainty in determining state variables because we assume that changes in acceleration are just because of motion and nothing else. Thus, Q_k should be close to zero. To track the acceleration due to

motion, we do not want Q_k to be zero. Thus, we sweep through a range of values to determine optimal Q_k for motion tracking purposes.

Figures 6.11 through 6.13 show the ideal, simulated, and filtered acceleration values for a reaching movement with α set to 1 and Q_k varying from 0.0001, 0.001 to 0.01 respectively. For lower Q_k of 0.0001 we can see that the filter filters out most of the noise. However, the filter output lags the actual signal. In addition, it estimates signal amplitude lower than ideal. In the case of Q_k of 0.001 [Fig. 6.12], the filter does not filter all of the noise. However, it not only reduces the lag between estimates and the ideal signal but also tracks the ideal signal's amplitude closely. For Q_k of 0.01 [Fig. 6.13], the filter does not filter most of the noise. Thus, filter output in this scenario follows its input quite closely.

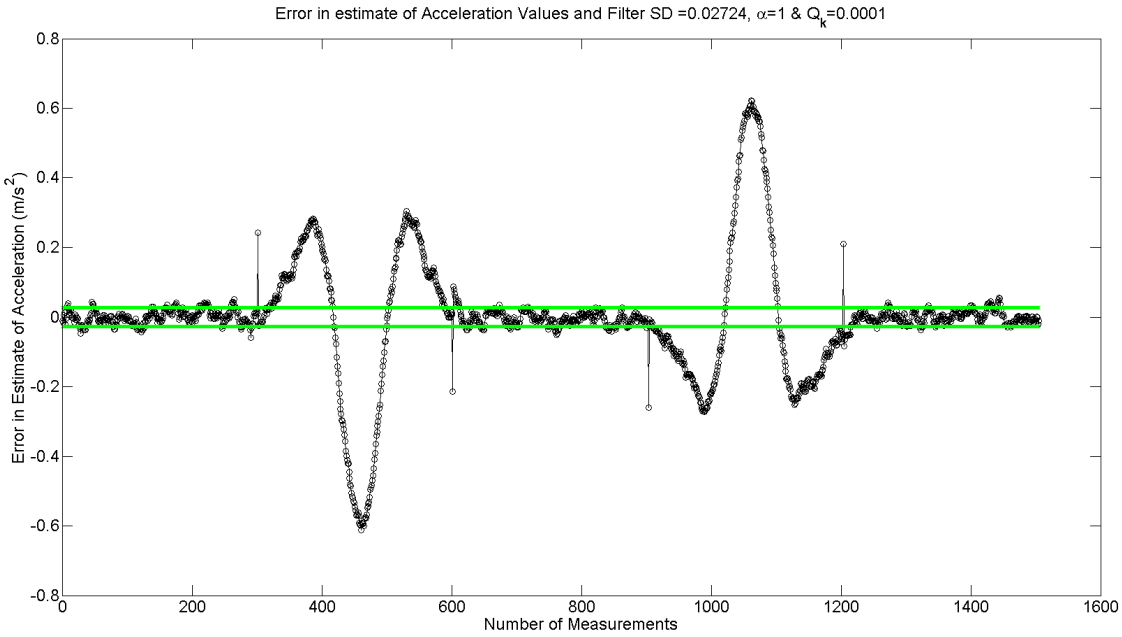

Fig. 6.14 Error in estimating acceleration values for dynamic case with R_k=0.079, α=1, and Q_k=0.0001

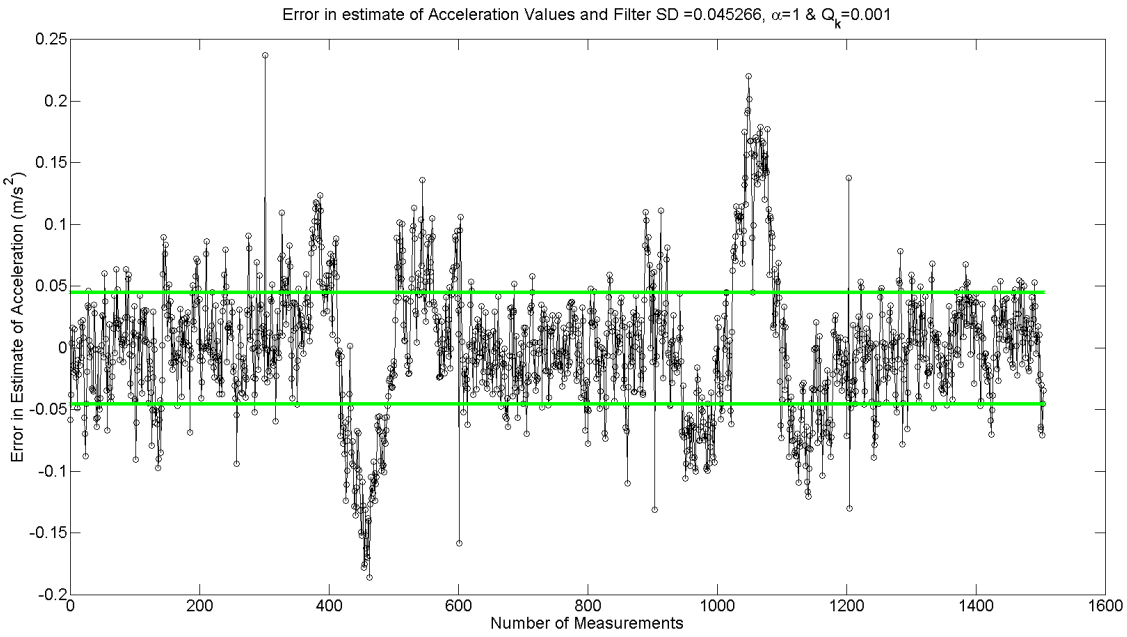

Fig. 6.15 Error in estimating acceleration values for dynamic case with R_k=0.079, α=1, and Q_k=0.001

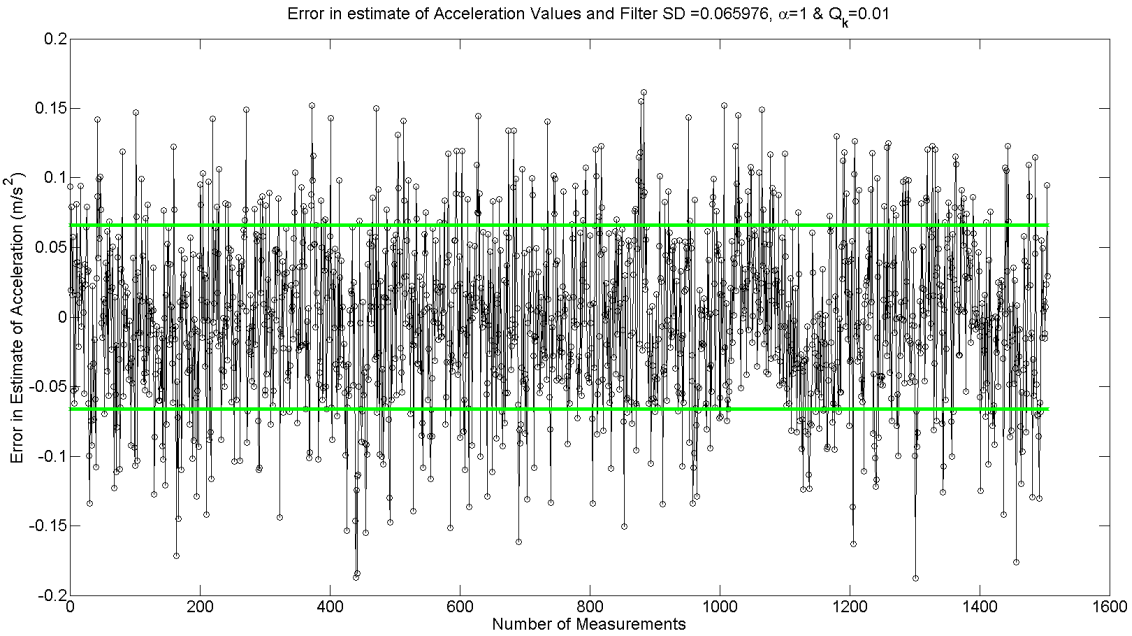

Fig. 6.16 Error in estimating acceleration values for dynamic case with R_k=0.079, α=1, and Q_k=0.01

Figures 6.14 through 6.16 show how change in Q_k affects the error in estimating acceleration for the Kalman filter. As Q_k increases, the standard deviation of error in estimating

acceleration increases. Put in other way, as Q_k increases, the Kalman filter settles to a higher covariance value.

Considering Fig. 6.11 through Fig. 6.16, we determine that Q_k = 0.001 is the optimal value that gives good acceleration estimates with sufficient noise filtering and acceptable noise in filtered output. With value of Qk determined, now we will present how the changes in α impact the Kalman filter performance.

6.7.2.2 Effect of α

The primary objective of this section is to find out if there exists some α which can improve Kalman filter performance for Q_k = 0.001. To determine value of α, we will vary α keeping Q_k constant. Value of α is changed from 0.1 to 1.5 to observe its impact on noise filtering and reducing standard deviation of error in estimating acceleration.

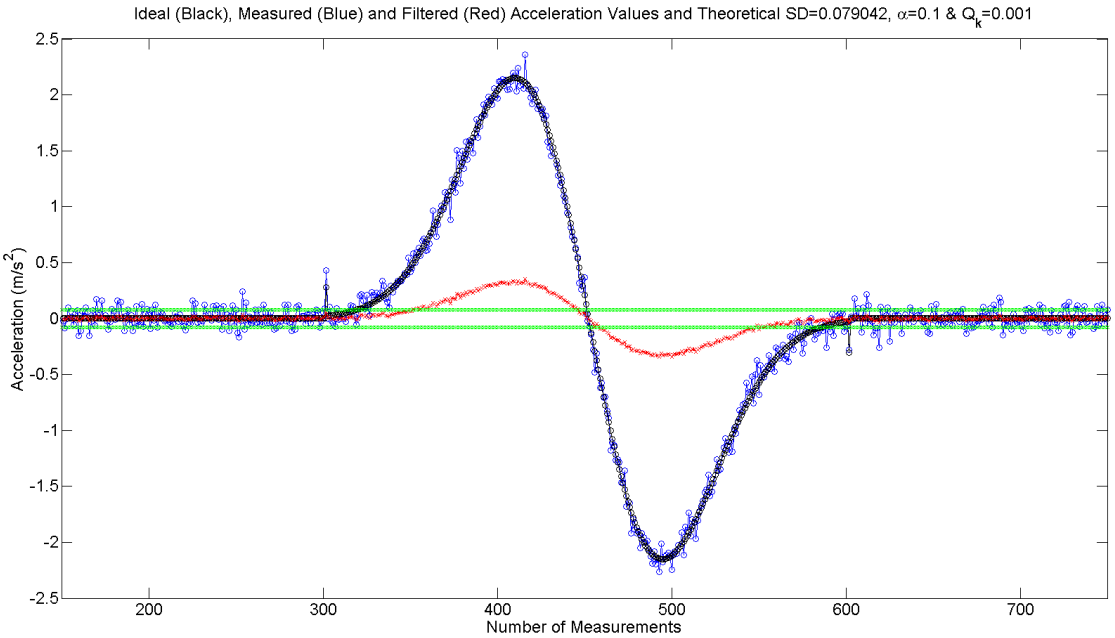

Fig. 6.17 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with Rk = 0.0790, α = 0.1, and Qk = 0.001

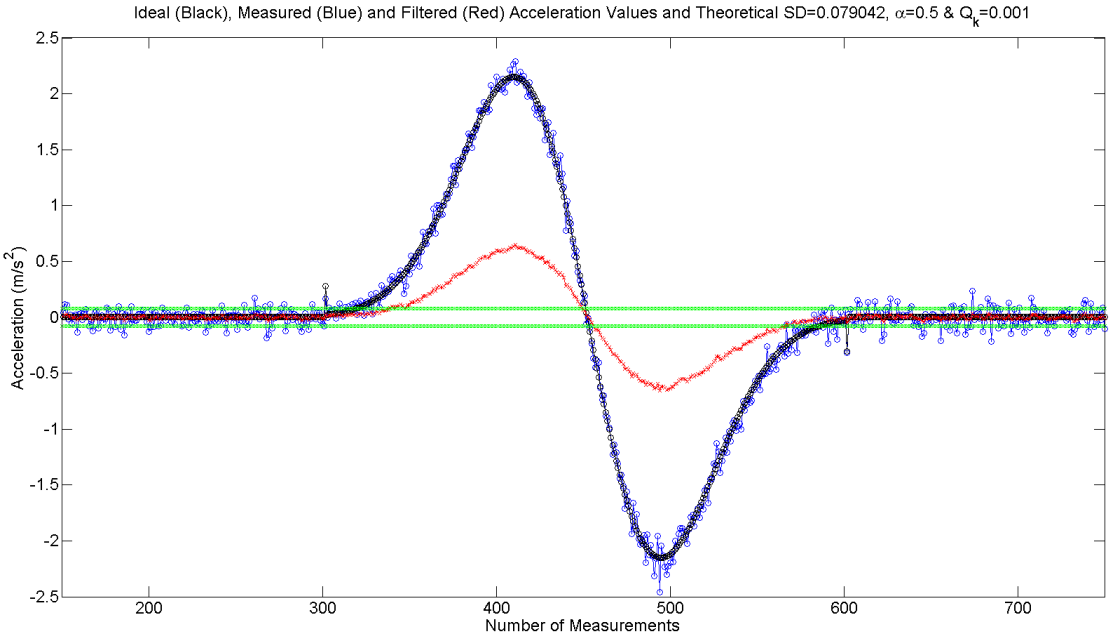

Fig. 6.18 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.0790, α = 0.5, and Q_k = 0.001

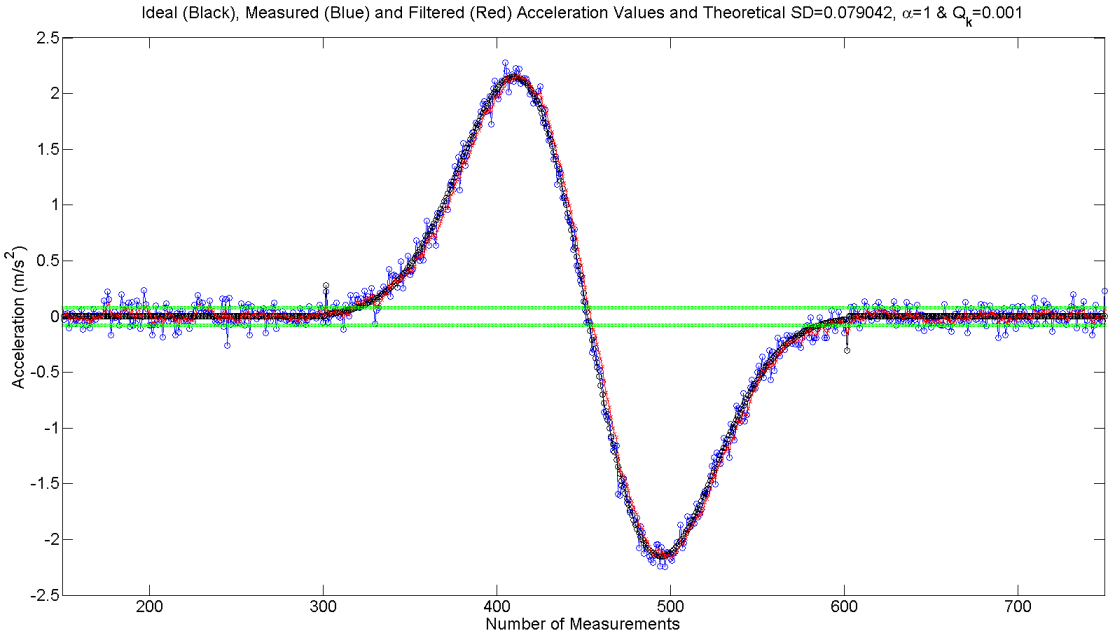

Fig. 6.19 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.0790, α = 1, and Q_k = 0.001

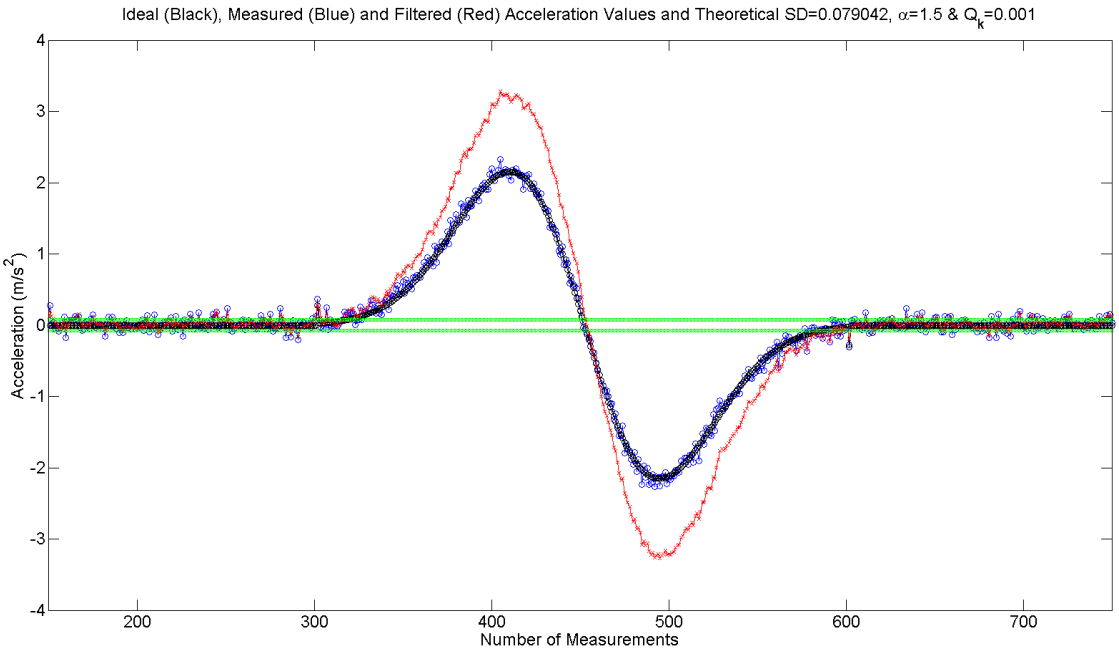

Fig. 6.20 Ideal (Black), Simulated (Blue), and Filtered (Red) acceleration values for dynamic case with R_k = 0.0790, α = 1.5, and Q_k = 0.001

As can be seen from Fig. 6.17 through Fig. 6.20, changes in α scale the filter estimate proportionately. When α is less than 1, the Kalman filter underestimates the samples while as when α is greater than 1, the filter overestimates the samples. In summary, whenever α is not 1, the filter performance degrades.

Thus, we choose α = 1 and Q_k = 0.001 as our parameters of choice to be used with sensor data. Appendix A.5 gives the source code used for analyzing Kalman filter's performance.

6.8 Removing Effects of Orientation

The motion-tracking algorithm proposed in this chapter is applicable for 1D and 2D motions with no orientation change. The Kalman filter developed earlier assumes that the accelerometer output changes purely because of linear motion. Nevertheless, in practice, it is extremely hard for us to perform any arm movement strictly in any one direction and more difficult to do so without any change in orientation. Thus, this section develops a method, which can eliminate orientation changes observed in human arm motions.

Figure 6.21 shows the sensor output for a reaching movement and its frequency spectrum. The activity performed for this trial involved stretching out the arm in X-axis of the sensor and then pulling it back in after some time. In Fig. 6.21, we can clearly see three different signal levels when there is no activity. The levels are marked red in top trace of Fig. 6.21. These changes in no activity signal output is an effect of changes in orientation. The bottom trace of Fig. 6.21 depicts the frequency spectrum of sensor output. From the frequency spectrum, it is clear that the most dominant component in the frequency spectrum is not present at 0 Hz but at a frequency marginally higher. This shift in dominant component of frequency spectrum is again because of orientation change.

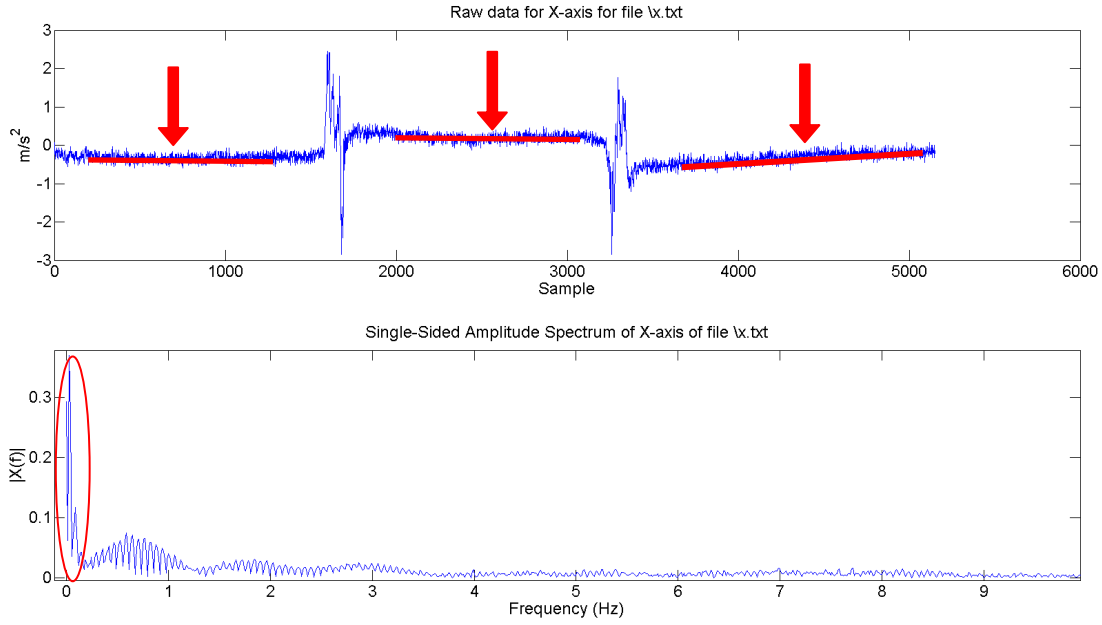

Fig. 6.21 (Top) Sensor output for a reaching movement, (Bottom) Frequency spectrum

Thus, to remove effects of orientation change, the low frequency components around 0 Hz must be filtered out. The Atomic 6DOF sensor samples at 100 Hz allowing us to observe any frequency from 0 Hz to 50 Hz. However, human arm motions are limited to about 10 Hz for athletic performance and about 5 Hz for ADL performance. Thus, by removing all frequency

components above 5 Hz we will minimize the measurement noise that is observed at higher frequencies. These requirements led us to develop a band pass filter.

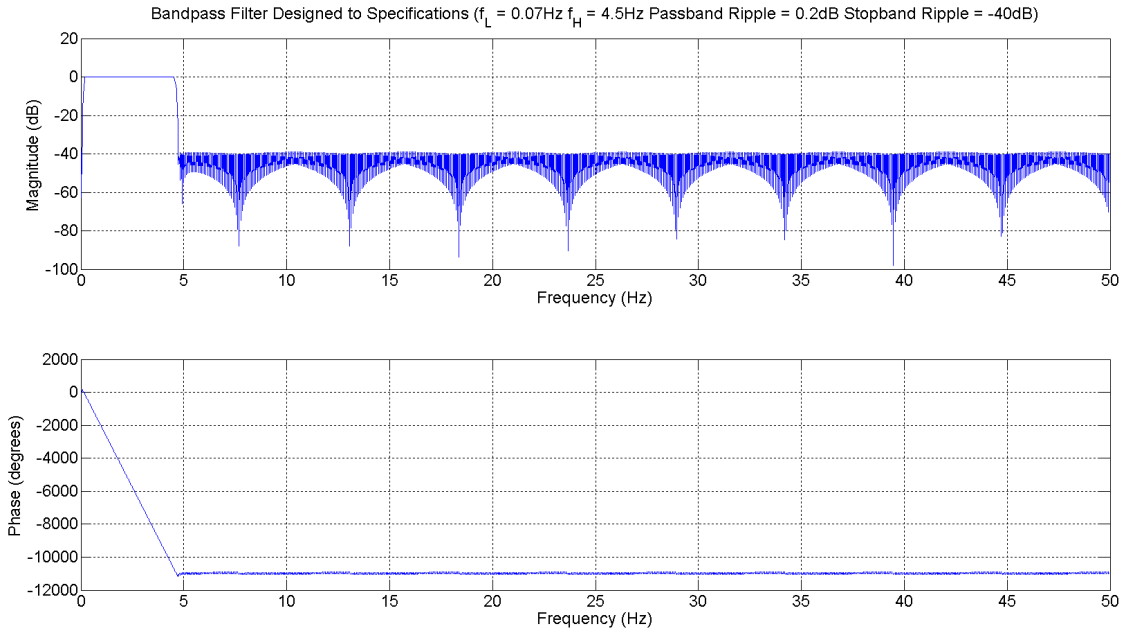

Fig. 6.22 (Top) Frequency response of the band passes filter, (Bottom) Phase response of the band pass filter

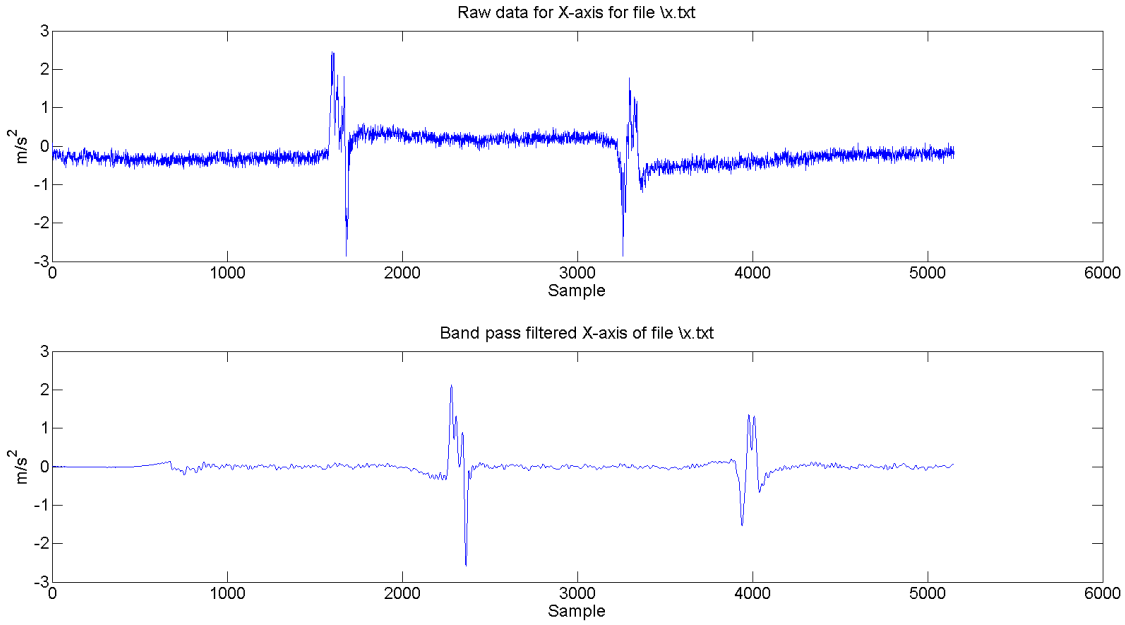

Fig. 6.23 (Top) Sensor output for a reaching movement, (Bottom) Band pass filtered sensor output

Top of Fig. 6.22 gives the frequency response of the band pass filter designed for filtering orientation changes while as the bottom section displays the phase response of the same band

pass filter. Figure 6.23 shows the raw sensor output and band pass filtered sensor output. As can be seen from Fig. 6.23, the raw sensor output and band pass filter outputs are not aligned in time domain. This is so because the band pass filter designed has a processing delay of about 900 samples or so. However, it is evident that the band pass filter does perform as per our expectations and we see all no activity signal levels being the same.

The development of band pass filter in this section has enabled us to transform a motion corrupted by orientation changes to a motion without any orientation changes. This will enable us to apply our algorithm with actual sensor output irrespective of whether orientation change, voluntary or involuntary, was present in the data or not.

6.9 Sensor Trials

After establishing the validity and usefulness of the motion-tracking algorithm through simulations, we will start using this algorithm with Atomic 6DOF data and analyze its performance.

6.9.1 1D Trials

As we started developing this algorithm for 1D case, we will use it for 1D motion tracking first. The 1D motion of stretching out the arm is sampled by Atomic 6DOF. The motion is performed along the Y-axis of accelerometer with no intentional change in sensor's orientation. The actual distance traversed for this motion was 40cm.

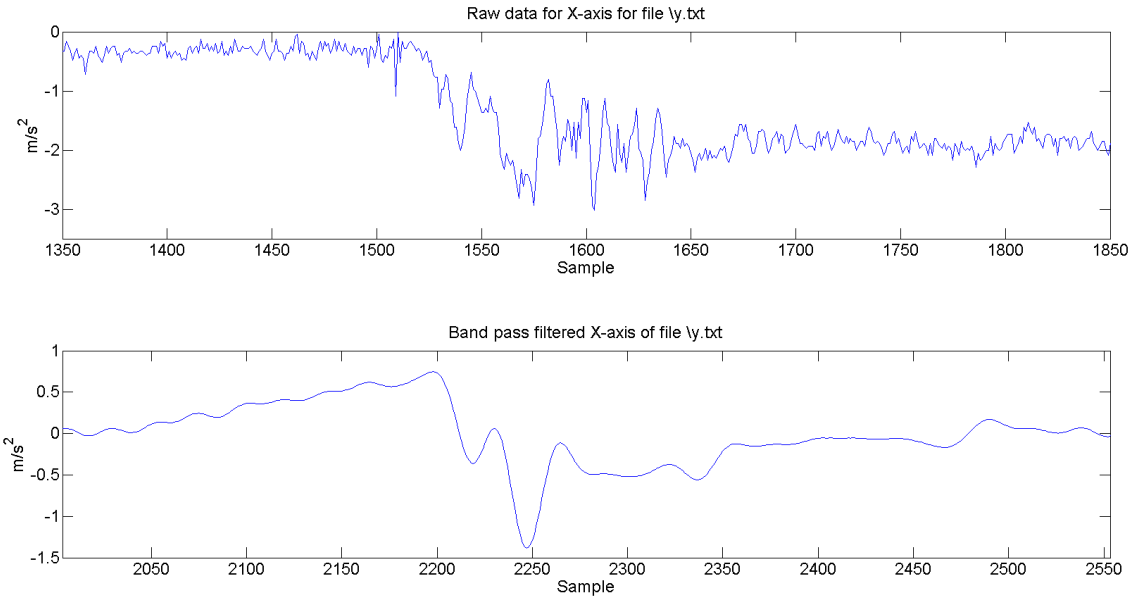

Fig. 6.24 (Top) Sensor output in X-axis for Y-axis reaching movement, (Bottom) Band pass filtered sensor output of X-axis

Figures 6.24 and 6.25 show the raw sensor output in both X and Y-axis in the top section and the output of band pass filter for these axes in bottom section. It is clear from Fig. 6.24 and Fig. 6.25 that the sensor output is corrupted by measurement noise but band pass filter smoothens the sensor output and we see the signal representing activity in Y-axis. The other interesting observation is that although the activity was performed along Y-axis, X-axis showed significant signal variation. This suggests that there is some other source of error that has not been modeled yet that affects the non-dominant axis of motion. This error source is called as cross-axis sensitivity of the accelerometer and arises primarily due to inaccuracies of manufacturing process. Section 6.10 discusses cross-axis sensitivity in detail. Nevertheless, as we know this activity was done along Y-axis we can completely ignore X-axis data and apply 1D motion-tracking algorithm.

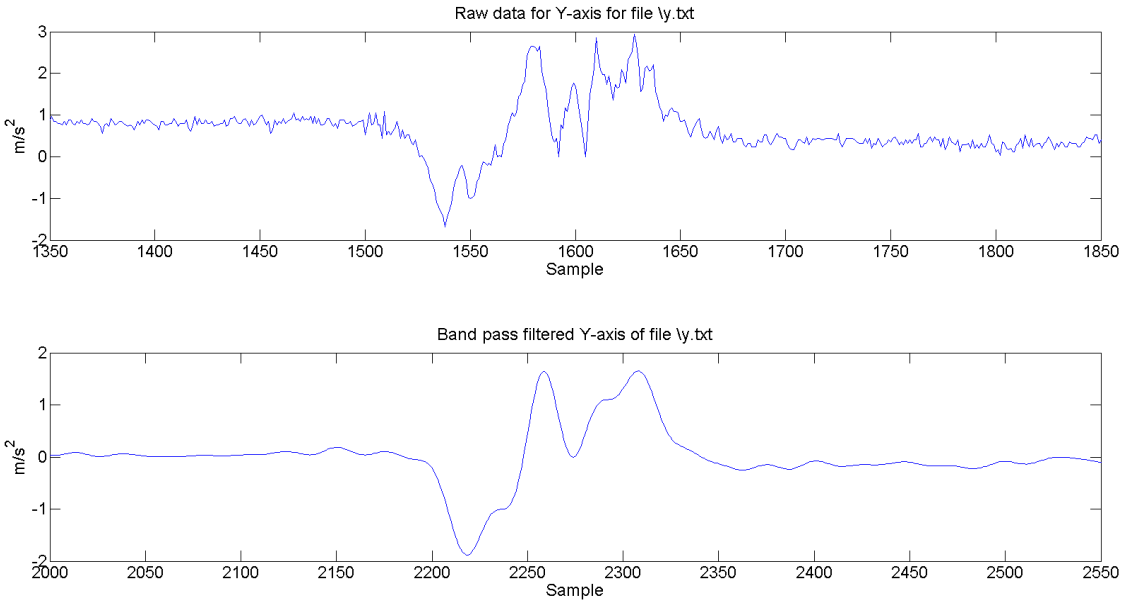

Fig. 6.25 (Top) Sensor output in Y-axis for Y-axis reaching movement, (Bottom) Band pass filtered sensor output of Y-axis

The output of 1D motion-tracking algorithm is shown in Fig. 6.26. The algorithm estimates a distance of about 32 cm for an actual distance of 40 cm, giving an error of 20%. To understand the error sources affecting the algorithm's performance we enable 2D motion tracking.

Figure 6.27 shows the output of the algorithm when 2D tracking is enabled. There are two observations to be made from Fig. 6.27. First, although the motion performed was purely 1D along Y-axis, X-axis also got affected. This resulted in some motion along X-axis in algorithm's output. However, this error in X-axis will not affect distance tracked along Y-axis and will be neglected for this section. Secondly, the underlying motion for Fig. 6.24 and Fig. 6.25 was stretching out the arm. This motion was monotonous in the direction along Y-axis without any change in direction. However, the algorithm output suggests that there was reversal in direction of motion. On close analysis of data, we concluded that the bias present in band-pass filtered data caused this direction change. We expect to see our data without any bias after band-pass filtering. However, as described in Chapter 2, some very low frequency phenomena affect the bias of any

MEMS device. In addition, the band-pass filter that we use has a high pass transition band from 0.07 Hz to 0.2 Hz. All frequencies from 0.2 Hz onwards until 4.5 Hz are passed as is by the filter.

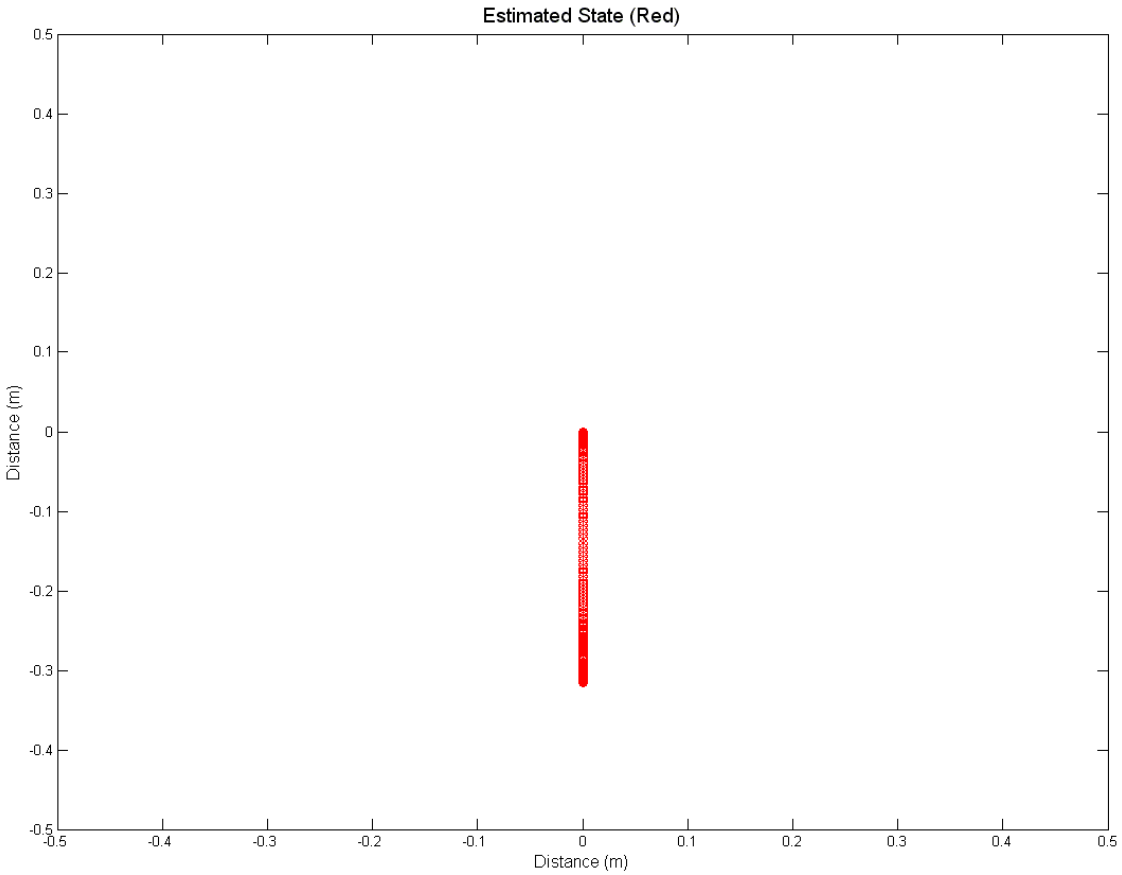

Fig. 6.26 Algorithm output when only Y-axis motion is tracked by ignoring X-axis

To maintain information related to actual activity we have to keep the pass-band corner frequency so low. Thus, to overcome this issue and improve the performance of our algorithm, we need to get rid of this bias.

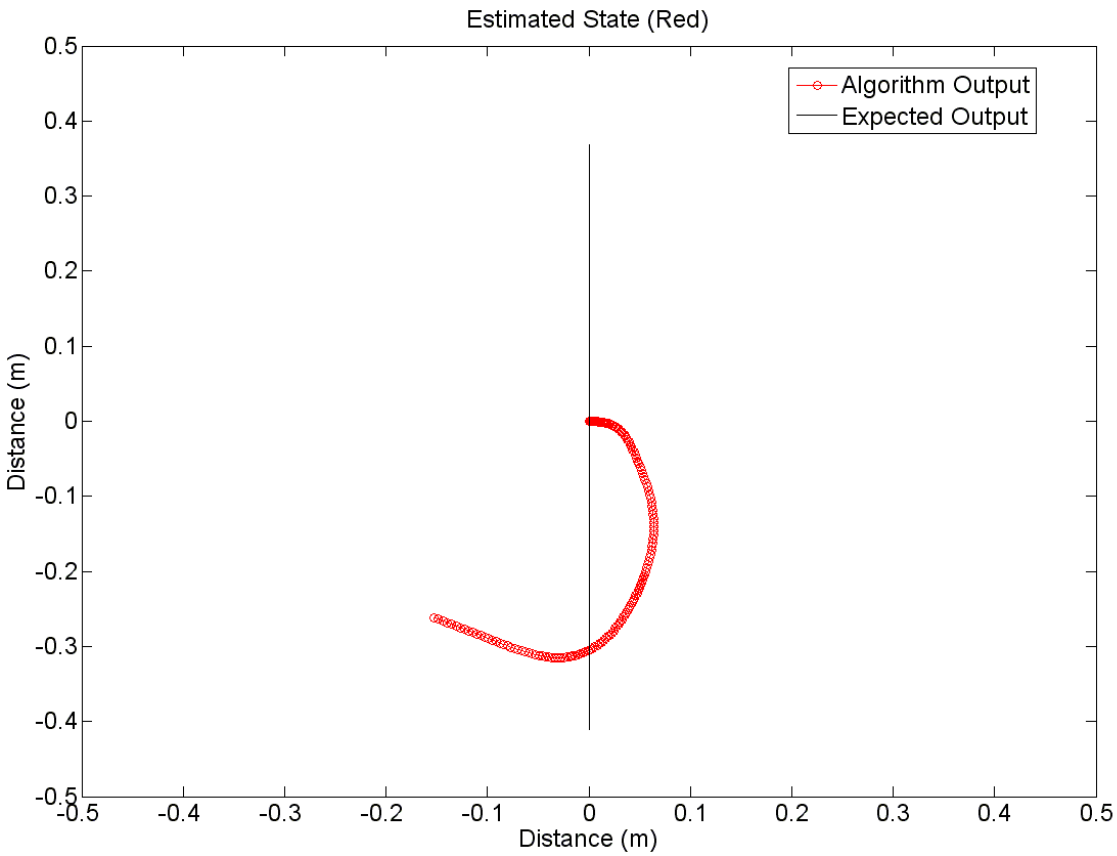

Fig. 6.27 Algorithm output when motion in both X and Y-axis is tracked

6.9.2 Bias Correction

To correct the bias that limits algorithm's performance, we need to find out the average of the data and subtract it from the data. This method is applied to both axes and the algorithm processes this bias corrected data. Figure 6.28 shows the algorithm's output after bias correction is applied.

After bias correction, the algorithm gives out a distance of 43 cm along Y-axis effectively reducing the error from 20% to 7.5%. The remainder error of 7.5% is dependent on the duration for which we run the algorithm. This error is systematic because of the way this algorithm is implemented. When there is no motion, theoretically the sensor output will always be zero.

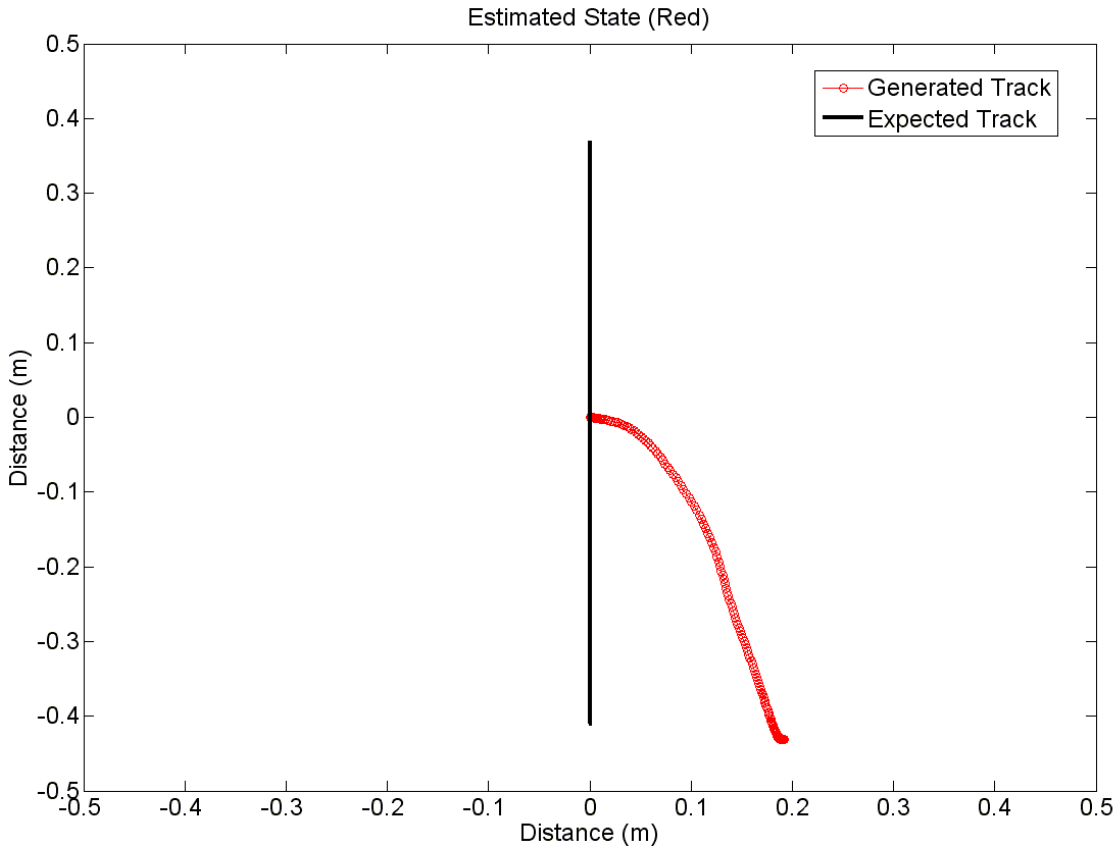

Fig. 6.28 Algorithm output when motion in both X and Y-axis is tracked after applying bias correction

However, we do see some measurement noise around zero in sensor output corresponding to no activity duration. In addition, with $Q_k \neq 0$, Kalman filter cannot remove this noise completely and hence we get non-zero sensor output for no activity duration. This non-zero acceleration accumulates over time introducing error in our distance estimates. Constraining the duration over which acceleration is integrated will result in additional performance improvement for the algorithm.

6.9.3 2D Trials

To analyze algorithm's performance, three categories of 2D motions are performed multiple times. Although, we present one typical case from each category, the analysis is consistently valid for all the tests in each category.

6.9.3.1 Diagonal Track

The diagonal track motion is achieved by performing a 1D motion along a line making an angle of about 45° each with negative X-axis and positive Y-axis. The distance traversed along the straight line was 60 cm. Thus, the distance traversed along X and Y-axis of Atomic 6DOF will be about 42 cm.

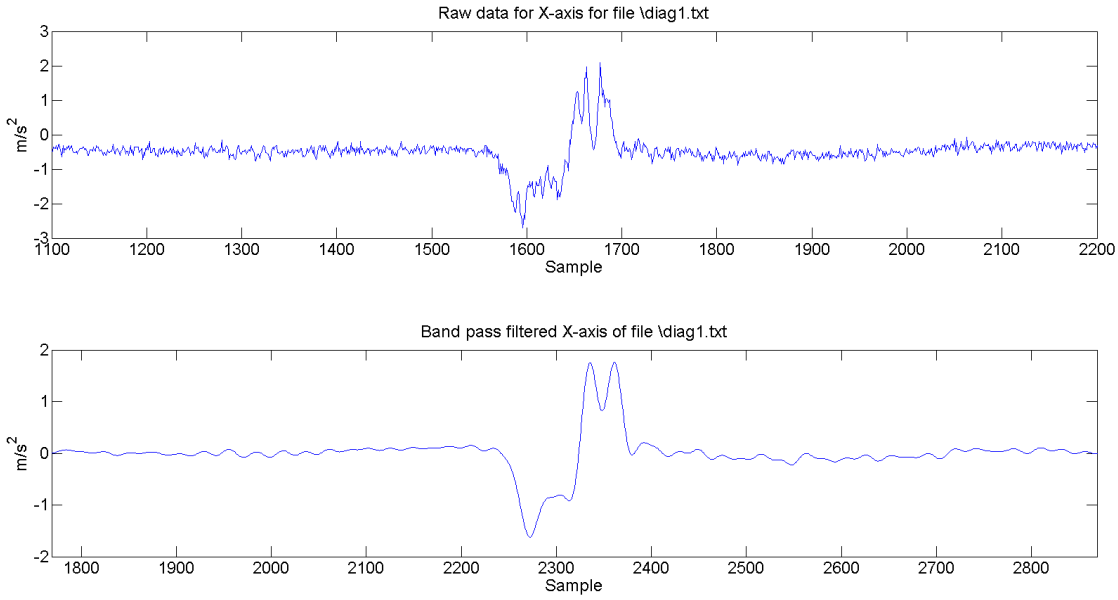

Fig. 6.29 (Top) Sensor output in X-axis for diagonal track, (Bottom) Band pass filtered sensor output of X-axis

Figures 6.29 and 6.30 show the sensor output in X and Y-axis for diagonal motion as well as the output of band-pass filter. Figure 6.31 shows the output of the algorithm with 2D motion tracking enabled. The final distance tracked by the algorithm is 44cm in X-axis and 47cm in Y-axis. The maximum error in this case is just above 10%.

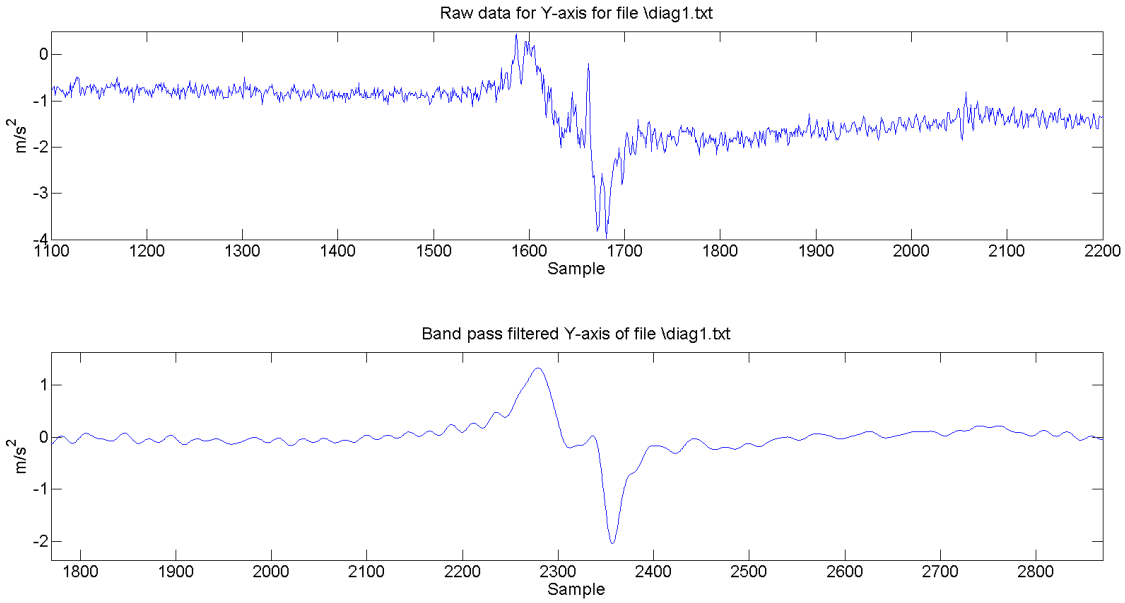

Fig. 6.30 (Top) Sensor output in Y-axis for diagonal track, (Bottom) Band pass filtered sensor output of Y-axis

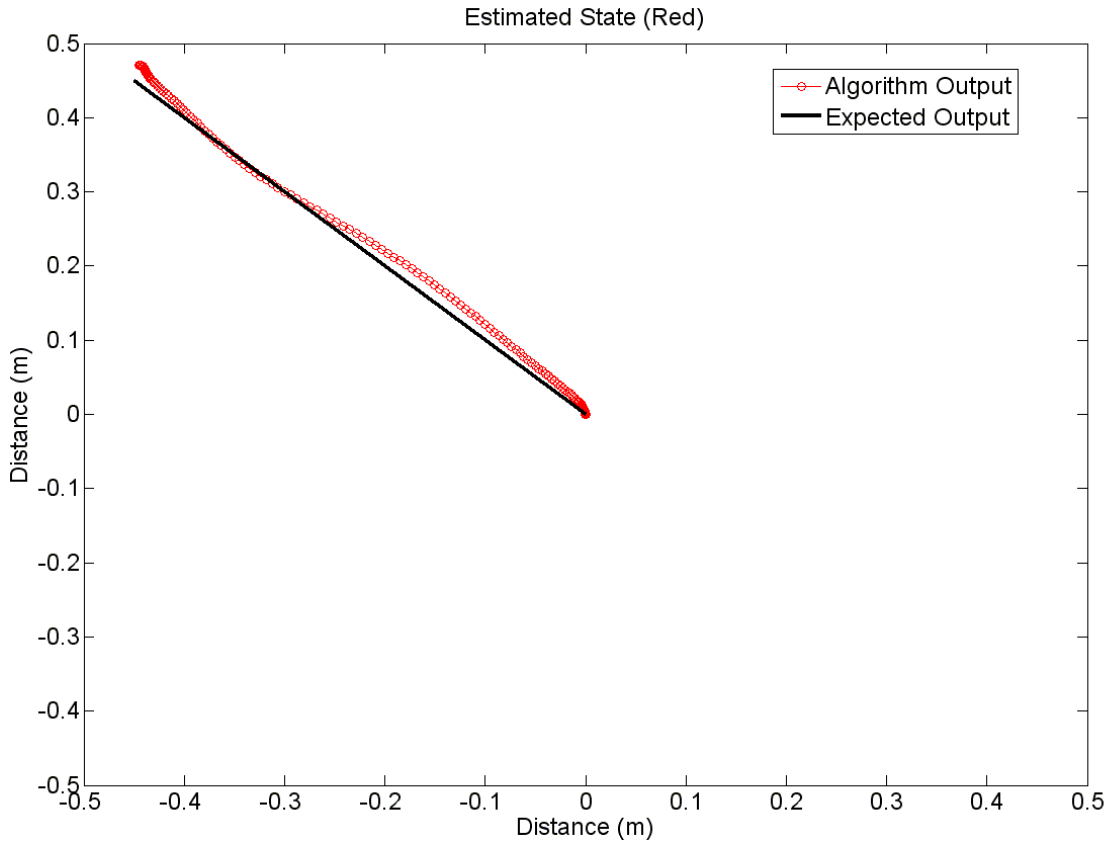

Fig. 6.31 Algorithm output for tracking diagonal motion

6.9.3.2 L-shaped Track

In 1D trial section, we observed that when the motion was performed along Y-axis, it affected the X-axis as well. To see if this effect also exists from X-axis to Y-axis we designed an

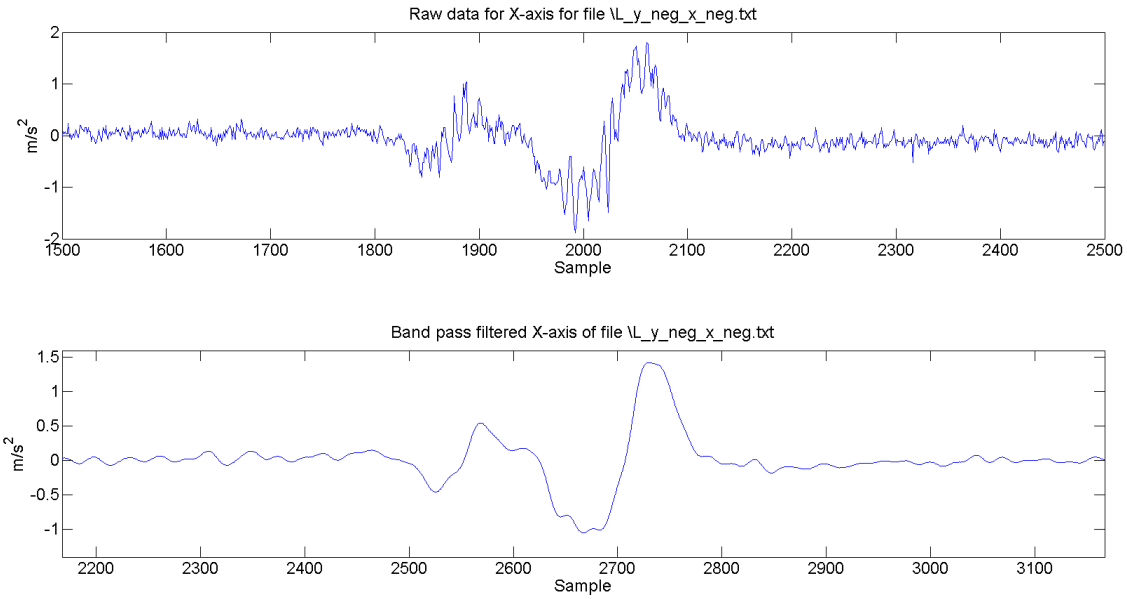

Fig. 6.32 (Top) Sensor output in X-axis for L-shaped track, (Bottom) Band pass filtered sensor output of X-axis

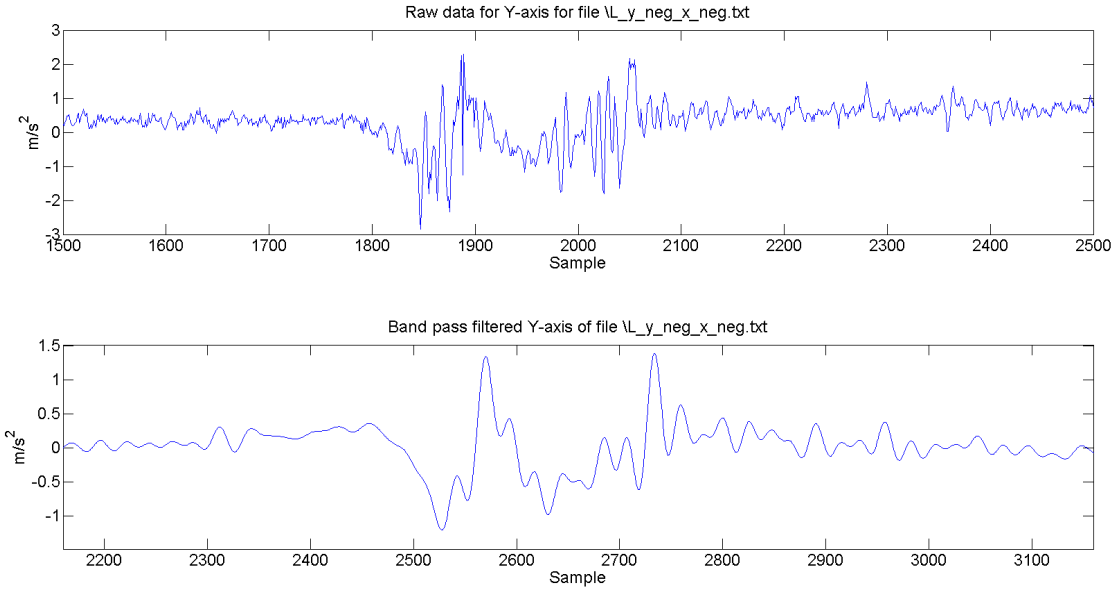

Fig. 6.33 (Top) Sensor output in Y-axis for L-shaped track, (Bottom) Band pass filtered sensor output of Y-axis

L-shape track. In addition, this track enables us to see how our algorithm responds to change in direction of motion.

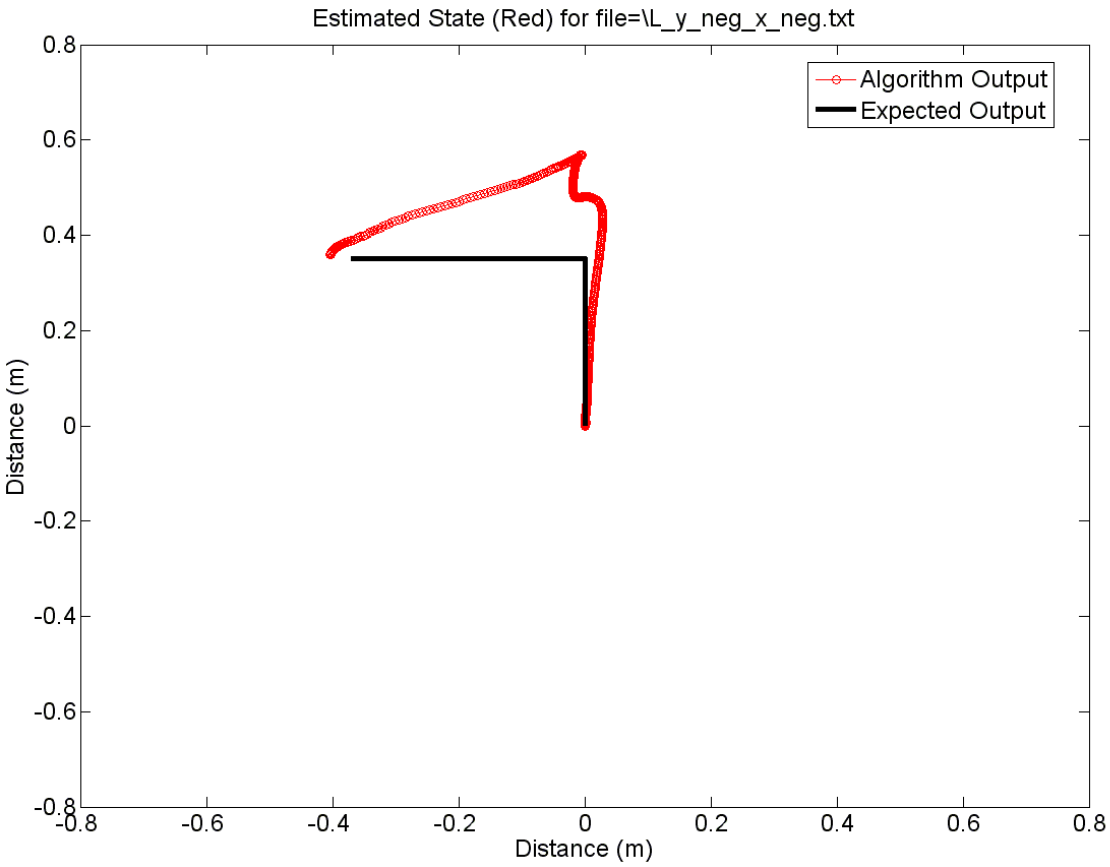

Fig. 6.34 Algorithm output for L-shaped track

The sensor output in X and Y-axis and the response of band-pass filter are shown in Fig. 6.32 and Fig. 6.33. Figure 6.34 shows the track estimated by the algorithm. From Fig. 6.34 it is clear that cross-axis effect is applicable to both axes. Considering the final position tracked by the algorithm, the error is once again about 10% or so. However, if we consider the shape of track then we have significant error along Y-axis, as maximum distance along both axes in this trial was 40cm. The cross-axis effect degrades the performance significantly. The X-axis motion starts after sample 2600 in band-pass filter output. After that sample, the Y-axis should have been stationary with just measurement noise. However, from Fig. 6.33 it is clear that Y-axis signal varies significantly and thus it keeps on accumulating more distance than actually travelled. Even

with this cross-axis effect the algorithm is able to respond to change in direction of motion and gives us a track that is a good approximate of the original track.

6.9.3.3 Rectangular Track

The primary objective for testing our algorithm with rectangular track is to observe the effect of integration interval on final output. The rectangular track presented in this section is obtained by tracing the perimeter of an A4 size (21 cm X 30 cm) paper.

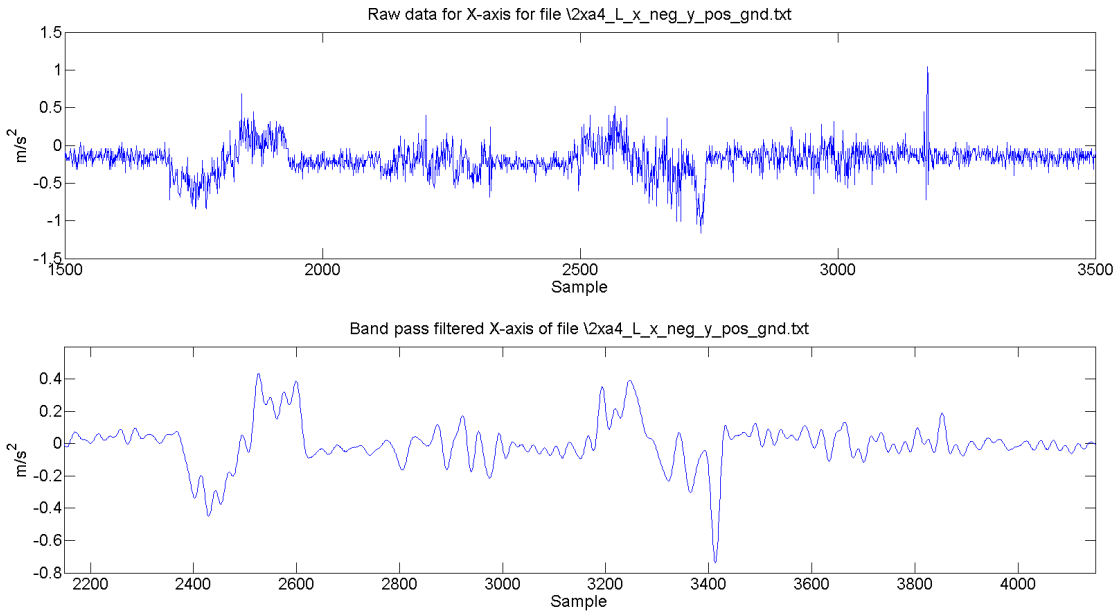

Fig. 6.35 (Top) Sensor output in X-axis for rectangular track, (Bottom) Band pass filtered sensor output of X-axis

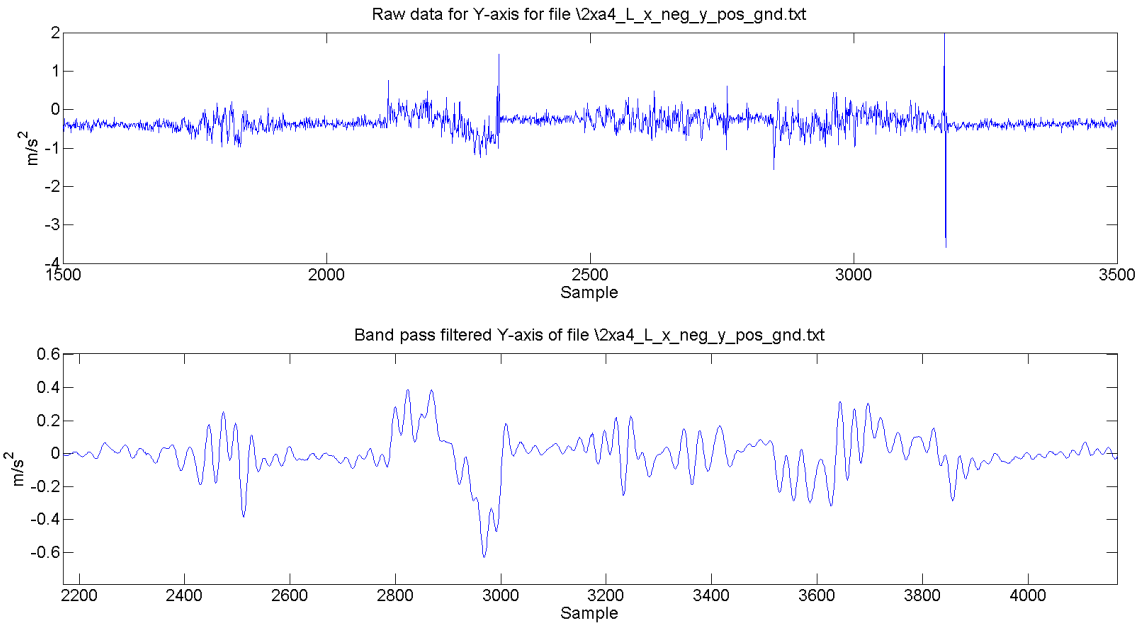

Fig. 6.36 (Top) Sensor output in Y-axis for rectangular track, (Bottom) Band pass filtered sensor output of Y-axis

Figures 6.35 and 6.36 show the sensor output and band-pass filter output for X and Y-axis in the case of rectangular track motion. Even in this case, we observe cross-axis effect in both of the axes. The long integration duration combined with cross-axis effect breaks the algorithm for rectangular tracking purposes. The algorithm's output track does not match the actual track either in shape or in distance.

To substantiate our claim that the primary source of error is the cross-axis effect, we disabled 2D tracking and tracked each side of rectangle as an individual 1D motion.

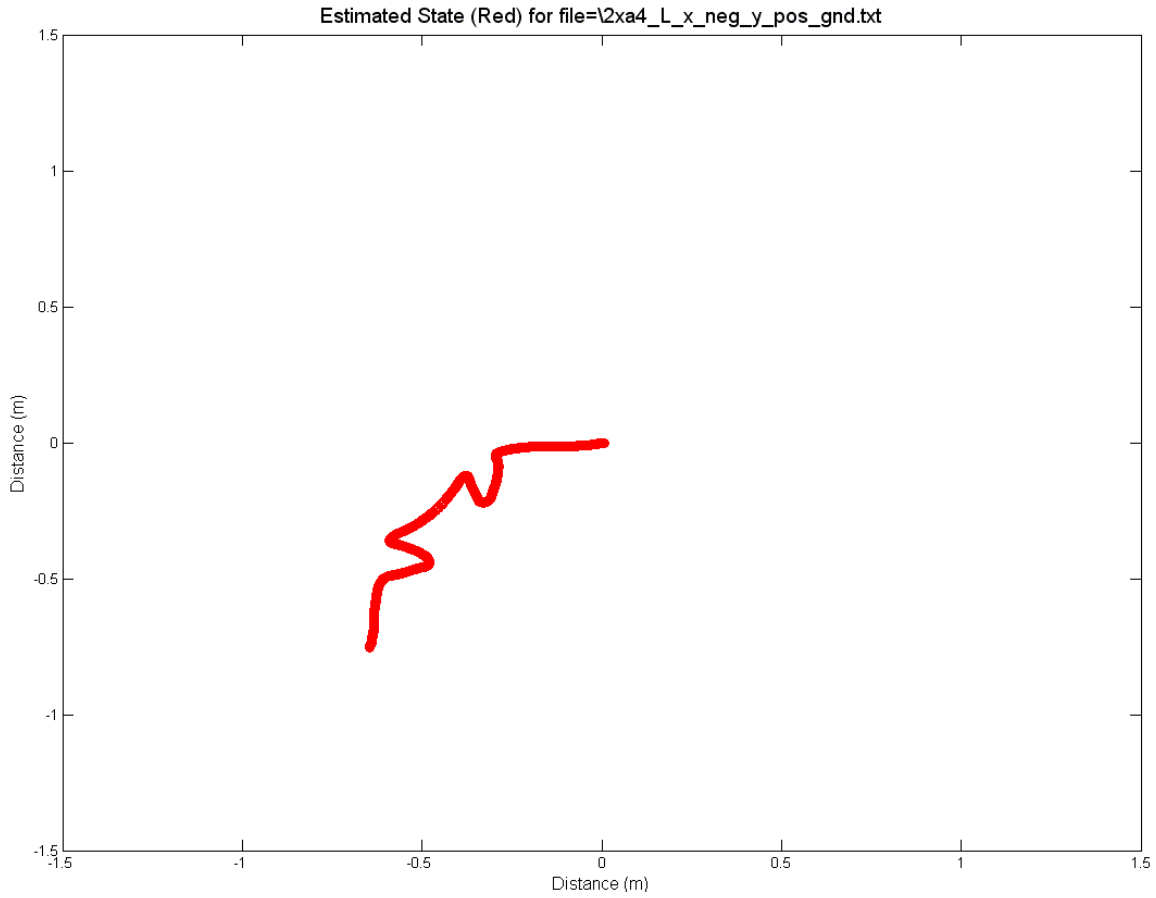

Fig. 6.37 Algorithm output for rectangular track

Figure 6.38 shows the output of these four 1D motions representing each side of the rectangle. For each segment, we restart the algorithm and that is why we see all of those four tracks starting from origin. The final distances tracked for each segment are in agreement with expected values with about 12% error in the output. Thus, this validates our claim that cross-axis effects degrade the algorithm performance, eventually making it ineffective for 2D motion tracking purposes.

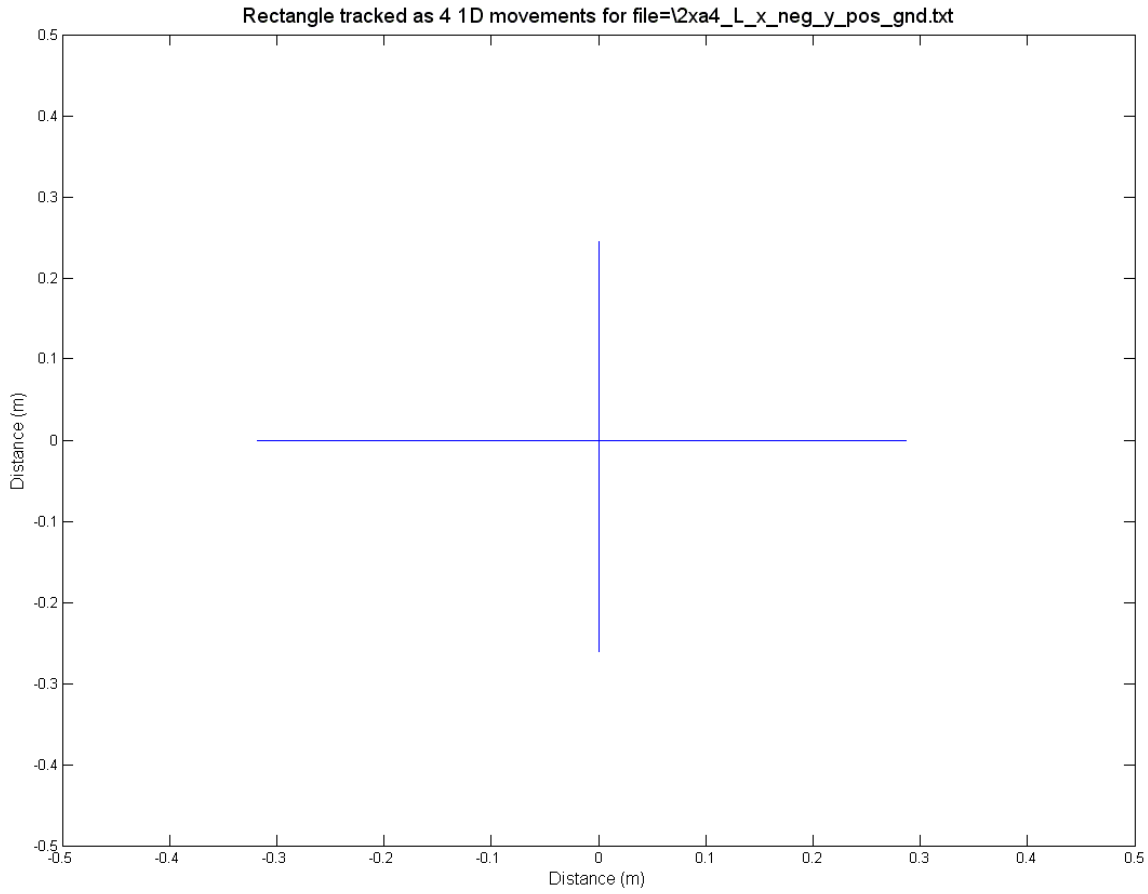

Fig. 6.38 Algorithm output in the case when each side of rectangle is tracked as 1D motion

6.10 Cross-axis Sensitivity

The cross-axis effect, which we have observed in earlier section, is characteristics of MEMS accelerometers. For MEMS accelerometers, the output created by forces in orthogonal axis is not equal to zero [22]. This phenomenon is called as *cross coupling*, which is measured by transverse sensitivity. Transverse sensitivity is defined as the ratio of the output caused by acceleration orthogonal to the primary sensitivity axis divided by the basic sensitivity in the primary direction. Cross-axis sensitivity is primarily caused by two factors –

- Inherent microstructure
- Inaccuracies in fabrication process, package orientation, and misalignment

End user cannot correct any of the above factors and only manufacturer can control these error sources. An end user can only model these error sources and minimize their impact on outcome.

Cross-axis sensitivity is critical for all applications involving linear motion sensing using accelerometers. As the majority of the work done on accelerometers relates to sensing and quantifying rotational motion, very few people have studied cross-axis sensitivity. To get an insight into cross-axis sensitivity, we present our preliminary analysis based on our observations from Atomic 6DOF trials.

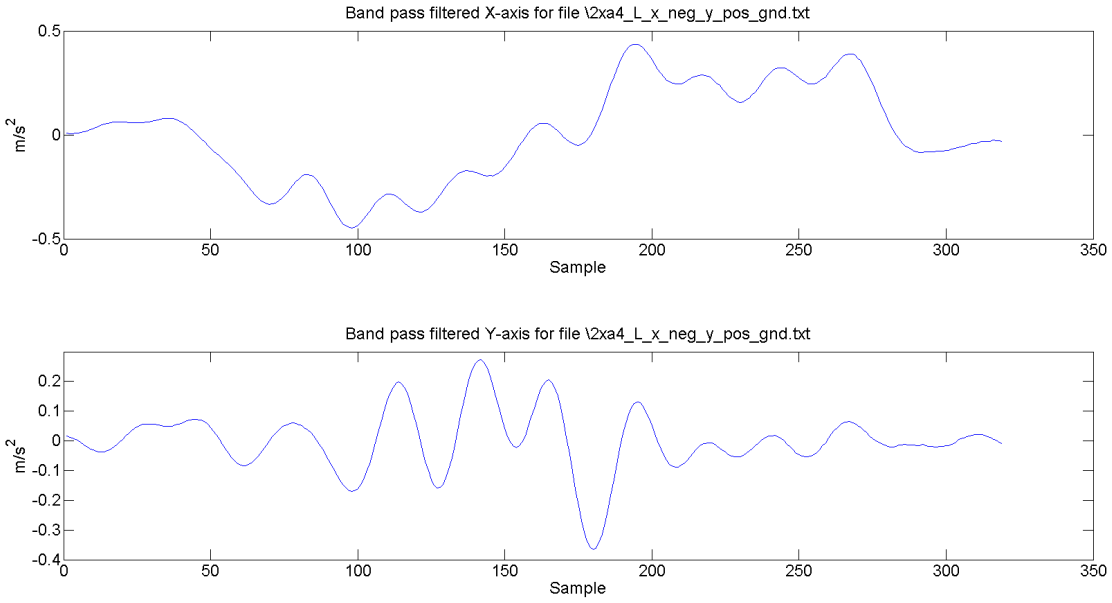

Fig. 6.39 (Top) Band-pass filtered X-axis for first segment of rectangular track, (Bottom) Band-pass filtered Y-axis for first segment of rectangular track

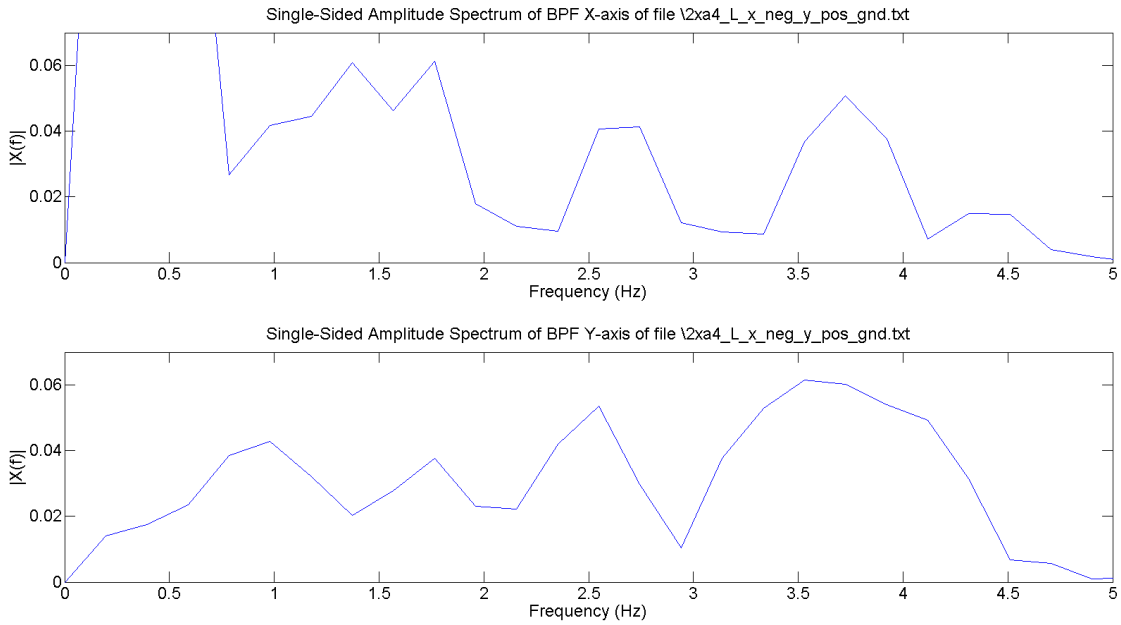

Fig. 6.40 (Top) FFT of X-axis for first segment of rectangular track, (Bottom) FFT of Y-axis for first segment of rectangular track

Figure 6.39 shows the band-pass filtered X and Y-axis representing the first segment of rectangular track when X-axis is the primary axis of motion. From Fig. 6.39 we observe that high frequency 'sinusoidal ringing' present in X-axis gets translated to an equivalent 'sinusoidal ringing' in Y-axis. The ringing in Y-axis is centered on zero while as the ringing in X-axis is overriding a slow frequency component. Figure 6.40 shows the FFTs of both X and Y-axis data shown in Fig. 6.39. Figure 6.40 reinforces our observations. The frequency components in Y-axis match exactly to those seen in X-axis with exception of most dominant low frequency component. From Fig. 6.39 and Fig. 6.40 we conclude that cross-axis sensitivity effects align in both time and frequency domain.

Still, we will need more information regarding cross-axis sensitivity so that we can model and correct it. We will need to investigate the relationship of frequencies with cross-axis sensitivity as well as the effect of amplitude of primary axis signal, among other things. In addition, we need to investigate how cross-axis sensitivity will change when the force applied is oblique to each axis. This information will enable us to evolve our algorithm to track 'true' 2D motions.

6.11 Tracking 2D Motions as Dependent 1D Motions

The results presented so far emphasize the usefulness of the motion-tracking algorithm developed in this chapter for 1D motion. Although, this algorithm has been tracking motion, it has done so with a consistent error of about 10% or so. However, as discussed earlier majority of this error is systematic in the sense that it depends on the duration for which the accelerometer signals is integrated. Thus, if we limit this integration duration exactly to motion duration, we will track the motion more accurately. In addition, we now understand that this algorithm falls short to track motion in 2D, which is when more than one axis is associated with a motion. Thus, if 2D motions are performed in a way such that at any point of time only one axis will have the motion associated with it, we can apply this algorithm to those individual motion durations and piece them together to determine the final track. In this section, we present a method, which identifies those individual motion durations to generate the final track.

The method takes band-pass filtered sensor signals as its input. First, this signal is centered on zero by subtracting the average value from each sample. Then, we smooth this signal using moving average filtering method. Parameter *half_win* determines the window used for moving average filtering purpose. Similar to the algorithm presented in Chapter 4, based on the value of *thres* we mark possible activity candidates. All sample values greater (smaller) than +*thres* (-*thres*) are marked +1 (-1).

Figure 6.41 shows the DC balanced and moving average filtered output of band-pass filter data. The red trace shows the possible activity durations. The selection of *thres* is a tradeoff between removing all noise and finding all activity samples. The primary purpose of smoothing band-pass filtered data is to get rid of local maxima caused by the ‘sinusoidal ringing’ due to cross-axis sensitivity. To smooth this ringing effectively, the window used for moving average should be wide enough to minimize this ringing.

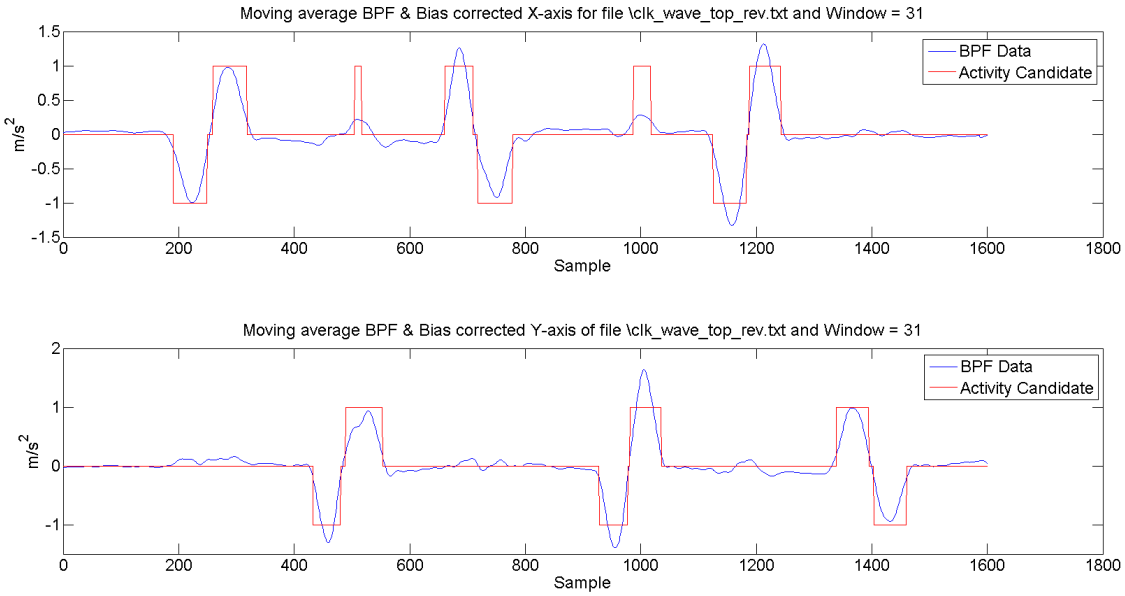

Fig. 6.41 Zero Centered moving average of Band-pass filter output (blue) with initial activity candidates (red) for X-axis (Top) and Y-axis (Bottom)

As it is extremely hard to determine accurate value of *thres*, we will have some spurious noise spikes marked as possible activity candidates. An activity usually lasts sufficiently long enough so that we expect to see a minimum number of samples above *thres*. If any of the possible activity candidates have less than *short* number of samples above *thres*, then that is highly likely to be a spurious activity candidate due to noise. We remove such spurious candidates by checking their time duration.

Figure 6.42 shows the output obtained after removing spurious activity candidates. All the activity candidates that remain are wide enough to be a valid activity. However, some of these candidates have signal changes just because of cross-axis sensitivity. Presence of cross-axis sensitivity is easy to determine. If we have activity candidate overlapping in two axes then we know it is a case of cross-axis sensitivity. Within this overlapping region, the axis with the highest signal is termed as primary axis while as the other axis is ignored completely.

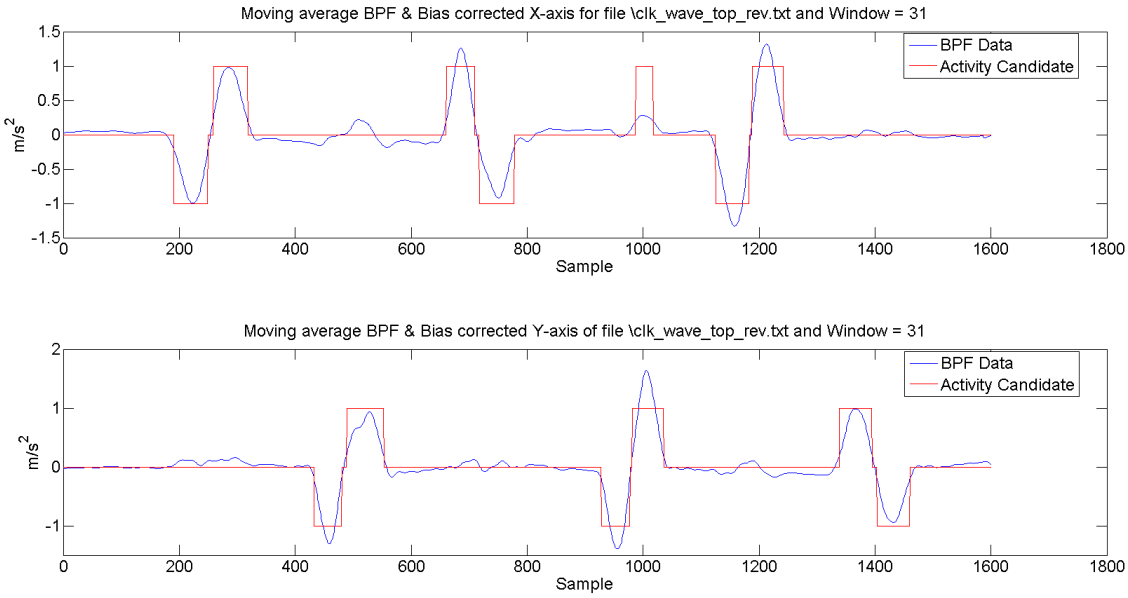

Fig. 6.42 Zero Centered moving average of Band-pass filter output (blue) with activity candidates after removing spurious candidates (red) for X-axis (Top) and Y-axis (Bottom)

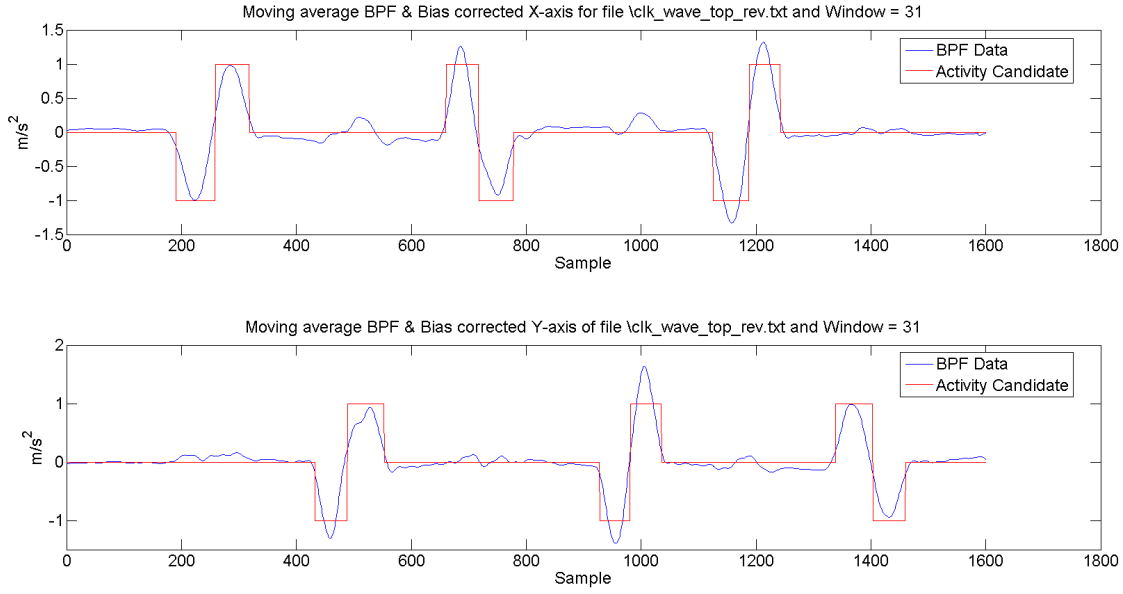

Fig. 6.43 Identified activity durations (Red) with zero centered moving average band-pass filter output (Blue) for X-axis (Top) and Y-axis (Bottom)

Figure 6.43 shows the activity candidates remaining after taking into account cross-axis sensitivity effects. These activity candidates are confirmed as valid activities. To determine the duration of the entire activity, it is important to combine the positive and negative peak durations of each activity and consolidate them as a single activity. After finding all such continuous activity durations, we apply the 1D motion-tracking algorithm for each activity duration. Between

activities, we ignore all sensor data and assume there is no movement. We store the distances tracked during each activity and accumulate all of them together to generate the final track. Figure 6.44 shows the band-pass filter output with activity durations marked on it for both X and Y-axis. To track the actual motion, band-pass filter output is used as input for the motion-tracking algorithm rather than zero centered moving average filtered data.

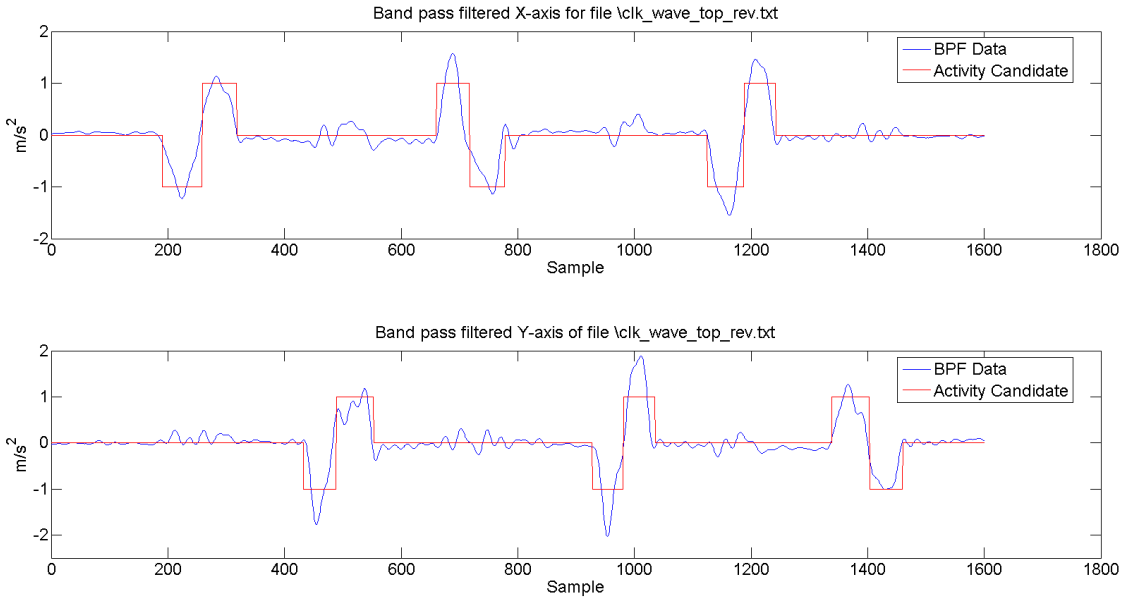

Fig. 6.44 Identified activity durations (Red) with band-pass filter output (Blue) for X-axis (Top) and Y-axis (Bottom)

Figure 6.45 shows the final track generated by this method. Going by the shape of track this method generates a track that accurately matches the actual motion track. Comparing with the final distance achieved in actual motion to that generated by this method, the error in final distances is about 5%, which for all practical purposes is insignificant. If we compare total distance along all six 1D motions then the error reduces significantly.

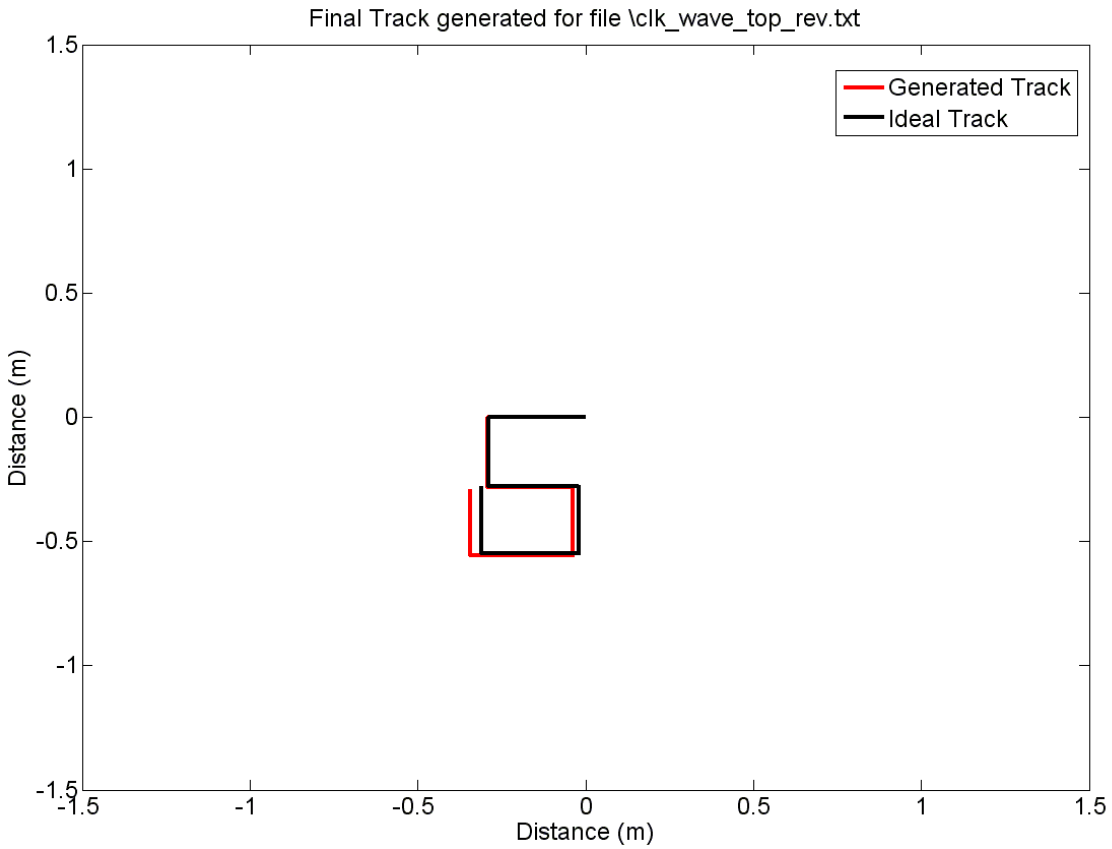

Fig. 6.45 Final track generated by tracking six 1D motions in 2D

Appendix A.6 has the source code for this method along with band-pass filter design and the motion-tracking algorithm. Appendix B.2 has supplementary results obtained from three more data files. From all of these figures, we conclude that the motion tracking algorithm developed in this chapter has very high accuracy and is robust for various range of motions.

6.12 Application

The motion-tracking algorithm presented in this chapter along with the method developed in Section 6.11 enable us to track any reaching movement. Two out of four activities from Chapter 5 belong to the category of reaching movements. Patients with severe stroke have minimal functional ability in their affected arms due to significant motor deficit. This reduced functional ability affects execution of ADLs and in turn patient's independence in daily life. Thus, stroke survivors need to regain these functional abilities to live the life the way they were living before stroke.

Figure 6.46 shows the video grab of patient trial for *activity 1* of Chapter 5. This activity consists of pulling in a gunny bag on a flat surface. The yellow arrow shows the direction of motion. Thus, this activity is effectively performed in 1D and thus the algorithm presented in this chapter will track this activity without any issues. It is worth pointing out from Fig. 6.46 that the patient does not have fine motor skills and hence not able to grab the gunny bag. In addition, the wearable IMU that is on patient's wrist can only sense gross movement and will not be affected by patient's fine motor skills.

Fig. 6.46 Video grab showing the activity of pulling in gunny bags ('*activity 1*' from Chapter 5)

Fig. 6.47 Video grab showing the activity of sliding out shower rings ('*activity 4*' from Chapter 5)

Similarly, Fig. 6.47 shows the video grab of a patient trial for '*activity 4*' of Chapter 5. Again, we observe that this activity is performed in 1D as well and hence the algorithm presented here can track this motion as well.

Thus, for patients with chronic stroke, gross motor skills are more critical (necessary) than the fine motor skills. This chapter provides a robust method that can track these gross motor skills (such as reaching movements) with high accuracy. The results generated by this algorithm will be a good source of feedback for both the patient and the therapist. Therapists can review the results generated by this method over a period to track the rehabilitation progress. The outcome from this algorithm is the actual distance traversed which is a true indicator of functional ability of the affected arm. Thus, even the patient himself can understand the results and try to improve his functional ability in affected arm.

6.13 Summary

We presented a motion-tracking algorithm employing basic Kalman filter. The accelerometer output is modeled as an AR process for Kalman filter design. Although, the filter was designed for 1D motions, we presented its extension for 2D motions with no orientation change. We developed a human arm motion simulator based on a simplified model of human arm motion. With help of this simulator, we analyzed Kalman filter performance and found optimal parameters that give accurate results.

Actual arm motion is complex and has unintended orientation changes. In addition, sensor calibration errors corrupt the signal representing the motion. Thus, we developed signal processing based methods such as band-pass filter, bias correction that would filter all these non-idealities from the sensor output.

The performance of these filtering methods and motion-tracking algorithm is evaluated using actual sensor data representing various forms of activities ranging from simple 1D to complex 2D motions. We verified that the motion-tracking algorithm performs well for 1D motion. The performance of the motion-tracking algorithm for 2D motion is limited by cross-axis

sensitivity of accelerometer. We presented a preliminary analysis of cross-axis sensitivity of our sensor and analyzed its effect on the algorithm.

Lastly, we presented a method that can identify the dominant axis for a motion under consideration at any point of time. This method assumes that at any point of time during a motion, only one axis will be dominant. With this assumption, the method breaks up a 2D motion into a set of 1D motion. With this approach, the performance of motion-tracking algorithm for such 2D activities improves dramatically.

Chapter 7

Summary and Future Work

With advances in manufacturing processes, performance of MIS has improved significantly. This has opened many opportunities in human motion tracking domain. Although, MIS have been used earlier for tracking orientation or linear motion or both, most of those efforts did not rely on MIS completely. They used other sensors along with MIS to improve performance of their systems. Those who used only MIS, were able to do so at the expense of high cost military grade sensors. The issue of cost and accuracy has prevented consumerization of MIS based systems. This thesis is a step towards developing a cheap and reliable MIS based technology and applying it in stroke rehabilitation. In this thesis, we develop three algorithms that will enable us to track the rehabilitation progress of stroke-affected arm. All of the three algorithms presented in this thesis are exclusively based on the off-the-shelf 3D IMUs, which makes it economical. Along with arm usage detection and counting number of activities performed, we have used IMU for developing an objective test that can not only track the rehabilitation progress but also compares and categorizes stroke-affected patients according to their functional ability in upper arm. The linear motion tracking algorithm presented in this thesis is novel as compared to earlier work, as no one has ever developed a human arm linear motion tracking solution exclusively based on IMUs.

7.1 Summary

CIMT is a standard stroke rehabilitation therapy that has been proven clinically effective. However, current method of administering CIMT relies on human observations and tracking therapy effectiveness is based on subjective and objective methods. Subjective methods are affected by manual observational errors while as the objective methods that are used currently

requires manual intervention and a special set up to carry out the test. These practices not only make current CIMT delivery inefficient but also are susceptible to human error. Thus, benefits of developing a technology that will perform these tasks automatically and objectively are invaluable. In this thesis we present three algorithms that improve the efficiency of CIMT delivery and in some cases impart the much needed accuracy to some parts of CIMT by getting rid of observational errors.

The basic principle underlying CIMT is – by forcing use of affected arm for ADLs, functional ability in the affected arm will recovered. This requires monitoring affected arm use throughout the day to analyze and track the rehabilitation progress. In Chapter 4, we presented a method that exactly addresses this issue. Along with activity detection, the method counts the number of activities performed if the same activity is repeated multiple times. Although, highly complex and accurate methods to detect arm movements have been developed earlier, the method presented in Chapter 4 is computationally simple as well as accurate, which makes it easy to apply this method by embedding it on the wearable sensor board. The simplicity of this method makes it easy to run on a system with mediocre microcontroller. Lack of a need to have dedicated digital signal processing microprocessor translates into cost saving. The method presented in Chapter 4 gives not only overall arm usage but also the arm usage along each axis of the sensor. As the sensor's orientation is fixed with respect to arm, therapist can track rehabilitation progress along these individual axes and update the CIMT for better results.

Chapter 5 presents a new objective measure to track rehabilitation progress. This measure has better temporal resolution when compared against existing measures. The measure identifies changes in functional ability of the affected arm over a period of two days. In addition, the measure is effective in classifying functional abilities of stroke patients. The method employed to calculate this objective measure utilizes accelerometer signal to calculate histogram of distance traversed in specific time duration. The measure is suitable for patients with chronic stroke who are midway through their rehabilitation therapy, that is, patients with certain degree of functional

ability in their affected arm. The effectiveness of the measure is verified using data from patient trials.

One of the factors limiting effectiveness of CIMT and in effect lengthening the duration to complete recovery is the lack of ability to track rehabilitation progress when the patient is away from therapist. MIS based systems offer a compelling solution in this regard. The method presented in Chapter 5 provides this ability partially as it derives a quantity from the activity data and thus is an indirect indicator of the rehabilitation progress. However, the ability of tracking affected arm motion as it happens will be a direct measure of rehabilitation progress and will be more meaningful for the therapist. The algorithm presented in Chapter 6 does exactly the same, tracking affected arm motion as it happens. The algorithm utilizes Kalman filter for fusing data from multiple sensors. It uses digital signal processing techniques such as band-pass filtering, moving average filtering, and bias correction to correct for errors arising due to different error sources discussed in Chapter 2. The method works well for 1D and 2D motions under the assumption that there will be no orientation change during the motion and at any point of time, only one axis will be primary axis for the motion. The algorithm's performance for 2D motion tracking is limited primarily by cross-axis sensitivity of the accelerometer. A preliminary analysis of cross-axis sensitivity is presented. An extension of the algorithm to minimize the impact of cross-axis sensitivity follows this preliminary analysis. The performance of this algorithm is verified with both 1D and 2D motion trials.

7.2 Future Work

The work presented in this thesis gives an overview of some of the fundamental approaches that will lead to the development of a more comprehensive MIS based solution. This thesis has used the fraction of the potential that MIS based systems' have. The work presented in this thesis opens up a number of leads for research. These leads will eventually help us in realizing the complete potential of MIS based systems. This thesis dealt with accelerometer signals and ignored gyroscope signals. To realize the complete potential of MIS based systems, it

is imperative that not only accelerometer but also other available sensors' (such as gyroscope's and magnetometer's) signals should be used. This section discusses some of the future work that will enable us to realize the complete potential of MIS based systems.

The algorithm developed in Chapter 4 and Chapter 5 has been shown to be effective. However, to influence CIMT administration and improve CIMT delivery, it is important that these methods be deployed for remote monitoring. Combining these both algorithms into a comprehensive algorithm will result in a method that can detect affected arm use, count the number of activities being performed and track the rehabilitation progress by evaluating histogram of distance traversed during a particular time window.

A smartphone application might be the easiest and the quickest way we will be able to develop a complete remote monitoring system for tracking stroke rehabilitation of arm. With advances in wireless and cellular communications and emergence of smartphones, we have access to Internet connected portable computers. Current smartphones have enough computing power that running these algorithms on smartphones will be an easy task. A smartphone application not only will monitor rehabilitation progress and give instantaneous feedback to patient but also will keep a time record of all data sampled and send it to therapist for review and analysis purposes. This technology will also be beneficial in elderly monitoring systems.

The motion tracking algorithm present in Chapter 6 is currently limited to a limited set of 2D actions due to lack of a model representing cross-axis sensitivity of 3D accelerometers. For quicker deployment of motion tracking algorithm of Chapter 6, a network of multiple sensors along with minor tweaks to algorithms will be useful. This will be achieved by orienting multiple accelerometers in a way such that for any activity one of the axes of one of the accelerometers will always be dominant. Such an arrangement of multiple sensors will allow us to ignore cross-axis sensitivity and still be able to track complex activities.

An algorithm that tracks 3D motion completely with all orientation changes will solve almost all of the issues that limit CIMT's efficiency and efficacy currently. To achieve complete

3D motion tracking capability, the limitations of motion-tracking algorithm of Chapter 6 must be resolved. The motion-tracking algorithm's performance is limited by cross-axis sensitivity of accelerometer. This limitation arises because the process model that we use for Kalman filter or other DSP techniques does not take into account this error source. Thus, to enable complete 2D linear motion-tracking, it is necessary to analyze and model cross-axis sensitivity of the accelerometer. The algorithm assumes that there will not be any change in orientation. This assumption came into existence because there was a necessity to analyze accuracy of a motion tracking solution exclusively based on accelerometers. The next logical step to pursue will be to consider gyroscope signal and update the algorithm such that it will be able to track motion and orientation change. This can be done incrementally by considering one gyroscope at a time and progressing from complete 2D motion tracking to complete 3D motion tracking.

Although optical motion tracking methods are more accurate and stable than MIS based systems, line of sight operation and bulky equipments limit their applications. On the other hand, MIS based systems are completely portable. Actually, these systems are an excellent candidate for wearable technology.

Any future work that will be developed will need to be verified by performing clinical trials with stroke survivors. Some of the methods developed in this thesis are not applicable to patients who have had a severe stroke. All future methods need to validate their applicability to various classes of stroke survivors and a comprehensive method will be preferable over a method that is applicable to selective classes of stroke survivors. All future methods, either should not have any kind of interaction with the devices/setup used for monitoring the arm motion or should minimize this interaction to minimal amount. This will be encouraging for stroke survivors to embrace such a method. Stroke related disabilities are behavioral as much as they are physiological. Thus, future methods should give some feedback to stroke survivors regarding their arm movements. For effective use of technology in stroke rehabilitation, along with affected

arm movement we need to understand whether the movement is intentional or unintentional with specificity. Future systems might need to use an array of various sensors to achieve this objective.

Although, the work presented in this thesis has been developed for stroke rehabilitation, it will be similarly applicable for elderly monitoring, other disability rehabilitation, and recreation.

Thus, the work presented in this thesis takes us a step closer in realizing telerehabilitation and telemedicine in general.

REFERENCES

[1] H. Allison, "Method for the acquisition of arm movement data using accelerometers," BS Thesis, Mechanical Engineering, MIT, June 2005

[2] American Heart Association & American Stroke Association, "Know the Facts, Get the Stats," 2007

[3] American Stroke Association, Available: http://www.strokeassociation.org/STROKEORG/AboutStroke/Impact-of-Stroke_UCM_310728_Article.jsp. (Accessed July 15, 2011)

[4] E. R. Bachmann, "Inertial and magnetic tracking of limb segment orientation for inserting humans in synthetic environments," PhD thesis, Naval Postgraduate School, Monterrey, CA, USA, 2000.

[5] S. Barreca, S. L. Wolf, S. Fasoli, R. Bohannon, "Treatment interventions for the paretic upper limb of stroke survivors: a critical review," Neurorehabil. Neural Repair. 2003; 17:220-226.

[6] B. Bobath, Adult Hemiplegia: Evaluation and Treatment. London, England: Heinemann; 1978.

[7] D. Bullock, S. Grossberg, "Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation," Psychological Review, 1988

[8] P. W. Duncan, R. Zorowitz, B. Bates, et al., "Management of adult stroke rehabilitation care: a clinical practice guideline," Stroke. 2005; 36:e100-e143.

[9] T. Flash, "Models of human arm trajectory control," IEEE Engineering in Medicine and Biology Society 10th Annual International Conference, 1988

[10] N. Foley, R. Teasell, J. Jutai, S. Bhogal, and E. Kruger, "Upper extremity interventions," Aug. 2009, Available: http://www.ebrsr.com/uploads/Module_10_upper_extremity_formatted.pdf

[11] E. Foxlin, Handbook of Virtual Environment Technologies, Chapter Motion Tracking Technologies and Requirements, pages 163–210, Lawrence Erlbaum Publishers, 2002.

[12] R. Guillemaud, Y. Caritu, D. David, F. Favre-Reguillon, D. Fontaine, and S. Onnet, "Body motion capture for activity monitoring," International Workshop on New Generation of Wearable Systems for eHealth, Dec. 11-14, Castelvecchio Pascoli, Lucca, Italy, 2003.

[13] M. Hersch, A. G. Billard, "A Model for Imitating Human Reaching Movements," HRI '06, March 2-4, 2006, ACM 1-59593-294-1/06/0003

[14] http://www.nrrl.cahs.colostate.edu/

[15] http://www.sparkfun.com/datasheets/Sensors/IMU/SFE-0012-DS-6DOFAtomic_v3.pdf

[16] http://www.sparkfun.com/products/9184

[17] S. Im, I. Kim, S. Ahn, H. Kim, "Automatic ADL classification using 3-axial accelerometers and RFID sensor," Proceedings of IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, Seoul, Korea, August 2008.

[18] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Journal of Basic Engineering, Vol.82, No. 1, March 1960, pp. 35-46

[19] M. Kawato, Y. Uno, M. Isobe, R. Suzuki, "A Hierarchical Neural-network Model for Control and Learning of Voluntary Movement," Biol. Cybernetics 57, pp 169-185

[20] S. M. Lai, S. Studenski, P. W. Duncan, and S. Perera, "Persisting consequences of stroke measured by the Stroke Impact Scale," *Stroke*, vol. 33, 2002, pp. 1840-1844.

[21] J. Liepert, I. Uhde, S. Graf, O. Leidner, C. Weiller, Motor cortex plasticity during forced-use therapy in stroke patients: a preliminary study. *JNeurol*. 2001;248: 315-321.

[22] Y. Liu, G. Wang, C. Guo, "Analysis for Transverse Sensitivity of the Microaccelerometer," Scientific Research, Engineering, 2009, 1, pp 196-200 (http://www.scirp.org/journal/eng)

[23] J. Lötters, W. Olthuis, P. H. Veltink, and P. Bergveld, "Design, realization and characterization of a symmetrical triaxial capacitive accelerometer for medical applications," *Sens. Actuat. A*, vol. 61, pp. 303–308, 1997.

[24] H. J. Luinge, "Inertial Sensing of Human Movement," PhD Thesis, University of Twente, 2002

[25] H. J. Luinge, P. H. Veltink, "Inclination Measurement of Human Movement Using a 3-D Accelerometer With Autocalibration," IEEE Transactions on Neural Systems and Rehabilitation Engineering, Vol. 12, No. 1, March 2004.

[26] H. J. Luinge, P. H. Veltink, and C. T. M. Baten, “Ambulatory measurement of arm orientation,” Journal of Biomechanics, vol. 40, 2007, pp. 78–85.

[27] D. M. Morris, E. Taub, “Constraint-induced therapy approach to restoring function after neurological injury,” Top Stroke Rehabil. 2001; 8:16-30.

[28] B. Najafi, K. Aminian, A. Ionescu, F. Loew, C. Bula, P. Robert, “Ambulatory system for human motion analysis using a kinematic sensor: Monitoring of daily physical activity in the elderly,” IEEE Transactions On Biomedical Engineering, Vol. 50, No. 6, June 2003

[29] D. S. Nichols-Larsen, P. C. Clark, A. Zeringue, A. Greenspan, and S. Blanton, “Factors influencing stroke survivors’ quality of life during subacute recovery,” Stroke,vol. 36, 2005, pp. 1480-1484.

[30] S. J. Page, S. Sisto, P. Levine, R. E. McGrath, “Efficacy of modified constraint-induced movement therapy in chronic stroke: a single-blinded randomized controlled trial,” Arch Phys Med Rehabil. 2004; 85: 14-18.

[31] M. Panyan, Kan Lawrence, How to Use Shaping: H & H Enterprises; 1980.

[32] S. Patel, R. Hughes, T. Hester, J. Stein, M. Akay, J. Dy, P. Bonato, “A Novel approach to monitor rehabilitation outcomes in stroke survivors using wearable technology,” Proceedings of the IEEE, Vol. 98, No. 3, March 2010.

[33] S. R. Pierce, K. G. Gallagher, S. W. Schaumburg, A. M. Gershkoff, J. P. Gaughan, L. Shutter, “Home forced use in an outpatient rehabilitation program for adults with hemiplegia: a pilot study,” Neurorehabil Neural Repair. 2003; 17:214-219.

[34] A. M. Sabatini, “Quaternion-based strap-down integration method for applications of inertial sensing to gait analysis,” Medical & Biological Engineering & Computing, Vol. 43, 2005.

[35] R. A. Schmidt, T. D. Lee, “Motor Control and Learning: A Behavioral Emphasis,” Champaign, Ill: Human Kinetics; 1999.

[36] M. Stikic, T. Huỳnh, K. Laerhoven, B. Schiele, “ADL Recognition Based on the Combination of RFID and Accelerometer Sensing,” Proceedings of the 2nd International Conference on Pervasive Computing Technologies for Healthcare (Pervasive Health 2008), January, Tampere, Finland, p.258–263, 2008

[37] W. Stockwell, Angle Random Walk, http://www.xbow.com

[38] E. Taub, “Somatosensory deafferentation research with monkeys: implications for rehabilitation medicine,” In: ed. Behavioral Psychology in RehabilitationMedicine: Clinical Applications. Baltimore, Md: Williams & Wilkins; 1980:371-401.

[39] E. Taub, J. E. Crago, L. D. Burgio, et al. “An operant approach to rehabilitation medicine: overcoming learned nonuse by shaping,” *J Exp Anal Behav*. 1994; 61:281-293.

[40] E. Taub, N. E. Miller, T. A. Novack, et al., “Technique to improve chronic motor deficit after stroke,” Arch Phys Med Rehabil. 1993; 74:347-354.

[41] E. Taub, G. Uswatte, T. Elbert, “New treatments in neurorehabilitation founded on basic research,” Nat. Rev. Neurosci. 2002; 3:228-236.

[42] D. Titterton and J. Weston, “Strapdown inertial navigation technology,” The American Institute of Aeronautics and Astronautics, second edition, 2004.

[43] J. Torres, B. O’Flynn, P. Angove, F. Murphy, C. O’Mathuna, “Motion tracking algorithms for inertial measurement,” ACM Conference 2004

[44] G. Uswatte, C. Giuliani, C. Winstein, A. Zeringue, L. Hobbs, and S. L. Wolf, “Validity of accelerometry for monitoring real-world arm activity in patients with subacute stroke: Evidence from the extremity constraint-induced therapy evaluation trial,” Arch Phys Med Rehabil, vol. 87, Oct. 2006.

[45] J. H. Van der Lee, H. Beckerman, G. J. Lankhorst, L. M. Bouter, “Constraint-induced movement therapy,” Arch. Phys. Med. Rehabil., 1999; 80:1606-1607.

[46] G. Welch, E. Foxlin, “Motion tracking: No silver bullet, but a respectable arsenal,” IEEE Computer Graphics and Applications, November/December 2002

[47] Wikipedia, http://en.wikipedia.org/wiki/Coriolis_effect

[48] C. J. Winstein, “Designing practice for motor learning: clinical impressions,” Contemporary Management of Motor Control Problems: Proceedings of the II Step Conference. Alexandria, Va: Foundation for Physical Therapy; 1991.

[49] C. J. Winstein, J. P. Miller, S. Blanton, et al., “Methods for a multisite randomized trial to investigate the effect of constraint-induced movement therapy in improving upper-extremity function among adults recovering from a cerebrovascular stroke,” Neurorehabil. Neural. Repair., 2003; 17:137-152.

[50] S. L. Wolf, S. Blanton S, H. Baer, J. Breshears, A. J. Butler, “Repetitive task practice: a critical review of constraint induced movement therapy in stroke,” Neurologist. 2002; 8:325-338.

[51] S. L. Wolf, D. E. Lecraw, L. A. Barton, B. B. Jann, “Forced use of hemiplegic upper extremities to reverse the effect of learned nonuse among chronic stroke and head injured patients,” Exp Neurol. 1989; 104:125-132.

[52] S. L. Wolf, C. J. Winstein, J. P. Miller, et al., “Effect of constraint-induced movement therapy on upper extremity function 3 to 9 months after stroke - The EXCITE randomized clinical trial,” JAMA-Journal of the American Medical Association. Nov 2006; 296(17):2095-2104.

[53] Oliver J. Woodman, “An Introduction to inertial navigation,” Technical Report 696, Computer Laboratory, University of Cambridge, August 2007, UCAM-CL-TR-696 ISSN 1476-2986

[54] Paul Zarchan, Howard Musoff, “Fundamentals of kalman filtering: A Practical approach,” Second Edition, Progress in Astronautics and Aeronautics, Volume 208, American Institute of Astronautics and Aeronautics

[55] H. Zheng, N. D. Black, N. D. Harris, “Position-sensing technologies for movement analysis in stroke rehabilitation,” Medical & Biological Engineering & Computing 2005, Vol. 43, 413-420

[56] H. Zhou, and H. Hu, “A survey human movement tracking and a research proposal,” Smart Equal Project Report

Appendix A

Source Code

A.1 Arm Usage Calculation (Method 1)

```
function quantify
ch=0;
FlagOpen=0;
while(ch ~= 4)
   fprintf('\n');
   disp(' --------------   Menu   -------------- ');
   disp(' 1. Open New File');
   disp(' 2. Plot Data');
   disp(' 3. View Results');
   disp(' 4. Exit');
   ch=input(' Enter your choice (1/2/3/4) -: ');

   if(ch==1)
      if(FlagOpen ~= 0)
         fclose('all');
      end
      FlagOpen=1;
      fprintf('\n');
      fid=input(' Enter the file name - : ','s');
      [X X_val Y Y_val Z Z_val]=textread(fid,'%s %f %s %f %s %f',-1,'delimiter','=');
      raw_data=[X_val Y_val Z_val];
      a=size(raw_data);
      avg=mean(raw_data);
      if (avg(1,1)>10 && avg(1,2)>10 && avg(1,3)>10)
         raw_data=raw_data/800;
      end
      avg=mean(raw_data);
   end

   if(ch==2)
      if(FlagOpen==0)
         fprintf('\n');
         disp(' ---------------------------------- ');
         disp(' |   ERROR : NO FILE OPENED !!!   |');
         disp(' ---------------------------------- ');
      else
         figure(1);
         subplot(3,1,1);
         plot(raw_data(:,1));
         title(' Raw Data in X Dir ( in terms of "g") ');
         subplot(3,1,2);
         plot(raw_data(:,2));
         title(' Raw Data in Y Dir ( in terms of "g") ');
```

```
        subplot(3,1,3);
        plot(raw_data(:,3));
        title(' Raw Data in Z Dir ( in terms of "g") ');
    end

elseif(ch==3)
    if(FlagOpen==0)
        fprintf('\n');
        disp(' ---------------------------------- ');
        disp(' |   ERROR : NO FILE OPENED !!!   |');
        disp(' ---------------------------------- ');
    else
        acti=zeros(1,4);
        acti_count=zeros(1,4);
        boolean=zeros(a);
        thres=0.022;
        flag=0;
        flag1=0;
        da(1,:)=0;
        for i=1:a(1)
            for j=1:3
                if ( raw_data(i,j)>avg(1,j) )
                    boolean(i,j)=1;
                else
                    boolean(i,j)=-1;
                end
                if( i~=1 )
                    da(i,j)=raw_data(i,j)-raw_data(i-1,j);
                    if (abs(da(i,j)) >thres)
                        flag=1;
                        acti(1,j)=acti(1,j)+1;
                    end
                end
            end
            if (flag == 1)
                acti(1,4)=acti(1,4)+1;
                flag=0;
            end
        end
        for i=2:a(1)
            for j=1:3
                da_boolean(i,j)=boolean(i,j)-boolean(i-1,j);
                if (abs(da_boolean(i,j)) == 2 )
                    acti_count(1,j)=acti_count(1,j)+1;
                    flag1=1;
                end
            end
            if( flag1==1 )
                flag1=0;
                acti_count(1,4)=acti_count(1,4)+1;
            end
        end
        acti=acti/a(1)*100;
        fprintf('\n');
        fprintf('The observation period was %0.1f seconds.',(a(1)/10));
        fprintf('\n');
```

```
            fprintf('\n');
            disp('Affected');
            disp('     Usage_X    Usage_Y      Usage_Z      % Usage');
            format('short','g');
            disp(acti);
            fprintf('\n');
            disp('Writing the results to Result.txt .....');
            fid_op=fopen('Result.txt','at');
            fprintf(fid_op,'\n');

fprintf(fid_op,'%s\t%0.1f\t%0.2f\t%0.2f\t%0.2f\t%0.2f',fid,(a(1,1)/10),acti(1,1),acti(1,2),acti(1,3),acti(1,4));
            fclose('all');
        end
    end
end
clear all;
```

A.2 Activity Count Calculation (Method 2) and Histogram of Distance Traversed

```
%
% Code to find out activity count
% First part locates each activity individually and finds distance
% measure
% Second part does the same on Noise removed data
% Third part finds the time based distance measure and plots histogram
%

clear all;
ch=0;
fid=input(' Enter the file name - : ','s');
[X X_val Y Y_val Z Z_val]=textread(fid,'%s %f %s %f %s %f',-1,'delimiter','=');

raw_data=[X_val Y_val Z_val];
a=size(raw_data);
avg=mean(raw_data);

% The following conversion needs to be revisited for WiTilt v3.0
if (avg(1,1)>10 && avg(1,2)>10 && avg(1,3)>10)
    raw_data=raw_data/800;
end

avg=mean(raw_data);
std_dev=std(raw_data,1,1);

m=input(' Enter the number of points to sample for filter -:');

wt=1;

mov_avg=zeros(a);
mov_avg(1:m,:)=raw_data(1:m,:);
mov_avg(a(1,1)-m+1:a(1,1),:)=raw_data(a(1,1)-m+1:a(1,1),:);

mov_std=zeros(a);
sharpen=zeros(a);
```

```
temp=zeros(2*m+1,a(1,2));

% Processing data using moving statistical measures
for i=m+1:1:(a(1,1)-m)
   temp=raw_data(i-m:i+m,:);
   mov_avg(i,:)=sum(temp);
   mov_avg(i,:)=(mov_avg(i,:)+(wt-1)*raw_data(i,:))/(2*m+wt);
   mov_std(i,:)=std(temp);
   for k=1:m
      sharpen(i,:)=sharpen(i,:)-(raw_data(i+k,:)+raw_data(i-k,:));
   end
   sharpen(i,:)=sharpen(i,:)+(2*m)*raw_data(i,:);
end
indirect_hpf=raw_data-mov_avg;

figure();
subplot(3,1,1);
hold on;
plot(raw_data(:,1));
plot(mov_avg(:,1), 'r');
hold off;
title(['X axis raw data & its Moving Average with m = ', int2str(m)]);
subplot(3,1,2);
hold on;
plot(raw_data(:,2));
plot(mov_avg(:,2), 'r');
hold off;
title(['Y axis raw data & its Moving Average with m = ', int2str(m)]);
subplot(3,1,3);
hold on;
plot(raw_data(:,3));
plot(mov_avg(:,3), 'r');
hold off;
title(['Z axis raw data & its Moving Average with m = ', int2str(m)]);

% Removing static noise of the sensor
% 0.09 is observed static sensor noise
ind_hpf_nr=indirect_hpf;
for j=1:3
   for i=1:a(1,1)
      if abs(indirect_hpf(i,j))<= 0.09
         ind_hpf_nr(i,j)=0;
      end
   end
end

figure();
subplot(3,1,1);
plot(indirect_hpf(:,1));
title('X axis data after subtracting moving average');
subplot(3,1,2);
plot(indirect_hpf(:,2));
title('Y axis data after subtracting moving average');
subplot(3,1,3);
plot(indirect_hpf(:,3));
```

```
title('Z axis data after subtracting moving average');

figure();
subplot(3,1,1);
plot(ind_hpf_nr(:,1));
title('X axis data after subtracting moving average & removing noise');
subplot(3,1,2);
plot(ind_hpf_nr(:,2));
title('Y axis data after subtracting moving average & removing noise');
subplot(3,1,3);
plot(ind_hpf_nr(:,3));
title('Z axis data after subtracting moving average & removing noise');

%% Plotting All Processed Data
figure();
subplot(4,1,1);
plot(raw_data(:,1));
title('Raw Data');
subplot(4,1,2);
plot(mov_avg(:,1));
title('Moving Average Filtered Data');
subplot(4,1,3);
plot(sharpen(:,1));
title('Moving High Pass Filtered Data');
subplot(4,1,4);
plot(indirect_hpf(:,1));
title('Indirect High Pass Filtered Data');
figure();
subplot(4,1,1);
plot(raw_data(:,2));
title('Raw Data');
subplot(4,1,2);
plot(mov_avg(:,2));
title('Moving Average Filtered Data');
subplot(4,1,3);
plot(sharpen(:,2));
title('Moving High Pass Filtered Data');
subplot(4,1,4);
plot(indirect_hpf(:,2));
title('Indirect High Pass Filtered Data');
figure();
subplot(4,1,1);
plot(raw_data(:,3));
title('Raw Data');
subplot(4,1,2);
plot(mov_avg(:,3));
title('Moving Average Filtered Data');
subplot(4,1,3);
plot(sharpen(:,3));
title('Moving High Pass Filtered Data');
subplot(4,1,4);
plot(indirect_hpf(:,3));
title('Indirect High Pass Filtered Data');

%% Noise Removed Indirect HPF Analysis
```

```
acti_count=zeros(1,4);
flag_b=zeros(a(1,1),4);
temp=zeros(a(1,1),1);
interval=zeros(1,2);

for i=1:a(1,1)
temp(i)=sqrt(ind_hpf_nr(i,1).^2+ind_hpf_nr(i,2).^2+ind_hpf_nr(i,3).^2);
end

ind_hpf_nr=[ind_hpf_nr temp];
dead=input(' Enter the no. of samples for dead period -: ');

for j=1:4
   i_prev=0;          % Previous non-zero 'i'
   prev_i=0;          % Previous 'i'
   cnt=0;             % Counter to count no. of consecutive zeroes
   flagz=0;
   for i=1:a(1,1)
        if ind_hpf_nr(i,j) ~= 0
           if(i_prev==0 || flagz==1)
              flagz=0;
              i_prev=i;
              acti_count(1,j)=acti_count(1,j)+1;
              flag_b(i,j)=1;
           end
        else
           if (i-prev_i==1 && flagz==0)
              cnt=cnt+1;
           else
              cnt=1;
           end
           prev_i=i;
           if (cnt==dead && i_prev~=0)
              flagz=1;
              if (j==4)
                 % array to store activity intervals
                 interval=[interval; i_prev (i-9)];
              end
           end
        end
   end
end

disp(acti_count);

%% Acc. Vel & Distance Calculation

r=ind_hpf_nr;
vel=zeros(a);
vel3=zeros(a);
vel4=zeros(a);
dist1=zeros(a(1,1),4);
dist2=zeros(a(1,1),4);
dist3=zeros(a(1,1),4);
```

```
dist4=zeros(a(1,1),4);

for i=2:acti_count(1,4)+1
   for j=interval(i,1):interval(i,2)
      for k=1:a(1,2)
         if r(j,k)>r(j-1,k)
            vel(j,k)=vel(j-1,k)+r(j,k);
         else
            vel(j,k)=vel(j-1,k)-r(j,k);
         end
      end
   end
end

for i=2:acti_count(1,4)+1
   for j=interval(i,1):interval(i,2)
      for k=1:a(1,2)
         dist1(j,k)=dist1(j-1,k)+(vel(j,k)+vel(j-1,k))/2;
         dist2(j,k)=dist2(j-1,k)+vel(j-1,k);
      end
      dist1(j,4)=sqrt(dist1(j,1).^2+dist1(j,2).^2+dist1(j,3).^2);
      dist2(j,4)=sqrt(dist2(j,1).^2+dist2(j,2).^2+dist2(j,3).^2);
   end
end

% Time based ditance calcultaion
win=50;     % Window width for time based activity factor measurement
part=ceil(a(1,1)/win);      % No. of windows in a sample file
for j=1:a(1,2)
   for i=1:part
      vel3((i-1)*win+1,j)=0;
      vel4((i-1)*win+1,j)=0;
      dist3((i-1)*win+1,j)=0;
      dist4((i-1)*win+1,j)=0;
      % Angle calculation here
      angle((i-1)*win+1,j)=0;
      for k=2:win
         if ((i-1)*win+k)<=a(1,1)
            if r((i-1)*win+k,j)>r((i-1)*win+k-1,j)
               vel3((i-1)*win+k,j)=vel3((i-1)*win+k-1,j)+r((i-1)*win+k,j);
            else
               vel3((i-1)*win+k,j)=vel3((i-1)*win+k-1,j)-r((i-1)*win+k,j);
            end
            if raw_data((i-1)*win+k,j)>raw_data((i-1)*win+k-1,j)
               vel4((i-1)*win+k,j)=vel4((i-1)*win+k-1,j)+indirect_hpf((i-1)*win+k,j);
            else
               vel4((i-1)*win+k,j)=vel4((i-1)*win+k-1,j)-indirect_hpf((i-1)*win+k,j);
            end
            dist3((i-1)*win+k,j)=dist3((i-1)*win+k-1,j)+vel3((i-1)*win+k-1,j);
            dist4((i-1)*win+k,j)=dist4((i-1)*win+k-1,j)+vel4((i-1)*win+k-1,j);
         else
            break;
         end
      end
   end
```

```
end

dist3(:,4)=sqrt(dist3(:,1).^2+dist3(:,2).^2+dist3(:,3).^2);
dist4(:,4)=sqrt(dist4(:,1).^2+dist4(:,2).^2+dist4(:,3).^2);

figure();
subplot(3,1,1);
plot(indirect_hpf(:,1));
title('X axis Indirect HPF');
subplot(3,1,2);
plot(indirect_hpf(:,2));
title('Y axis Indirect HPF');
subplot(3,1,3);
plot(indirect_hpf(:,3));
title('Z axis Indirect HPF');
```

A.3 Weighted Area of Histogram of Distance Traversed

```
clear all;
path = './Trial 8/';
file = {'sub', '_', 'act'};
type = {'.txt', '.xls'};
col = {'B2', 'C2', 'D2', 'E2', 'F2', 'G2', 'H2', 'I2', 'J2', 'K2','L2','M2','N2','O2','P2','Q2'};
max_dist = 120;
act_in = 4;
sub_in = 12;
histo = zeros( max_dist+1, sub_in);
hist_norm = zeros( max_dist+1, sub_in);
temp = zeros( 1, max_dist+1);
bins = [0:max_dist]';
m_in = 3;
win_in = 5;
f_sample = 83;
t_sample = 1/f_sample;

colors = [ 0.25, 0.25, 0.25; 0.25, 0.25, 0; 0.25, 0, 0.25; 0.4, 0.3, 0.6;
        0, 0.25, 0.25; 0.5, 0.4, 0.2; 0.4, 0.1, 0.6; 0.75, 0.5, 0.4;
        0.6, 0.8, 0.5; 0.75, 0.75, 0.75; 0, 0.75, 0.5; 0.5, 0.75, 0.8];

set(gcf,'DefaultAxesColorOrder', colors);

% Zero 'g' values obtained from statistical analysis
x_zero = 333;
y_zero = 338;
z_zero = 344;

for acti = 2:act_in
   s2 = strcat(path, file{1,3}, file{1,2}, int2str(acti), file{1,2}, int2str(m_in), file{1,2}, int2str(win_in),
type{1,2});
   for sub = 1:sub_in
      s1 = strcat(path, file{1,1}, int2str(sub), file{1,2}, file{1,3}, file{1,2}, int2str(acti), type{1,1});
      [a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12] = textread(s1, '%d\t %d\t %d\t %d\t %d\t %d\t %d\t %d\t
%d\t %d\t %d\t %d\t %*d');
      raw_data = [ a10, a11, a12];
```

```
a = size(raw_data);
avg = mean(raw_data);

% If the data has raw ADC values convert them to 'g' values
% by centering the data around zero
% and dividing by ADC count for '1g'
% For a range of -2g to +2g -> ADC(1g) = ADC(0g)/2
if (avg(1,1) > 10 && avg(1,2) > 10 && avg(1,3) > 10)
   raw_data(:,1) = raw_data(:,1) - x_zero;
   raw_data(:,2) = raw_data(:,2) - y_zero;
   raw_data(:,3) = raw_data(:,3) - z_zero;
   raw_data(:,1) = raw_data(:,1) / (x_zero/2);
   raw_data(:,2) = raw_data(:,2) / (y_zero/2);
   raw_data(:,3) = raw_data(:,3) / (z_zero/2);
end

avg = mean(raw_data);
std_dev = std(raw_data,1,1);

m = m_in;
m = m_in * f_sample;
wt = 1;
mov_avg = zeros(a);
mov_avg(1:m,:) = raw_data(1:m,:);
mov_avg(a(1,1)-m+1:a(1,1),:) = raw_data(a(1,1)-m+1:a(1,1),:);
mov_std = zeros(a);
temp = zeros( 2*m+1, a(1,2));

% Processing data using moving statistical measures
for i = m+1:1:(a(1,1)-m)
   temp = raw_data(i-m:i+m,:);
   mov_avg(i,:) = sum(temp);
   mov_avg(i,:) = (mov_avg(i,:)+(wt-1)*raw_data(i,:))/(2*m+wt);
   mov_std(i,:) = std(temp);
end
indirect_hpf = raw_data - mov_avg;

% Removing static noise of the sensor
% 0.09 is observed static sensor noise
% Need to check for latest sensor
ind_hpf_nr = indirect_hpf;
for j = 1:3
   for i = 1:a(1,1)
      if abs(indirect_hpf(i,j)) <= 0.09
         ind_hpf_nr(i,j) = 0;
      end
   end
end


% Convert raw ADC values to 'g'
ind_hpf_nr = 9.80665 * ind_hpf_nr;

% Time based distance calcultaion
```

```
        % Calculating actual distance traversed in meters
        % Achieved by multiplying by t_sample
        r = ind_hpf_nr;
        vel = zeros(a);
        dist = zeros(a(1,1),4);
        win_in = 5;
        win = win_in * f_sample;     % Window width for time based activity factor measurement
        part = ceil(a(1,1)/win);      % No. of windows in a sample file
        for j = 1:a(1,2)
           for i = 1:part
              vel((i-1)*win+1,j) = 0;
              dist((i-1)*win+1,j) = 0;
              % Angle calculation here
%                angle((i-1)*win+1,j)=0;
              for k = 2:win
                 if ((i-1)*win+k) <= a(1,1)
                    if raw_data((i-1)*win+k,j)>raw_data((i-1)*win+k-1,j)
                       vel((i-1)*win+k,j) = vel((i-1)*win+k-1,j) + t_sample*indirect_hpf((i-1)*win+k,j);
                    else
                       vel((i-1)*win+k,j) = vel((i-1)*win+k-1,j) - t_sample*indirect_hpf((i-1)*win+k,j);
                    end
                    dist((i-1)*win+k,j) = dist((i-1)*win+k-1,j) + t_sample*vel((i-1)*win+k-1,j);
                 else
                    break;
                 end
              end
           end
        end

        % Calculate equivalent distance in 'cm'
        dist(:,4) = 100 * sqrt(dist(:,1).^2+dist(:,2).^2+dist(:,3).^2);
        xlswrite(s2, dist(:,4), 'Equi_Distance(Window based)(cm)', col{1,sub});
        temp = hist( dist(:,4), bins);
        histo(:,sub) = temp';
        hist_norm(:,sub) = 100 * (histo(:,sub)/(a(1,1)));
     end

     % Calculate the area under the normalized historgram curve
     % as an observation based on which we would rank the
     % performance of subjects
     start_offset = 0.5*max_dist;
     perf = zeros(1, sub_in);
     for i = (start_offset + 1):max_dist
        perf(1,:) = (i - start_offset)*hist_norm(i,:) + perf(1,:);
     end
     perf = perf';

     start_offset = 0.75*max_dist;
     perf2 = zeros(1, sub_in);
     for i = (start_offset + 1):max_dist
        perf2(1,:) = (i - start_offset)*hist_norm(i,:) + perf2(1,:);
     end
     perf2 = perf2';

     % Plot composite Normalized Historgram of all subjects for comparison
```

```
    figure();
    title(['Performace Comparison for Activity ', int2str(acti)]);
    xlabel('Distance Covered (cm)');
    ylabel('Normalized Histogram Count (%)');
    set(gca,'NextPlot','add');
    set(gcf,'DefaultAxesColorOrder',colors);
    hold on;
    plot(bins, hist_norm(:,1), '-rx');
    plot(bins, hist_norm(:,2), '-bo');
    plot(bins, hist_norm(:,3), '-ms');
    plot(bins, hist_norm(:,4), '-cd');
    plot(bins, hist_norm(:,5), '-kh');
    plot(bins, hist_norm(:,6), '-rp');
    plot(bins, hist_norm(:,7), '-b>');
    plot(bins, hist_norm(:,8), '-mv');
    plot(bins, hist_norm(:,9), '-c^');
    plot(bins, hist_norm(:,10), '-k<');
    plot(bins, hist_norm(:,11), '-mo');
    plot(bins, hist_norm(:,12), '-bs');

    for i=1:sub_in
        legends{i} = strcat(file{1,1}, file{1,2}, int2str(i));
    end
    h = legend(legends);
%     h = legend('sub_1', 'sub_2', 'sub_3', 'sub_4', 'sub_5', 1);
    set(h,'Interpreter','none');
    hold off;
end
```

A.4 Simulator

A.4.1 Top Level File

```
clear all;
close all;
clc;

quadrant = 1;
f_sample = 100;
t        = 1/f_sample;
g = 9.80665;
tick = 0.00403;

sim_track_leg1_end = 300;
state = simulate_test_track(1, f_sample, sim_track_leg1_end/f_sample);
```

A.4.2 Simulate Rectangular Track

```
function state = simulate_test_track(quadrant, f_sample, t_end)

if nargin ~= 3
    error('Wrong number of input arguments...');
```

```
    error('Need 3 arguments - quadrant, sampling frequency and duration of motion...');
end

if (quadrant < 0 || quadrant > 4)
   error('Quardrant value should be an interger from 1 to 4...');
end

if ( quadrant == 1 || quadrant == 4)
   xscale = 1;
else
   xscale = -1;
end

if ( quadrant == 1 || quadrant == 2)
   yscale = 1;
else
   yscale = -1;
end

amp = 1.5/t_end;

[ tmp_t, tmp_v] = gen_vel(amp*xscale, 0, 0.02, t_end-0.02, t_end, f_sample);
t = tmp_t;
vx = tmp_v;

[ tmp_t, tmp_v] = gen_vel(0*yscale, 0, 0.02, t_end-0.02, t_end, f_sample);
vy = tmp_v;

[ tmp_t, tmp_v] = gen_vel(0*xscale, t_end, t_end+0.02, 2*t_end-0.02, 2*t_end, f_sample);
t = [t; tmp_t];
vx = [vx; tmp_v];

[ tmp_t, tmp_v] = gen_vel(amp*yscale, t_end, t_end+0.02, 2*t_end-0.02, 2*t_end, f_sample);
vy = [vy; tmp_v];

[ tmp_t, tmp_v] = gen_vel(-amp*xscale, 2*t_end, 2*t_end+0.02, 3*t_end-0.02, 3*t_end, f_sample);
t = [t; tmp_t];
vx = [vx; tmp_v];

[ tmp_t, tmp_v] = gen_vel(0*yscale, 2*t_end, 2*t_end+0.02, 3*t_end-0.02, 3*t_end, f_sample);
vy = [vy; tmp_v];

[ tmp_t, tmp_v] = gen_vel(0*xscale, 3*t_end, 3*t_end+0.02, 4*t_end-0.02, 4*t_end, f_sample);
t = [t; tmp_t];
vx = [vx; tmp_v];

[ tmp_t, tmp_v] = gen_vel(-amp*yscale, 3*t_end, 3*t_end+0.02, 4*t_end-0.02, 4*t_end, f_sample);
vy = [vy; tmp_v];

ax = calc_acc(vx, f_sample, 0);
ay = calc_acc(vy, f_sample, 0);

jx = calc_jerk(ax, f_sample, 0);
```

```
jy = calc_jerk(ay, f_sample, 0);

sx = calc_dist(vx, f_sample, 0);
sy = calc_dist(vy, f_sample, 0);
%
figure();
plot(vx, 'bx');
title('Velocity in X', 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('Velocity (m/s)', 'FontSize', 12);

figure();
plot(vy, 'bx');
title('Velocity in Y', 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('Velocity (m/s)', 'FontSize', 12);

figure();
plot(ax, 'b');
title('Acceleration in X', 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('Acceleration (m/s^2)', 'FontSize', 12);

figure();
plot(ay, 'b');
title('Acceleration in Y', 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('Acceleration (m/s^2)', 'FontSize', 12);

figure();
plot(sx, sy, 'x');
title('Simulated Track', 'FontSize', 14);
xlabel('Distance (m)', 'FontSize', 12);
ylabel('Distance (m)', 'FontSize', 12);

th = zeros(size(vx));
w = zeros(size(vx));

state = [sx sy vx vy ax ay th w]';

end
```

A.4.3 Velocity Generation

```
function [t, vel] = gen_vel(peak, t_start, start_offset, end_offset, t_end, f_sample)

t = t_start:1/f_sample:t_end;

start_indx = ceil( (start_offset - t_start) * f_sample);
end_indx = floor( (end_offset - t_start) * f_sample);

mean = (end_offset + start_offset)/2;
```

```
std_dev = (end_offset - start_offset)/7;

% Generate the bell shaped curve as velocity profile against time
v = zeros(size(t));

if (peak ~= 0)
   v(1:length(t)-1) = peak * exp( -0.5 * ( ( t(1:length(t)-1) - mean) / std_dev) .^ 2);
end

vel = v';
t = t';
end
```

A.4.4 Acceleration Calculation

```
function acc = calc_acc(vel, f_sample, init_val)

acc(1) = init_val;
for i = 2:length(vel)
   acc(i) = ( vel(i) - vel(i-1) ) * f_sample;
end;

acc = acc';
end
```

A.4.5 Jerk Calculation

```
function jerk = calc_jerk(acc, f_sample, init_val)

jerk(1) = init_val;
for i = 2:length(acc)
   jerk(i) = ( acc(i) - acc(i-1) ) * f_sample;
end;

jerk = jerk';
end
```

A.4.6 Distance Calculation

```
function dist = calc_dist(vel, f_sample, init_val)

dist(1) = init_val;
for i=2:length(vel)
   dist(i) = vel(i)/f_sample + dist(i-1);
end

dist = dist';
end
```

A.5 Kalman Filter Analysis

```
clear all;
close all;
clc;

quadrant = 1;
f_sample = 100;
t        = 1/f_sample;
g = 9.80665;
tick = 0.00403;

sim_track_leg1_end = 300;
state = simulate_test_track(1, f_sample, sim_track_leg1_end/f_sample);

t_end = length(state);  % Time till which Kalman filter will be ON
Q_scale = 0.01;
Q_val = 0.1;
alpha = 1.0;

% mux = zeros(6,1); Q = diag([0.00001, 0.00001, 0.001, 0.001, 0.1, 0.1]);
mux = zeros(6,1);
Q = Q_scale*diag([0.0, 0.0, 0.0, 0.0, Q_val, Q_val]);

muy = mean(state(5:6,:)');
stdev = 2*tick*g;
sigy = stdev;
sigy1 = 1*sigy;
R = diag([sigy1*sigy1, sigy1*sigy1]);

const_acc = 0;
X0 = zeros(6,1);

P0 = 10*diag([0.1, 0.1, 0.1, 0.1, 0.1, 0.1]);%Q;

X_pri = zeros(6,t_end);
y     = zeros(2,t_end);
X_pos = zeros(6,t_end);
X_tru = state(1:6,1:t_end);
z     = zeros(size(state(5:6, 1:t_end)));

i=1;

F0 = [1  0   t   0   0   0;
      0  1   0   t   0   0;
      0  0   1   0   t   0;
      0  0   0   1   0   t;
      0  0   0   0   alpha   0;
      0  0   0   0   0   alpha];

H = [0  0   0   0   1   0;
     0  0   0   0   0   1];
```

```
F = F0;

i = 1;
% z(:,i)     = H*X0 + normrnd(muy, sigy);
z(:,i)     = state(5:6,i) + normrnd(muy, sigy)';
X_pri(:,i) = F*X0;
X_tru(:,i) = F*X0;
P_pri      = F*P0*F' + Q;
y(:,i)     = z(:,i) - H*X_pri(:,i);
S          = H*P_pri*H' + R;
K          = P_pri*H'*inv(S);
X_pos(:,i) = X_pri(:,i) + K*y(:,i);
P_pos      = ( eye(6) - K*H )*P_pri;

figure();
plot(1,P_pos(5,5), 'r+');
hold on;

for i = 2:t_end
   z(:,i)     = H*X_tru(:,i) + normrnd(muy, sigy)';
   X_pri(:,i) = F*X_pos(:,i-1);
   P_pri      = F*P_pos*F' + Q;
   y(:,i)     = z(:,i) - H*X_pri(:,i);
   S          = H*P_pri*H' + R;
   K          = P_pri*H'*inv(S);
   X_pos(:,i) = X_pri(:,i) + K*y(:,i);
   P_pos      = ( eye(6) - K*H )*P_pri;
   plot(i,P_pos(6,6), 'r+');
end

hold off;
title('Covariance in Y acceleration', 'FontSize', 14);
xlabel('Samples', 'FontSize', 12);
ylabel('Covariance Value', 'FontSize', 12);

figure();
hold on;
plot(X_pos(1,:),X_pos(2,:),'-or','Markersize',6);

% figure();
plot(X_tru(1,:),X_tru(2,:),'-ok','Markersize',6);
title('True State (Black) and Estimated State (Red)', 'FontSize', 14);
xlabel(' Distance (m) ', 'FontSize', 12);
ylabel(' Distance (m) ', 'FontSize', 12);

figure();
plot(1:length(X_tru), z(2,:), '-ob');
hold on;
plot(1:length(X_tru), X_tru(6,:), '-ok');
plot(1:length(X_tru), X_pos(6,:), '-xr');
plot(1:length(X_tru), sigy1, '-xg');
plot(1:length(X_tru), -sigy1, '-xg');
hold off;
```

```
title(['Ideal (Black), Measured (Blue) and Filtered (Red) Acceleration Values and Theoretical SD=' num2str(sigy), ', \alpha=', num2str(alpha), ' & Q_k=', num2str(Q_scale*Q_val)], 'FontSize', 14);
xlabel('Number of Measurements', 'FontSize', 12);
ylabel('Acceleration (m/s^2)', 'FontSize', 12);
xlim([150 750]);

figure();
plot(1:length(X_tru), X_tru(6,:)-X_pos(6,:), '-ok');
hold on;
plot(1:length(X_tru), sqrt(P_pos(6,6)), '-xg');
plot(1:length(X_tru), -sqrt(P_pos(6,6)), '-xg');
hold off;
title(['Error in estimate of Acceleration Values and Filter SD =' num2str(sqrt(P_pos(6,6))), ', \alpha=', num2str(alpha), ' & Q_k=', num2str(Q_scale*Q_val)], 'FontSize', 14);
xlabel('Number of Measurements', 'FontSize', 12);
ylabel('Error in Estimate of Acceleration (m/s^2)', 'FontSize', 12);
```

A.6 2D Motion Tracking Algorithm

```
clear all;
close all;

type = {'.txt', '.xls'};
files = {'clk_wave_top_rev', 'clk_wave_3phase', 'clk_wave_2period', 'clk_wave_1period_rh'};

g = 9.80665;
tick = 0.0041;
Fs = 100;
T = 1/Fs;
flp = 4.5;       % LPF cutoff freq
fhp = 0.07;     % HPF cutoff freq
lp_tb = 0.05;   % HPF transition band (as a % of fhp)
hp_tb = 02;     % HPF transition band (as a % of flp)
hp_pbr = 0.1;   % HPF Pass band ripple (dB)
hp_sbr = -40;   % LPF Stop band ripple (dB)
lp_pbr = 0.1;   % HPF Pass band ripple (dB)
lp_sbr = -40;   % LPF Stop band ripple (dB)
bp_pbr = 0.2;   % HPF Pass band ripple (dB)
bp_sbr = -40;   % LPF Stop band ripple (dB)

data_midpt = [0 483 476 550 512 512 512];

start = [2800; 2500; 2500; 2400; 2100];
finish = [4400; 4000; 4000; 3600; 4200];

bias_corr = 1;
mov_avg_d = 0;
buf = 0;

i=1;
s1 = strcat(files{i}, type{1});
```

```
[A cnt X_val Y_val Z_val p_val r_val y_val Z] = textread(s1, '%c %d %d %d %d %d %d %d %c', -1,'delimiter','\t');
raw_data=[cnt X_val Y_val Z_val p_val r_val y_val];
raw_avg = mean(raw_data(:,2:7));
std_dev = std(raw_data(:,2:7),1,1);

% Convert to SI units
% raw_data(:,2:4) = raw_data(:,2:4) - data_midpt(1,2:4);
raw_data(:,2) = raw_data(:,2) - data_midpt(1,2);
raw_data(:,3) = raw_data(:,3) - data_midpt(1,3);
raw_data(:,4) = raw_data(:,4) - data_midpt(1,4);
raw_data(:,2:4) = raw_data(:,2:4) * g * tick;

% FFT Calculation
len = length(raw_data);
NFFT = 2^nextpow2(len); % Next power of 2 from length of raw_data
Xf = fft(raw_data(:,2),NFFT)/len;
Yf = fft(raw_data(:,3),NFFT)/len;
Zf = fft(raw_data(:,4),NFFT)/len;
f = Fs/2*linspace(0,1,NFFT/2);
figure();
subplot(2,1,1);
plot(raw_data(:,2));
title(['Raw data for X-axis for file \', s1], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(f,2*abs(Xf(1:NFFT/2)))
title(['Single-Sided Amplitude Spectrum of X-axis of file \', s1], 'FontSize', 14);
xlabel('Frequency (Hz)', 'FontSize', 12);
ylabel('|X(f)|', 'FontSize', 12);
figure();
subplot(2,1,1);
plot(raw_data(:,3));
title(['Raw data for Y-axis for file \', s1], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(f,2*abs(Yf(1:NFFT/2)))
title(['Single-Sided Amplitude Spectrum of Y-axis of file \', s1], 'FontSize', 14);
xlabel('Frequency (Hz)', 'FontSize', 12);
ylabel('|X(f)|', 'FontSize', 12);

f_bpf = [fhp fhp*(1 + hp_tb) flp flp*(1 + lp_tb)];
a_bpf = [0 1  0];
dev_bpf = [10^(bp_sbr/20) (10^(bp_pbr/20)-1)/(10^(bp_pbr/20)+1)  10^(bp_sbr/20)];
[n_bpf,f0_bpf,a0_bpf,w_bpf] = firpmord(f_bpf, a_bpf, dev_bpf, Fs);
b_bpf = firpm(n_bpf, f0_bpf, a0_bpf, w_bpf);

bpf_data = filter(b_bpf, 1, raw_data);

%% Remove DC and apply moving average

data_in = bpf_data(start(i):finish(i),:);
```

```
act_avg = mean(data_in(:,2:3))
if (bias_corr)
   data_in(:,2) = data_in(:,2) - act_avg(1);
   data_in(:,3) = data_in(:,3) - act_avg(2);
end

in_size = size(data_in);
half_win_in = 3*0.05;
half_win = floor(half_win_in * Fs);
wt = 1;
mov_avg = zeros(in_size);
mov_avg(1:half_win,:) = data_in(1:half_win,:);
mov_avg( in_size(1,1)-half_win+1:in_size(1,1), :) = data_in( in_size(1,1)-half_win+1:in_size(1,1), :);
mov_std = zeros(in_size);
temp = zeros( 2*half_win+1, in_size(1,2));

% Processing data using moving statistical measures
for ind = half_win+1:1:(in_size(1,1)-half_win)
   temp = data_in(ind-half_win:ind+half_win,:);
   mov_avg(ind,:) = sum(temp);
   mov_avg(ind,:) = (mov_avg(ind,:)+(wt-1)*data_in(ind,:))/(2*half_win+wt);
   mov_std(ind,:) = std(temp);
end

%% Find samples above a threshold to determine
%  possible activity period candidtates

bool = zeros(in_size);
thres = 0.2;

for a=1:in_size(1,1)
   for b=2:3
      if (mov_avg(a,b) > thres)
         bool(a,b) = 1;
      end
      if (mov_avg(a,b) < -thres)
         bool(a,b) = -1;
      end
   end
end

%% Calculate the locations of change in bool values

d_bool = zeros(in_size);

for a=2:in_size(1,1)
   d_bool(a,:) = bool(a,:) - bool(a-1,:);
end

figure();
subplot(2,1,1);
plot(mov_avg(:,2));
hold on;
plot(bool(:,2), 'r');
```

```
hold off;
title(['Moving average BPF & Bias corrected X-axis for file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(mov_avg(:,3));
hold on;
plot(bool(:,3), 'r');
hold off;
title(['Moving average BPF & Bias corrected Y-axis of file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);

%% Remove spurious activity candidates

% Any activity candidate that's shorter than 'short' samples
% should be ignored as we can not possibly have an activty
% for just 'short' samples after expanding initial candidates
integral = 0;
st = 0;
fin = 0;
short = 20;

for axis=2:3
   for a=1:in_size(1,1)
      if (d_bool(a,axis) ~= 0)
         integral = integral + d_bool(a,axis);
         if (integral ~= 0)
            st = a;
         else
            fin = a;
            if ( (fin-st) <= short)
               for ind=st:fin
                  bool(ind,axis) = bool(st-1,axis);
               end
            end
         end
      end
   end
end

figure();
subplot(2,1,1);
plot(mov_avg(:,2));
hold on;
plot(bool(:,2), 'r');
hold off;
title(['Moving average BPF & Bias corrected X-axis for file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(mov_avg(:,3));
```

```
hold on;
plot(bool(:,3), 'r');
hold off;
title(['Moving average BPF & Bias corrected Y-axis of file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);

%% Remove the effects of cross-axis sensitivity

% We know that at any point of time only one axis
% will be active. So, if there is overlap of activity
% then the axis with largest signal in that interval
% will prevail and other axis should be ignored

overlap = [0 0];
bool_sum = zeros(in_size(1,1));

for a=1:in_size(1,1)
   bool_sum(a) = abs(bool(a,2)) + abs(bool(a,3));
end

for a=1:in_size(1,1)
   if (bool_sum(a) == 2 & bool_sum(a-1) ~= 2)
      ovlp_st = a;
   end
   if (bool_sum(a) == 2 & bool_sum(a+1) ~= 2)
      ovlp_fi = a;
      overlap = [overlap; ovlp_st ovlp_fi];
   end
end

ovlp_vals = size(overlap);
if (ovlp_vals(1,1) > 1)
   for a=2:ovlp_vals(1,1)
      x_max = max(abs(mov_avg(overlap(a,1):overlap(a,2),2)));
      y_max = max(abs(mov_avg(overlap(a,1):overlap(a,2),3)));
      if (x_max > y_max)
         b = overlap(a,1) - 1;
         while (bool(b,3) ~=0)
            bool(b,3) = 0;
            b = b-1;
         end
         for b=overlap(a,1):overlap(a,2)
            bool(b,3) = 0;
         end
         b = overlap(a,2)+1;
         while (bool(b,3) ~= 0)
            bool(b,3) = 0;
            b = b+1;
         end
      end
      if (y_max > x_max)
         b = overlap(a,1) - 1;
         while (bool(b,2) ~=0)
```

```
            bool(b,2) = 0;
            b = b-1;
        end
        for b=overlap(a,1):overlap(a,2)
            bool(b,2) = 0;
        end
        b = overlap(a,2)+1;
        while (bool(b,2) ~= 0)
            bool(b,2) = 0;
            b = b+1;
        end
    end
end

figure();
subplot(2,1,1);
plot(mov_avg(:,2));
hold on;
plot(bool(:,2), 'r');
% plot(d_bool(:,2), 'k');
hold off;
title(['Moving average BPF & Bias corrected X-axis for file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(mov_avg(:,3));
hold on;
plot(bool(:,3), 'r');
% plot(d_bool(:,3), 'k');
hold off;
title(['Moving average BPF & Bias corrected Y-axis of file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);

%% Merge +ve and -ve half cycle of acceleration

rest_period = 10;
for axis=2:3
    for a=2:in_size(1,1)
        if (bool(a,axis) == 0 && bool(a-1,axis) ~= 0)
            if (sum(abs(bool(a:a+rest_period,axis))) ~= 0)
                ind = a;
                while (bool(ind,axis) == 0)
                    bool(ind,axis) = bool(a-1,axis);
                    ind = ind + 1;
                end
                a = ind - 1;
            end
        end
    end
end
```

```
figure();
subplot(2,1,1);
plot(mov_avg(:,2));
hold on;
plot(bool(:,2), 'r');
% plot(d_bool(:,2), 'k');
hold off;
title(['Moving average BPF & Bias corrected X-axis for file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(mov_avg(:,3));
hold on;
plot(bool(:,3), 'r');
% plot(d_bool(:,3), 'k');
hold off;
title(['Moving average BPF & Bias corrected Y-axis of file \', s1, ' and Window =
',num2str(2*half_win+1)], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);

figure();
subplot(2,1,1);
plot(data_in(:,2));
hold on;
plot(bool(:,2), 'r');
% plot(d_bool(:,2), 'k');
hold off;
title(['Band pass filtered X-axis for file \', s1], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);
subplot(2,1,2);
plot(data_in(:,3));
hold on;
plot(bool(:,3), 'r');
% plot(d_bool(:,3), 'k');
hold off;
title(['Band pass filtered Y-axis of file \', s1], 'FontSize', 14);
xlabel('Sample', 'FontSize', 12);
ylabel('m/s^2', 'FontSize', 12);

%% Create an array with each activity's details

activity = [0 0 0 0];

for a=2:in_size(1,1)
   if (bool(a,2) ~= 0 && bool(a,3) == 0)
      if (bool(a,2) ~= 0 && bool(a-1,2) == 0)
         act_st = a;
         x_stat = 1;
         y_stat = 0;
      end
      if (bool(a,2) ~= 0 && bool(a+1,2) == 0)
         act_en = a;
```

```
            activity = [activity; act_st act_en x_stat y_stat];
        end
    end
    if (bool(a,3) ~= 0 && bool(a,2) == 0)
        if (bool(a,3) ~= 0 && bool(a-1,3) == 0)
            act_st = a;
            x_stat = 0;
            y_stat = 1;
        end
        if (bool(a,3) ~= 0 && bool(a+1,3) == 0)
            act_en = a;
            activity = [activity; act_st act_en x_stat y_stat];
        end
    end
end


%% Kalman Filter setup

F0 = [1  0   T   0   0   0;
      0  1   0   T   0   0;
      0  0   1   0   T   0;
      0  0   0   1   0   T;
      0  0   0   0   1   0;
      0  0   0   0   0   1];

H = [0  0   0   0   1   0;
     0  0   0   0   0   1];

%% Kalman filter to track the motion

Q_scale = 0.01;
Q_val = 0.1;
F = F0;
sigy_scale = 0.3;
max_dist = 1.5;

init_pos = [0 0];

dist = [0 0];

act_size = size(activity);

if (buf)
    headroom = half_win;
else
    headroom = 0;
end

if (mov_avg_d)
    din = mov_avg;
else
    din = data_in;
end
```

```
for k=2:act_size(1,1)
   KF_in = din(activity(k,1) - headroom:activity(k,2) + headroom,:);
   length(KF_in)

   act_avg = mean(KF_in(:,2:3))
   if (bias_corr)
      KF_in(:,2) = KF_in(:,2) - act_avg(1);
      KF_in(:,3) = KF_in(:,3) - act_avg(2);
   end

   z = zeros(2,length(KF_in));

   trackX = activity(k,3);
   if (trackX)
      z(1,:) = KF_in(:,2)';
   end

   trackY = activity(k,4);
   if (trackY)
      z(2,:) = KF_in(:,3)';
   end

   % Time till which Kalman filter will be ON
   t_end = length(KF_in);

   Q = Q_scale*diag([0.0, 0.0, 0.0, 0.0, Q_val, Q_val]);

   stdev = 2*tick*g;
   sigy = stdev;
   sigy1 = sigy_scale*sigy;
   R = diag([sigy1*sigy1, sigy1*sigy1]);

   X0 = zeros(6,1);
   X0(5,1) = init_pos(k-1,1);
   X0(6,1) = init_pos(k-1,2);

   P0 = 1*diag([0.1, 0.1, 0.1, 0.1, 0.1, 0.1]);%Q;

   X_pri = zeros(6,t_end);
   y     = zeros(2,t_end);
   X_pos = zeros(6,t_end);

   iter=1;

   X_pri(:,i) = F*X0;
   P_pri      = F*P0*F' + Q;
   y(:,i)     = z(:,i) - H*X_pri(:,i);
   S          = H*P_pri*H' + R;
   K          = P_pri*H'*inv(S);
   X_pos(:,i) = X_pri(:,i) + K*y(:,i);
   P_pos      = ( eye(6) - K*H )*P_pri;
```

```
    for iter = 2:t_end
        X_pri(:,iter) = F*X_pos(:,iter-1);
        P_pri         = F*P_pos*F' + Q;
        y(:,iter)     = z(:,iter) - H*X_pri(:,iter);
        S             = H*P_pri*H' + R;
        K             = P_pri*H'*inv(S);
        X_pos(:,iter) = X_pri(:,iter) + K*y(:,iter);
        P_pos         = ( eye(6) - K*H )*P_pri;
    end

    init_pos = [init_pos; trackX*X_pos(1,t_end)+init_pos(k-1,1) trackY*X_pos(2,t_end)+init_pos(k-1,2)];
    dist = [dist; X_pos(1,:)'+ init_pos(k-1,1) X_pos(2,:)'+ init_pos(k-1,2)];
end

figure();
plot(init_pos(:,1), init_pos(:,2));
title(['Final Track generated for file \', s1], 'FontSize', 14);
xlabel('Distance (m)', 'FontSize', 12);
ylabel('Distance (m)', 'FontSize', 12);
xlim([-max_dist max_dist]);
ylim([-max_dist max_dist]);
```

Appendix B

Supplementary Results

B.1 Chapter 5

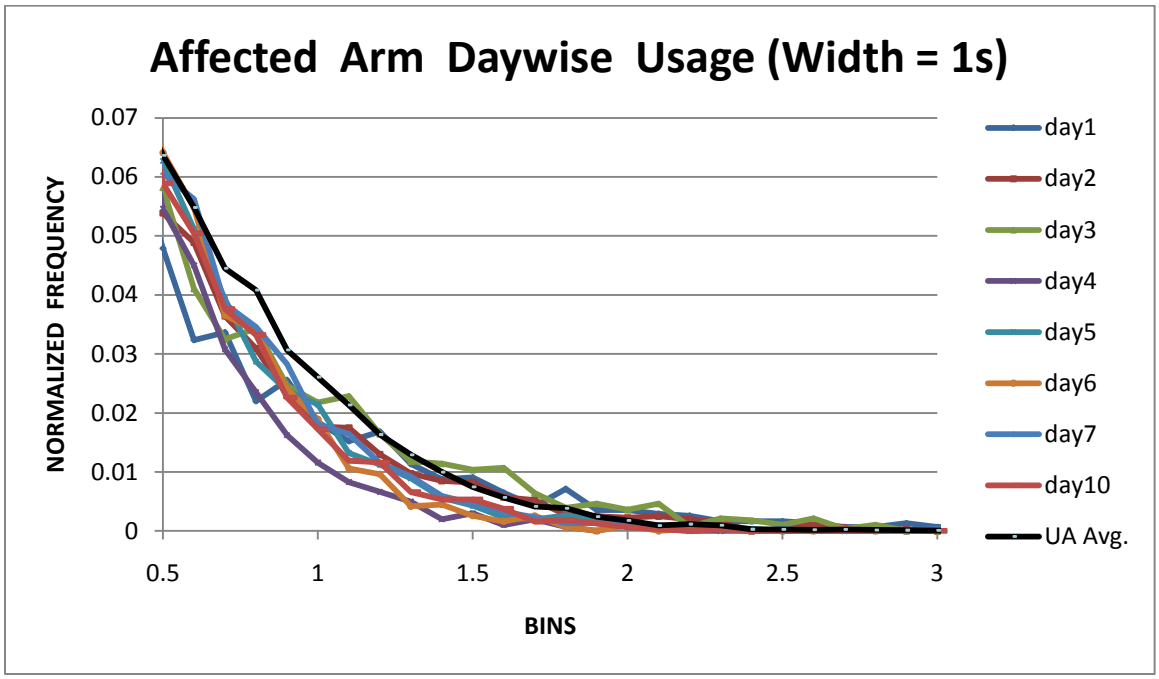

Fig. B.1 Normalized Histogram of Distance Covered with width = 1*s* for Subject 2

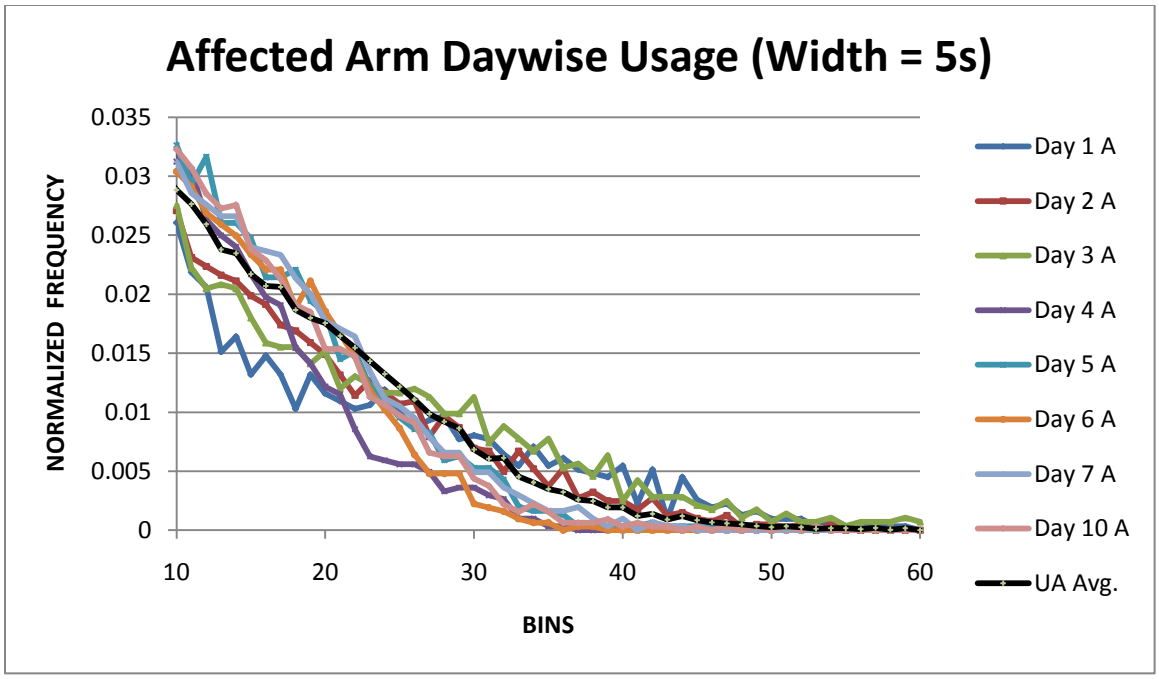

Fig. B.2 Normalized Histogram of Distance Covered with width = 5*s* for Subject 2

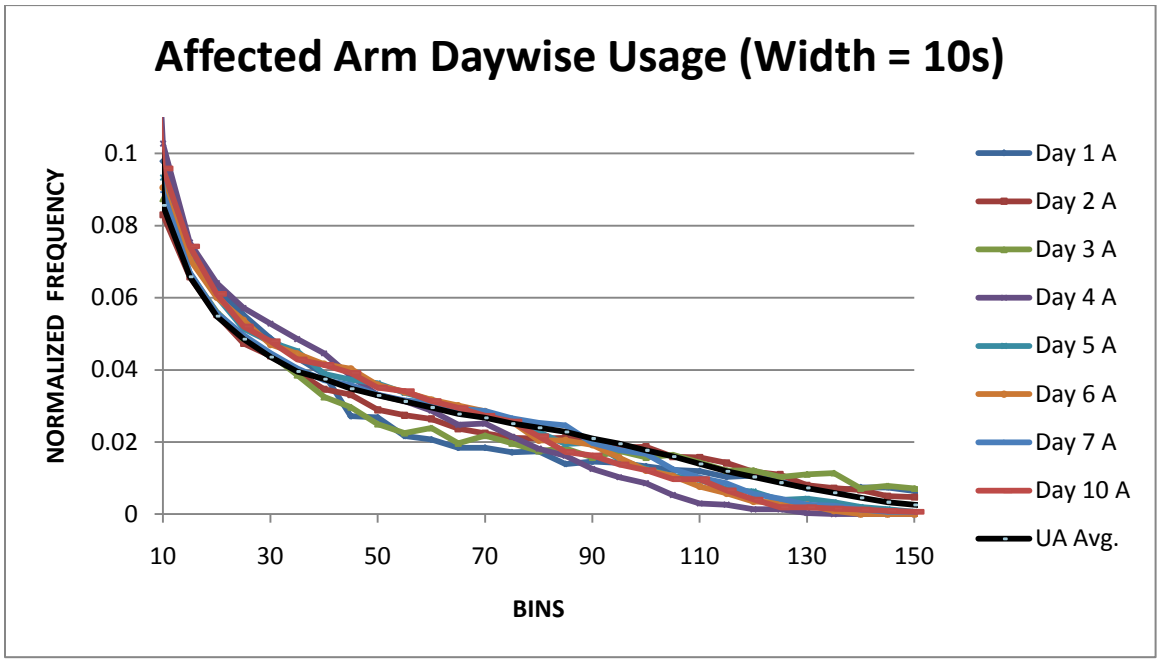

Fig. B.3 Normalized Histogram of Distance Covered with width = 10*s* for Subject 2

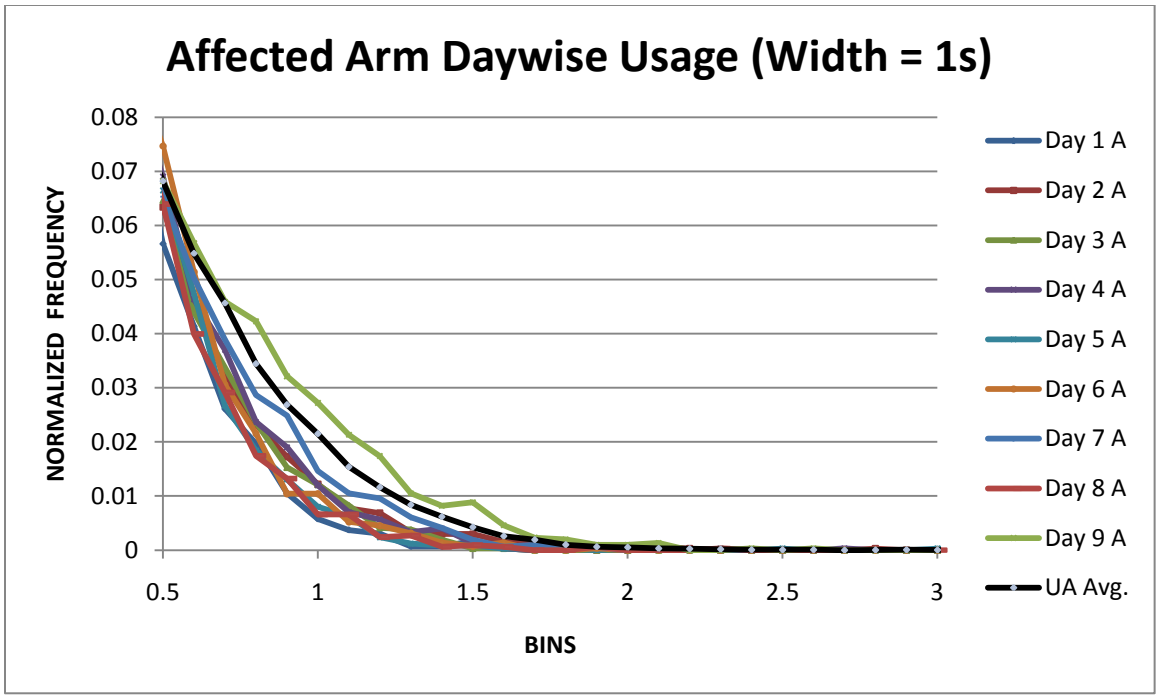

Fig. B.4 Normalized Histogram of Distance Covered with width = 1*s* for Subject 3

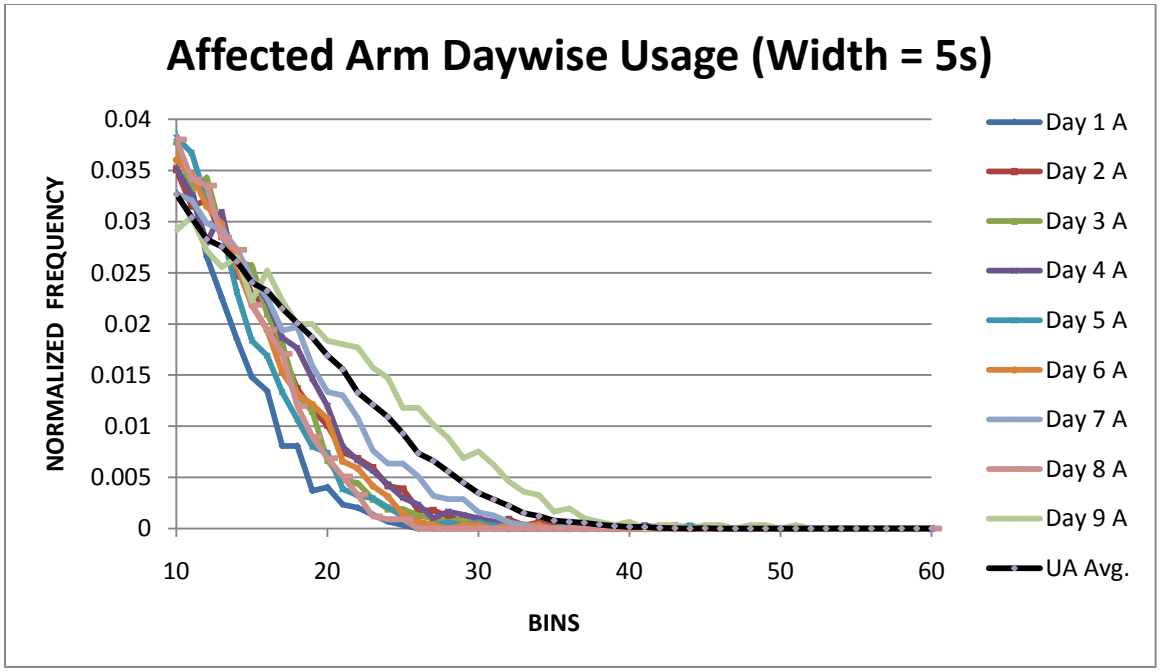

Fig. B.5 Normalized Histogram of Distance Covered with width = 5*s* for Subject 3

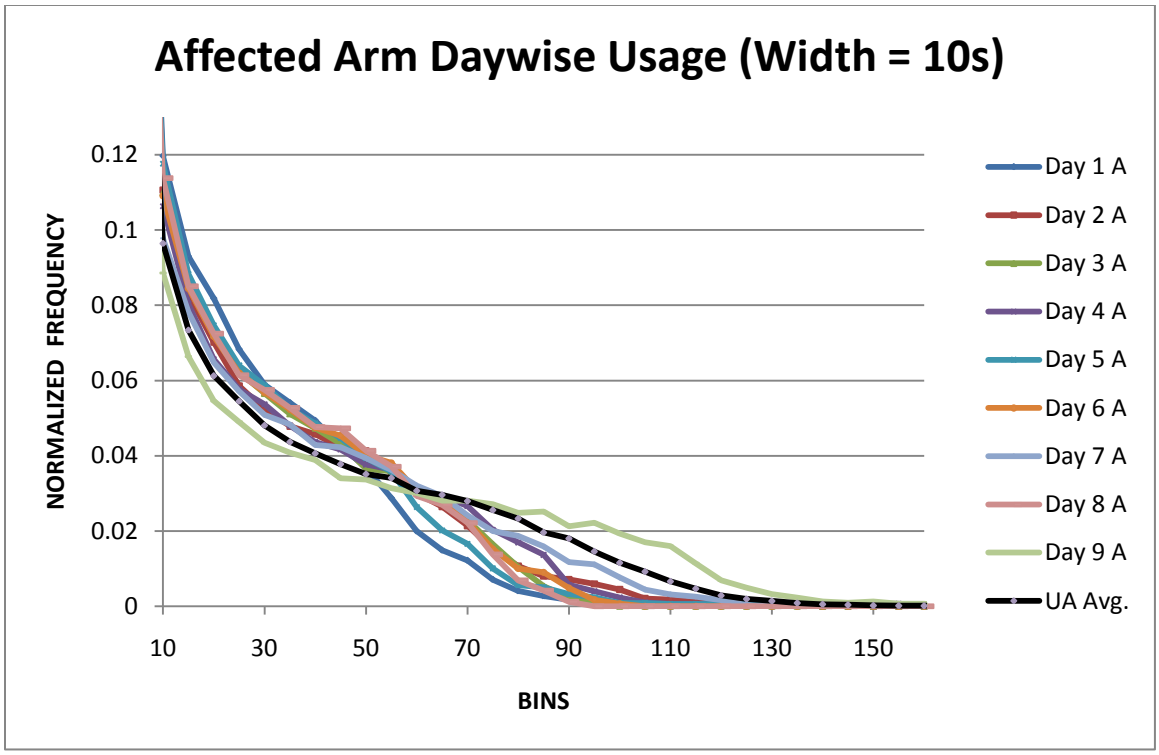

Fig. B.6 Normalized Histogram of Distance Covered with width = 10s for Subject 3

B.2 Chapter 6

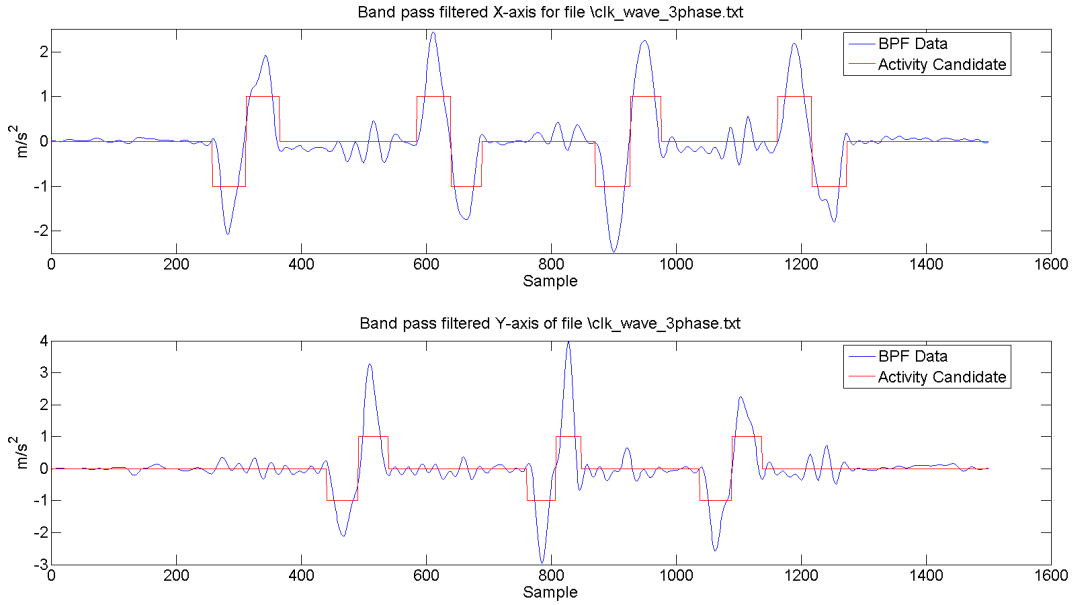

Fig. B.7 Identified activity durations (Red) with band-pass filter output (Blue) for X-axis (Top) and Y-axis (Bottom)

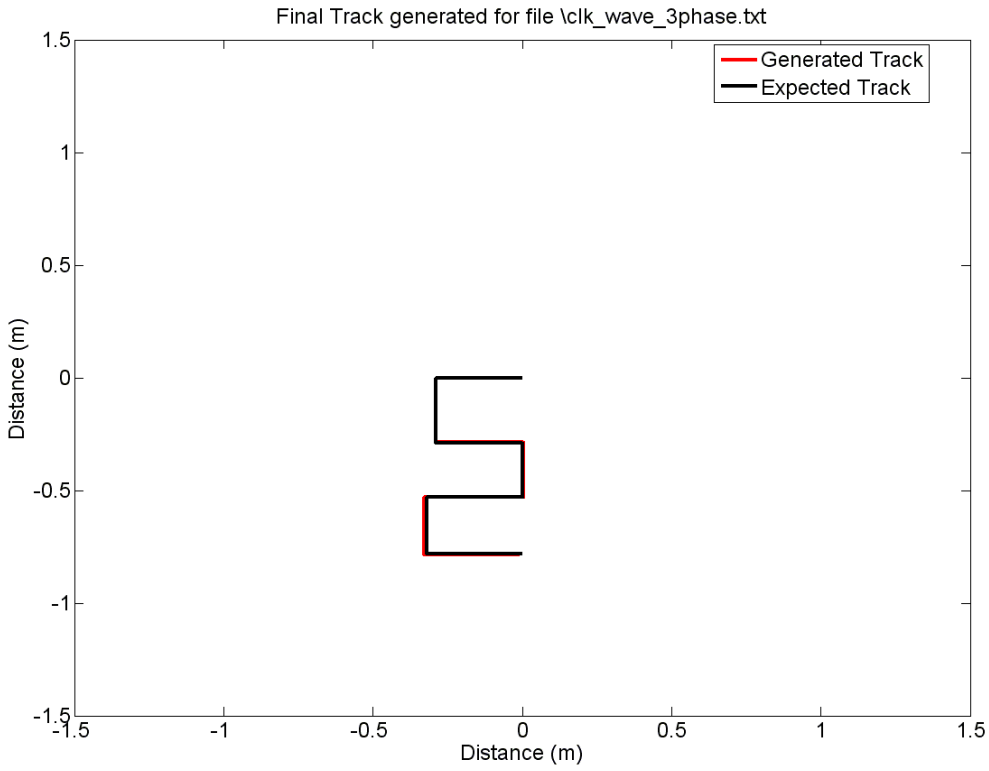

Fig. B.8 Final track generated by tracking seven 1D motions in 2D

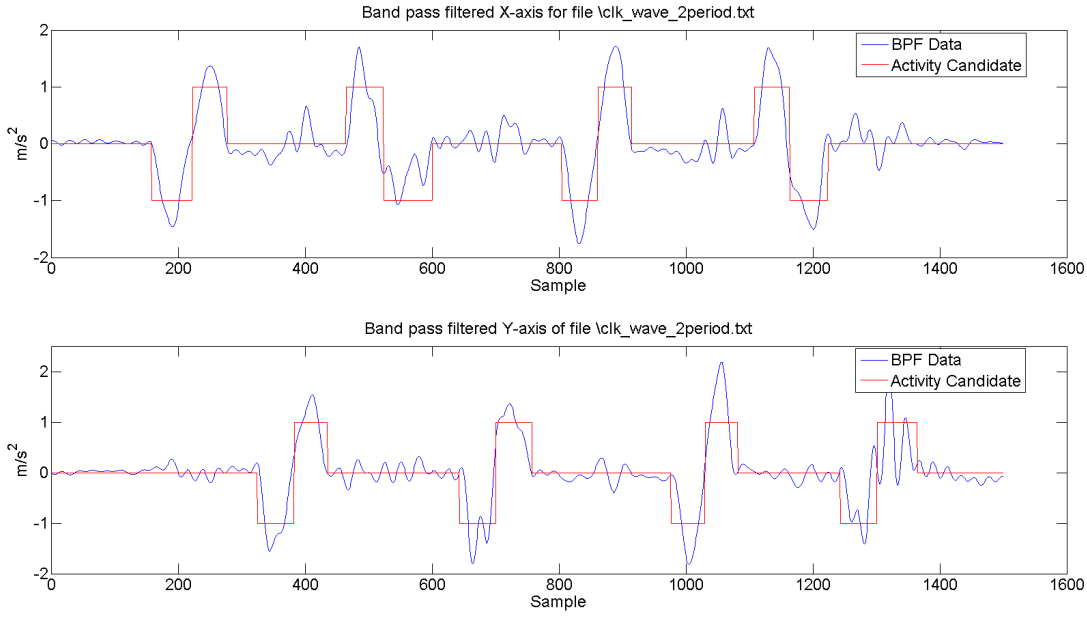

Fig. B.9 Identified activity durations (Red) with band-pass filter output (Blue) for X-axis (Top) and Y-axis (Bottom)

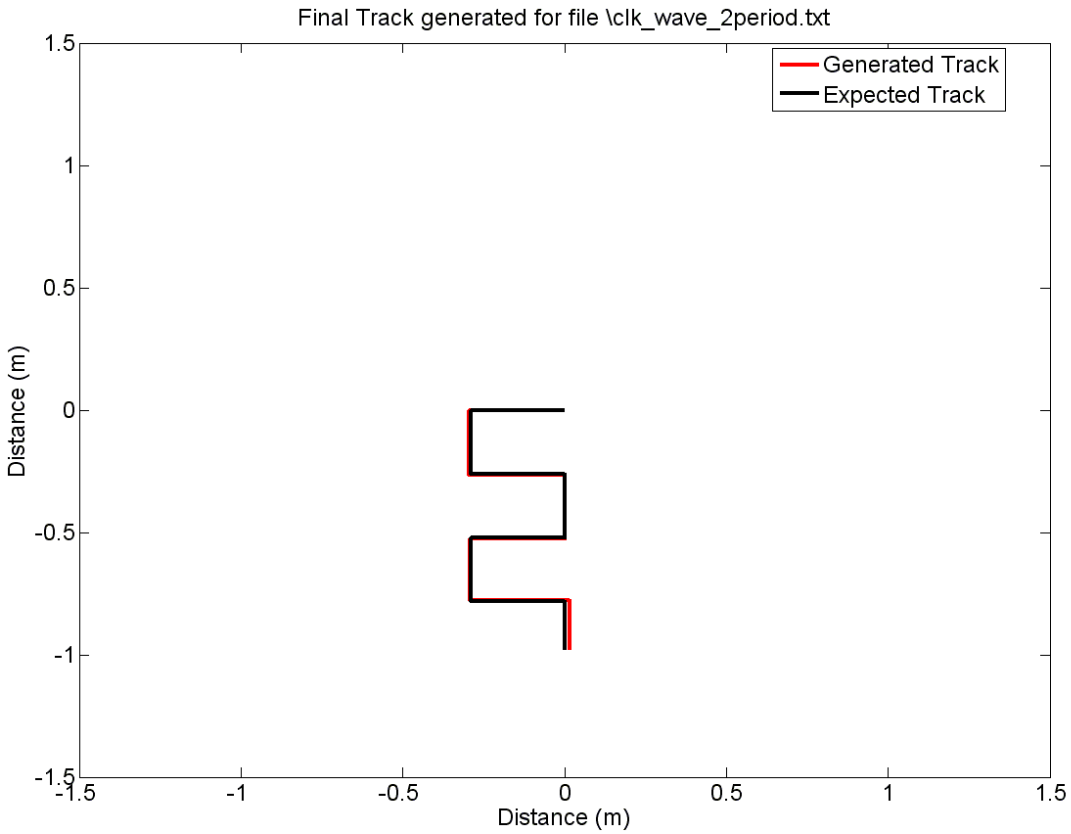

Fig. B.10 Final track generated by tracking eight 1D motions in 2D

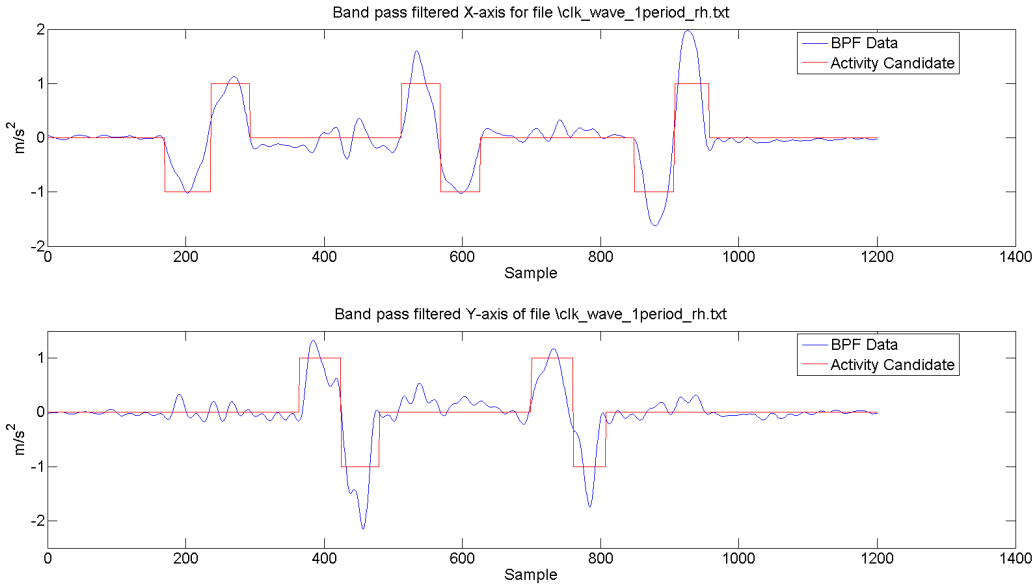

Fig. B.11 Identified activity durations (Red) with band-pass filter output (Blue) for X-axis (Top) and Y-axis (Bottom)

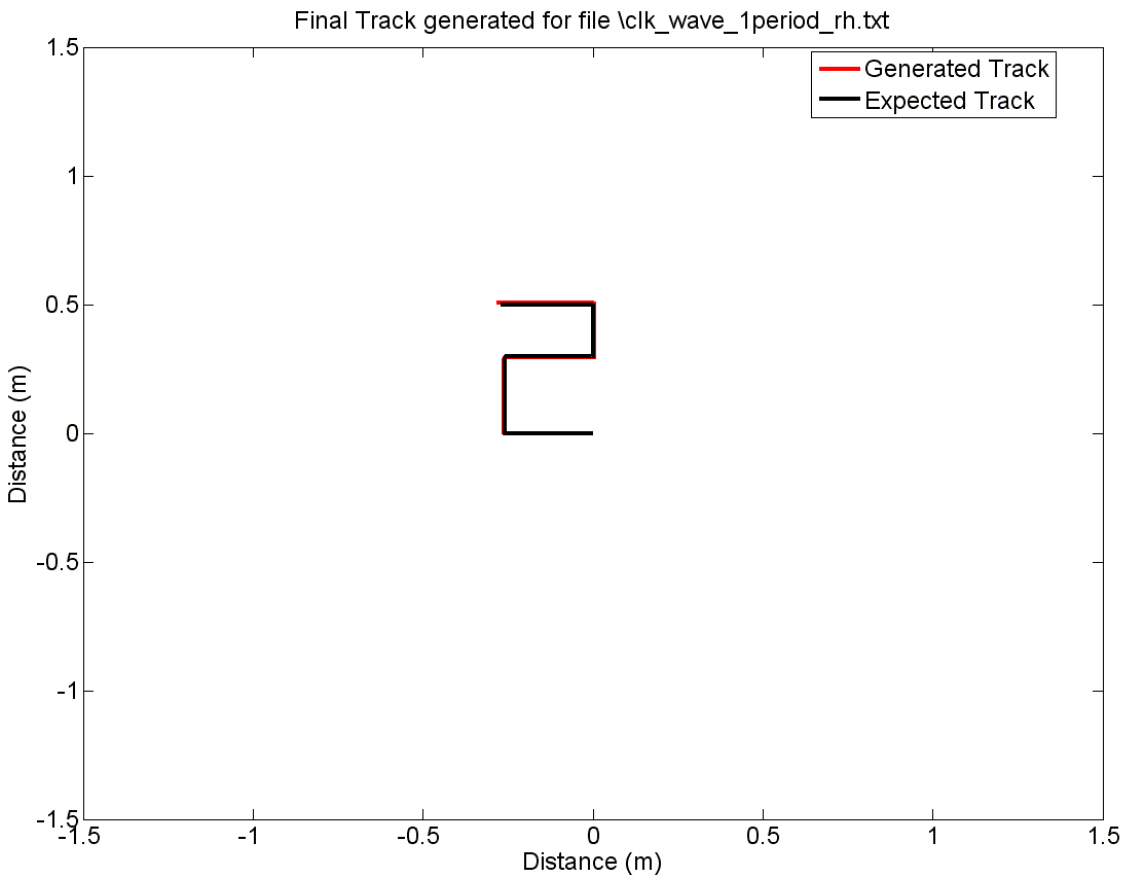

Fig. B.12 Final track generated by tracking five 1D motions in 2D

ABBREVIATIONS

AAUT	Actual Amount of Use Test
ADC	Analog to Digital Converter
ADL	Activities of Daily Living (Life)
AR	Autoregressive process
BBT	Box and Block Test
CIMT	Constraint Induced Movement Therapy
FAS	Functional Ability Scale
IADL	Instrumental Activities of Daily Living (Life)
IMU	Inertial Measurement Unit
MAL	Motor Activity Log
MEMS	Micro-electro mechanical system
MIS	MEMS Inertial Sensors
NRRL	NeuroRehabilitation Research Laboratory
SAW	Surface Acoustic Wave
SIS	Stroke Impact Scale