UNIVERSITY OF CALIFORNIA

Los Angeles

Inference of Human Motion using Low-cost Sensors

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Electrical Engineering

by

Chieh Chien

2013

ABSTRACT OF THE DISSERTATION

Inference of Human Motion using Low-cost Sensors

By

Chieh Chien

Doctor of Philosophy in Electrical Engineering

University of California, Los Angeles, 2013

Professor Gregory J. Pottie, Chair

A wireless health system that collects and processes data of human activities can help both users and medical professionals to monitor health status remotely. Therefore it saves tremendous medical resources and costs compared to traditional treatment in which a huge amount of human effort is involved. We present two systems that can correctly classify human daily life activities with little training, and another system to reconstruct human motion trajectories from commercial low cost MEMS inertial measurement units (IMUs) and the Microsoft® Kinect.

A system that reliably classifies daily life activities can contribute to more effective and economical treatments for patients with chronic conditions or undergoing rehabilitative therapy. We propose a universal hybrid decision tree classifier for this purpose. The tree classifier can flexibly implement different decision rules at its internal nodes, and can be adapted from a population-based model when supplemented by training data for individuals. Compared to other methods, the experimental results showed a high accuracy of classifying human daily live activities.

After we have an accurate classification of human activities, we present a system to further reconstruct motion trajectories using IMUs and the Kinect. The system fuses different motion reconstruction models to give a better tracking result, in which each model is weighted and transformed to a universal basis. This model is also expandable to accommodate different resources and environments. Experimental results showed a great improvement over past methods only using a single motion reconstruction scheme.

The dissertation of Chieh Chien is approved.

William Kaiser

Lara Doecek

Mario Gerla

Gregory J. Pottie, Committee Chair

University of California, Los Angeles

2013

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First I would like to give my deepest thanks to my advisor Prof. Gregory J. Pottie as being the nicest advisor in my life. He always gave me full supports and allowed me to do any kinds of strange experiments as long as I thought they benefited our research. Prof. Pottie seldom pushed me for speeding up on research or writing up more papers. While this means that I always have to supervise myself and self-examine, this also means that Prof. Pottie gave his full trust in me in my research. I think the lesson he wanted to teach me was not only "By knowing how to research independently then you earn your M.S. degree, and by knowing how to lead a research project and master that specific field, which even I cannot fully understand, then you qualify as a Ph.D.," but also how to be self-responsible, always learn actively, self-examine and think independently. To me, Prof. Pottie is not only my academic advisor, but also like my friend in life.

I would like to thank Prof. Bill Kaiser, Prof. Lara Dolecek, and Prof. Mario Gerla who served as my committee in Ph.D. defense and qualifying examination, with their accommodation in scheduling the defense and qualifying examination, and with their challenging questions and kindly help during these exams.

I was lucky enough to have some awesome lab mates when in UCLA graduate school. First I would like to thank my first and only senior lab mate Nabil Hajj Chehade who inspired me and gave lots of valuable knowledge and priceless experiences in doing researches, launching experiments, and mentoring internship students. I would also like to thank James Xu who provided tons of technical support and programming experiences. Also, I would like to thank Yan Wang who always shared the research ideas through ferocious arguing with each other while remaining friends. I would like to thank Hua-I Chang who also helped me on programming issues and companionship when only he and I worked together during late 2011 and early 2012. Finally I would like to thank Xiaoxu Wu who helped me in collecting data and mentoring the internship students.

I would also like to thank many of my mentored students. First I would like to acknowledge the CENS summer internship program held by Wes Uehara, for which I supervised 10 students in summer 2010 and 2011. They are Natali Ruchanski, Claire Lochner, Elizabeth Do, Tremaine Rawls, Benjamin Fish, Ammar Khan, Pinar Ozirik, James Gomes, Travis Rodriguez, and Ascher Friedman. I would like to thank another 3 internship students of Prof. Pottie's research seminar (class EE199). They are Jingtao Xia, Oscar Santana, and Debbie Tray. These internship students helped me collecting data, recruited their friends for more data, and implemented algorithms on their own under my guidance. During this time not only did I mentor them with a series of discussions and experiments, but their creative ideas and thoughts also inspired me a lot.

Then I would like to acknowledge the reprint of copyrighted material and thank my co-authors and the ones who helped me for the research.

Chapter 2 is in part a reprint of "A universal hybrid decision tree classifier design for human activity classification," in Engineering in Medicine and Biology Society (EMBC), 2*012 Annual International Conference of the IEEE*, "Monitoring Workspace Activities Using Accelerometers." *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2011*, "Estimation of Accelerometer Orientation for Activity Recognition" *IEEE Engineering In Medicine & Biology Society 34th Annual International Conference, 2012*, and "Feature Selection based on Mutual Information for Human Activity Recognition." *IEEE International Conference* *on Acoustics, Speech, and Signal Processing, 2012*. I would like to thank the following co-authors who helped with this project: Prof. Bill Kaiser, Nabil Hajj Chehade, Natali Ruchanski, Claire Lochner, Elizabeth Do, Tremaine Rawls, Benjamin Fish, Ammar Khan, Ascher Friedman and Prof. Gregory J. Pottie. I would like to thank the following lab mates and students for data collection: James Xu, Pinar Ozirik, James Gomes, and Travis Rodriguez.

Chapter 3 is in part a reprint "Non-linear Complementary Filter based Upper Limb Motion Tracking using Wearable Sensors" *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2013*. I would like to thank the following co-authors with this project: Jingtao Xia, Oscar Santana, Yan Wang and Prof. Gregory J. Pottie. I would also like to thank the following lab mates and students for data collection: James Xu, Hua-I Chang, Xiaoxu Wu, and Debbie Tray.

I would like to dedicate this dissertation to my family. To Chung-ming Chien, my father, he took care me ever since I was a little boy. He fed me, washed my clothes, took care of my room, supervised my homework, and more importantly, he gave a whole love to the family so we can be united as one. To Li-chin Zhou, my mother, she provided me full mental and financial support so that I can study here in UCLA. She always encouraged me to study abroad to broaden my horizons, though she knew it was a tough decision since the family will be separated apart. I would like to thank my brother I-zen Chien who is my pillar of strength. I also would like to thank my grandmother and aunts who flew to the USA to attend my graduation commencement especially for me. Finally, I would like to thank all other members of my family who always gave me warm support when I studied abroad.

I would also like to thank all my friends here in Los Angeles. I could not have completed my degree when being alone without them. Thank you Chung-Yu Lou, Chung-Tse Michael Wu, Wen-Sao Wilson Hong, Chi-Wen Stasia Su, Chris Hsu, Angie Liao, Frank Wang, Meng-Ning Monica Cheng, Sheng-Wen Wayne Chang, Jim Sun, Mars Lan, Peiyun Lu, James Chung, Wenjae Joanne Chang, Fang-Li Justin Yuan, and Sabrina Saykham. I hope I did not forget anyone, and if I did, please accept my full apology.

A special acknowledge goes to all my college friends in my hometown Taiwan. To Hisn-Yao Lin, Hong-Yi Jack Lu, Tzu-Chang Lin, Chung-Hsiu Ko, Chi-Kuang Luke Chen, Meng-Han Andy Tang, Yen-Yu Andrew Lee, Po-Chun Chen, Tony Chun-Ming Lo, Kung-Yi Li, Chien-Wei Chen, Tsung-Han Ryan Wu, Yuan-Heng Chien, Wei-Kang Ricky Hsu, Yu-Jung Karl Chen, Albert Yu-Ying Lee , Hsu-Kuang Chiu, Chun-Te Randy Chu, Jeffery Lee, Chung-Kai Yu, Bernie Jord Yang, Pei-Yin Lin, Jian-Hung Liu, Sheng-Yung Chen, Yen-Tien Lu, Wen-Yang Ku, Tao Todd Ben, Meng-Chao Boris Peng, Sheng-Hui Derek Lu, Shu-Sheng Huang, Min-Chun Jenny Shih, Lene Yung Ling Cheng, Wan-Ling Annie Tsai, Li-Wen Eva Huang, Elaine Tung, Yun-Chun Christine Hua, and Hsin-Yin Annie Peng. Thank you for all your support ever since I met you in different time of my life.

Last but not least, I would like to give a special thank to my girlfriend Yamin Maxine Lu. Thank you for always support me and take care of tons of stuff in our lives, so that I can focus myself solely on research. Without you I could not have been graduated so soon. Thank you for all the happy memories that we have been through for the past 3 years, and hope we can be together longer and longer.

VITA

Mar, 2010 – Oct, 2013	Research Assistant, Electrical Engineering, University of California, Los Angeles
Mar, 2010 – Oct, 2013	M.S., Electrical Engineering University of California, Los Angeles
Jan, 2008 – Jul, 2008	Research Assistant Institute of Biomedical Engineering, National Taiwan University
Sep, 2002 – Jun, 2006	B.S., Electrical Engineering National Taiwan University

PUBLICATIONS

[J1] J. Y. Xu, H-I Chang, C. Chien, W. Kaiser, and G. J. Pottie, "Context-driven, Prescription based Personal Activity Classification: Methodology, Architecture and End-to-End Implementation", *IEEE Journal of Biomedical and Health Informatics, 2014*

[C1] H-I Chang, C. Chien, J. Y. Xu, and Greg J. Pottie, "Context-guided Universal Hybrid Decision Tree for Activity Classification", *Wearable and Implantable Body Sensor Networks, 2013*

[C2] X. Wu, Y. Wang, C. Chien, and G. J. Pottie, "Self-Calibration of Sensor Misplacement Based on Motion Signatures" *Wearable and Implantable Body Sensor Networks, 2013*

[C3] C. Chien, J. Xia, O. Santana, Y. Wang, and G. J. Pottie, "Non-linear Complementary Filter based Upper Limb Motion Tracking using Wearable Sensors" *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2013*

[C4] Y. Wang, C. Chien, J. Xu, G. J. Pottie, and W. Kaiser, "Gait Analysis using 3D Motion Reconstruction with an Activity-specific Tracking Protocol" *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2013*

[C5] C. Chien, J. Y. Xu, H-I Chang, X. Wu, G. J. Pottie, "Model construction for Human Motion Classification using Inertial Sensors" in *Information Theory and Applications Workshop (ITA), 2013. IEEE, 2013.*

[C6] C. Chien, and G. Pottie, "A Universal Hybrid Decision Tree Classifier Design for Human Activity Classification" *IEEE Engineering In Medicine & Biology Society 34th Annual International Conference, 2012*

[C7] N. Hajj Chehade, A. Friedman, C. Chien, and G. Pottie, "Estimation of Accelerometer Orientation for Activity Recognition" *IEEE Engineering In Medicine & Biology Society 34th Annual International Conference, 2012*

[C8] B. Fish, A. Khan, N. Hajj Chehade, C. Chien, G. Pottie, "Feature Selection based on Mutual Information for Human Activity Recognition." *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2012.*

[C9] N. Ruchansky, E. Do, C. Lochner, T. Rawls, N. Hajj Chehade, J. Chien, G. Pottie, W. Kaiser, "Monitoring Workspace Activities Using Accelerometers." *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2011*

[C10] C. Chien, Y-H Chen, and L-G Chen, "Skip Control Algorithm of Motion Estimation for Power-scalable H.264 Video Encoder," *The 18th VLSI Design/CAD Symposium, 2007*

Chapter 1

INTRODUCTION AND PRELIMINARIES

1.1 Introduction

Classifying human activities is important for many medical applications. Consider a patient whose arm is injured and who is in the rehabilitation process. The patient will preferentially use the healthy arm since it is more convenient, while doctors will prescribe exercising the injured arm and using it in ordinary daily activities. Traditionally this was done in medical clinics where health care professionals can supervise patients' rehabilitation progress. However, with a system of activity classification and monitoring, professionals can monitor this progress when patients stay at home. This saves considerable medical resources since rehabilitation often takes a very long time.

Gathering statistics on people's daily activities through automatic monitoring systems also has application to wellness. According to the World Health Organization (WHO), at least 60% of the global population fails to achieve the minimum recommendation of 30 minutes of moderate intensity physical activity daily[6]. A habit of daily activity strongly protects against many chronic diseases. Yet it has heretofore been difficult to cheaply and reliably record such activities, and provide useful feedback to both individuals and health care providers. Therefore, increased research effort is going into the creation of systems that record human motions with feasible cost (manufacture cost, power, storage and communication resources), classify activities with good accuracy, and then analyze these activities with respect to different rules that lead to a better life [1][2][3][4].

Other than activity classification, it is also important that we can track human motions in 3D space at any time. This helps us observe and monitor the human motions at a more detailed scale. For instance, for patients with Parkinson's disease, we can monitor patients' sickness by recording the resting tremors continuously with a motion reconstruction system; for injured patients who are in their rehabilitation states, we can know how well they perform day by day if there exists a system telling us how high they can lift their arms, or the lengths of their steps. Other applications of the motion reconstruction system include gait analysis, remote health monitoring, health care analysis, etc.

For motion tracking using low-cost commercial micro-electromechanical systems (MEMS) such as accelerometers, gyros and magnetometers, we usually failed to acquire accurate results due to various kinds of imperfections due to the nature of MEMS sensors. These imperfections include bias and noise from sensor measurements, misalignment between coordinates of the sensors, magnetic field interference caused by environments [5], etc. Therefore, when one tries to reconstruct trajectories using numerical integration, integration error from the noise will accumulate. What is worse, the miscalculation of object orientation when estimating trajectories has a large effect since one has to know the orientation of the object in order to cancel the gravity to integrate the acceleration.

The rest of the thesis is organized as follows. In this chapter, we will give some preliminary knowledge that is necessary for the thesis. We provide our system for activity classification in Chapter 2. Chapter 3 will discuss the coordinate fusion used for human motion tracking and reconstruction. In Chapter 4 we present our conclusions and suggestions for future research.

This chapter introduces some basic knowledge regarding to activity classification and motion tracking, and also the devices used in this research. For activity classification, we will introduce some machine learning techniques that are generally used, including the tree classifier (1.2.1), naïve Bayes classifier (1.2.2), and support vector machine (1.2.3). For motion tracking, we will introduce how to use complementary filters to find the orientation of an object given we have MEMS measurements (1.3), and how we decompose human motions using biomechanical models (1.3.2). Finally, we introduce the devices used in this research, including MEMS inertial sensors (1.4.1) and the Microsoft® Kinect (1.4.2).

1.2 Activity Classification

In this section, we introduce several classical machine learning methods and techniques that classify activities. They are well defined and found useful when it comes to activity classification [6][7][8][10][10][11][12][13][14][15][16][17][18][19][20]. However, each of these methods has drawbacks for our purpose of classifying activities, when the set of activities to be classified should be expandable easily with a small amount of training data.

A classifier is a function that maps the calculated feature vectors into classes. Suppose we have collected a set of data consisting of n observations, each observation has p features, and there is one label out of q classes associated with it. The classifier f can be thought of as a mapping function that maps the training data to the classification or partition

$$f : TD_i \to \hat{y}_i \tag{1.1}$$

where $\hat{y}_i$ is the classification of the i^{th} set of data, TD_i is the i^{th} set of training data of n points of the form

$$TD = \left\{ \left(\mathbf{x}_i, y_i \right) \middle| \mathbf{x}_i \in \mathbb{R}^p, y_i \in \left\{ 1, 2, \cdots, q \right\} \right\}, i = 1, 2, \cdots, n \tag{1.2}$$

where $\mathbf{x}_i$ is the feature vector, and the y_i are the labeled classes associated with the features.

1.2.1 Decision Tree Classifier

A decision tree classifier is a supervised machine learning technique, which breaks down the multiclass classification into simpler subsets. Because of its nature of divide and conquer of the decision-making procedure, the decision tree classifier avoids the curse of dimensionality in multivariate analysis [6][7]. The curse of dimensionality says when doing multiclass classification, as the number of classes increases, one usually has to select more features and makes decisions in a high-dimensional feature space. Therefore, in order to collect enough training data that is representative of the nature of each class, the amount of labeled ground truths grows enormously as the dimension of feature space increases, or the predictive power reduces as the dimensionality increases [8]. The decision tree classifier uses a conditional independence assumption to avoid this issue by performing many classifications targeting smaller classes instead of a single stage with a huge number of states. Thus each decision is done in a feature space with lower dimensionality.

Generally, a tree classifier T with l internal nodes can be depicted as shown in Figure 1.1. An internal node of a tree is a node that is not a leaf node. This tree classifier can be thought as a set of l single-stage classifiers, each with its subset of classes, features and the decision rules used for the node. Therefore the tree classifier can be written as

$$T = \left\{ C(t), F(t), D(t) \right\} \tag{1.3}$$

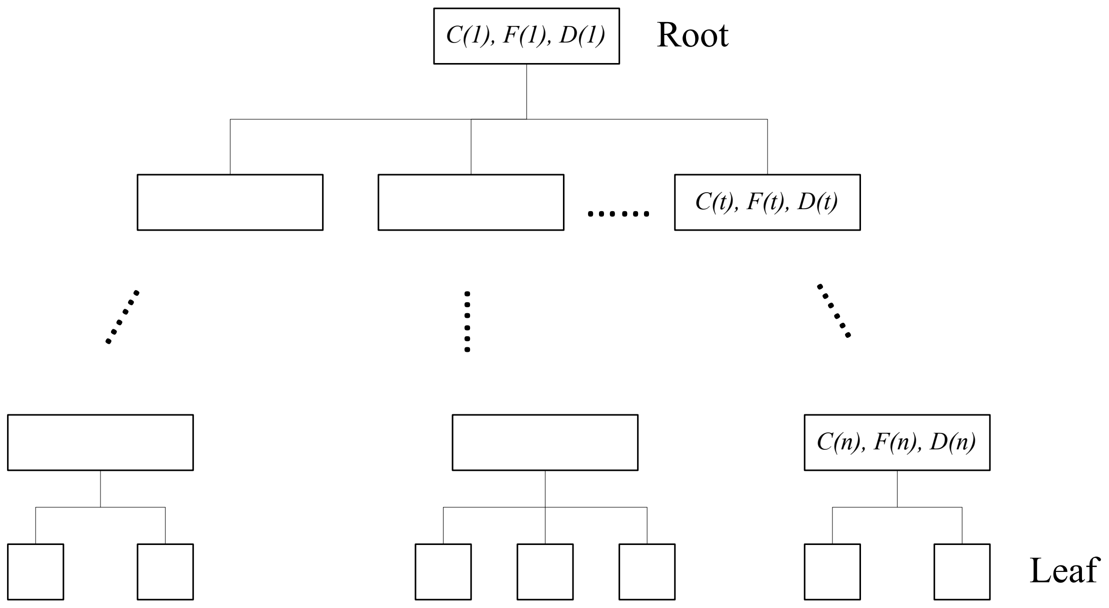


Figure 1.1 Decision tree classifier

where

$$C(T) \in \mathbb{C}, \mathbb{C} = \{c \mid c = \text{possible combination of classes}\} \tag{1.4}$$

is the subset of classes of node *t*, indicating how to group classes in that node; and

$$F(t) \in \mathbb{F}, \mathbb{F} = \{f \mid f = \text{possible combination of features}\} \tag{1.5}$$

is the subset of classes of node *t*, indicating how to group classes in that node; and

$$F(t) \in \mathbb{F}, \mathbb{F} = \{f \mid f = \text{possible combination of features}\} \tag{1.6}$$

is the feature set used for node t ; and

$$D(t) \in \mathbb{D}, \mathbb{D} = \{d \mid d \text{ is a decision rule}\} \tag{1.7}$$

is the decision rule of node t .

Forming a tree classifier consists of deciding upon $C(t)$, $F(t)$ and $D(t)$ for each internal node based on prior knowledge and observation of the training data.

According to [10], the optimal decision tree design T^* can be represented as the following optimization problem

$$T^* = \arg\min_{(C,F,D)} P_{\mathrm{e}}\left(C,F,D\right) \text{ s.t. Limited training size} \tag{1.8}$$

where $P_{\mathrm{e}}\left(C,F,D\right)$ is the overall probability of error associated with specific set of tree structure, features and decision rules selected. That is to say, we are looking for a combination of C, F and D that minimize the probability of error. Then following [11], the optimization problem can be broken down into two steps

$$\begin{gathered}\text{Step 1: For a given } C \text{ and } F\text{, find } D^* = D^*\left(C,F\right) \text{ such that} \\ P_{\mathrm{e}}\left(C,F,D^*\left(C,F\right)\right) = \min_{D} P_{\mathrm{e}}\left(C,F,D\right)\end{gathered} \tag{1.9}$$

$$\begin{gathered}\text{Step 2: Find } C^*, F^* \text{ such that} \\ P_{\mathrm{e}}\left(C^*,F^*,D^*\left(C^*,F^*\right)\right) = \min_{C,F} P_{\mathrm{e}}\left(C,F,D^*\left(C,F\right)\right)\end{gathered} \tag{1.10}$$

In most designs of decision tree classifiers [12][13], the selected decision rules D^* associated with the optimized tree T^* are fixed and the same for every node of the tree classifier. Commonly used rules are Gini index, twoing rule or maximum deviance reduction; some researchers also use naïve Bayes classifiers to separate classes in each node. However, in our research, in which we try to classify various activities with different characteristics, only using a single decision rule in making the decision tree classifier is limited. Therefore, a more flexible design is needed.

1.2.2 Naïve Bayes Classifier

Naïve Bayes classifiers [14][15][16] are yet another probabilistic classifier based on Bayes' theorem, with the assumption of conditional independence with respect to the input features in each class, and Gaussian kernels. Although these assumptions seem unrealistic, naïve Bayes classifiers very often work well in real-world situations, provided the features are carefully chosen. One major advantage of their use is that only a small amount of training data is needed to build class parameters. Another advantage is that only the variances of features belonging to each class are needed, obviating the time and resources required for computing the whole covariance matrix, which makes the classification procedure fast and efficient. Additionally, the advantage of using a probability-distribution-based classifier is that sometimes an instance may not be classified as one of the labeled classes, known as a reject option. This is important in some medical decision-making. In that case, there is not enough confidence in believing the answer, and so human effort is flagged for examining the data in more detail. This could of course be a deficiency in other applications.

The naïve Bayes classifier is a probabilistic classifier, which estimates the probability of each class y given the set of available feature vector $\mathbf{x}$, which we call the posterior probability $p\left(y\middle|\mathbf{x}\right)$ in this section. The classifier then selects the final class y^* such that the posterior probability is maximized.

$$y^*\left(\mathbf{x}\right) = \arg\max_y p\left(y\middle|\mathbf{x}\right) \tag{1.11}$$

Generally, the posterior probability is not easily acquired. Therefore, the classifier incorporates Bayes' theorem

$$p\left(y\middle|\mathbf{x}\right)=\frac{p\left(\mathbf{x}\middle|y\right)p\left(y\right)}{p\left(\mathbf{x}\right)} \tag{1.12}$$

In this formula, $p\left(\mathbf{x}\middle|y\right)$ is called the likelihood, or the conditional probability of the feature vector $\mathbf{x}$ given class y happened. The distribution of this conditional probability is estimated in the training phase. $p\left(y\right)$ is called the prior probability, which states our prior knowledge before estimating the model. $p\left(\mathbf{x}\right)$ is the evidence, or the normalization factor which makes the total summation of probabilities equal to 1.

The naïve Bayes classifier further assumes that the distribution of each feature value given the class is independent to simplify the computation (which states there is no correlation between feature values). Given this assumption, we calculate the likelihood as follows:

$$\begin{aligned} p\left(y\middle|\mathbf{x}\right) &\propto p\left(\mathbf{x}\middle|y\right)p\left(y\right) \\ &\propto p\left(y\right)\prod_i p\left(x_i\middle|y\right) \end{aligned} \tag{1.13}$$

Therefore, by calculating the distribution of each feature value given the class $p\left(x_i\middle|y\right)$ during the training phase we can estimate the probability of each class given the feature values acquired during testing.

The Naïve Bayes classifier, which proved useful in many fields including activity classification, has some drawbacks thus making it insufficient to our study. First, when building the statistics of features of some activities that are prone to noise, the noise might disturb the data so badly that the calculated statistics are not representative of the distributions of those activities. Therefore the trained distribution, and the naïve Bayes classifier models, cannot be used to classify activities in testing phase. Second, when one tries to classify more than 2 classes, the curse of dimensionality makes one to collect more ground truth in training phase, thus making the model not easily expandable.

1.2.3 Support Vector Machine

The support vector machine (SVM) [17][18][19] is a non-probabilistic classifier. It is a supervised learning algorithm for classification that observes data and labels, and then recognizes patterns. Given a set of feature vectors mapped into the feature space, an SVM tries to find the largest gap that can separate the classes. Given we have p different features, SVM forms a hyperplane of dimension $p-1$ that divides categories given feature vectors of p elements lying in a p -dimensional space. Figure 1.2 shows an example of using SVM to classify 2 classes using 2 features. As shown in the figure, SVM tries to find a gap that maximizes the distance between support vectors (marked in circles).

The benefit of using an SVM to classify activities, especially of using it in internal nodes of tree classifiers, is that it draws a decision boundary for activities that are not easily characterized by probabilistic models. In internal nodes of a decision tree classifier that contains more than one activity, using a single-peak Gaussian model may not be a good characterization of the multiple activities represented. SVM, on the other hand, does not care about interior feature points and only boundary points matter. This leads to another benefit of using an SVM,

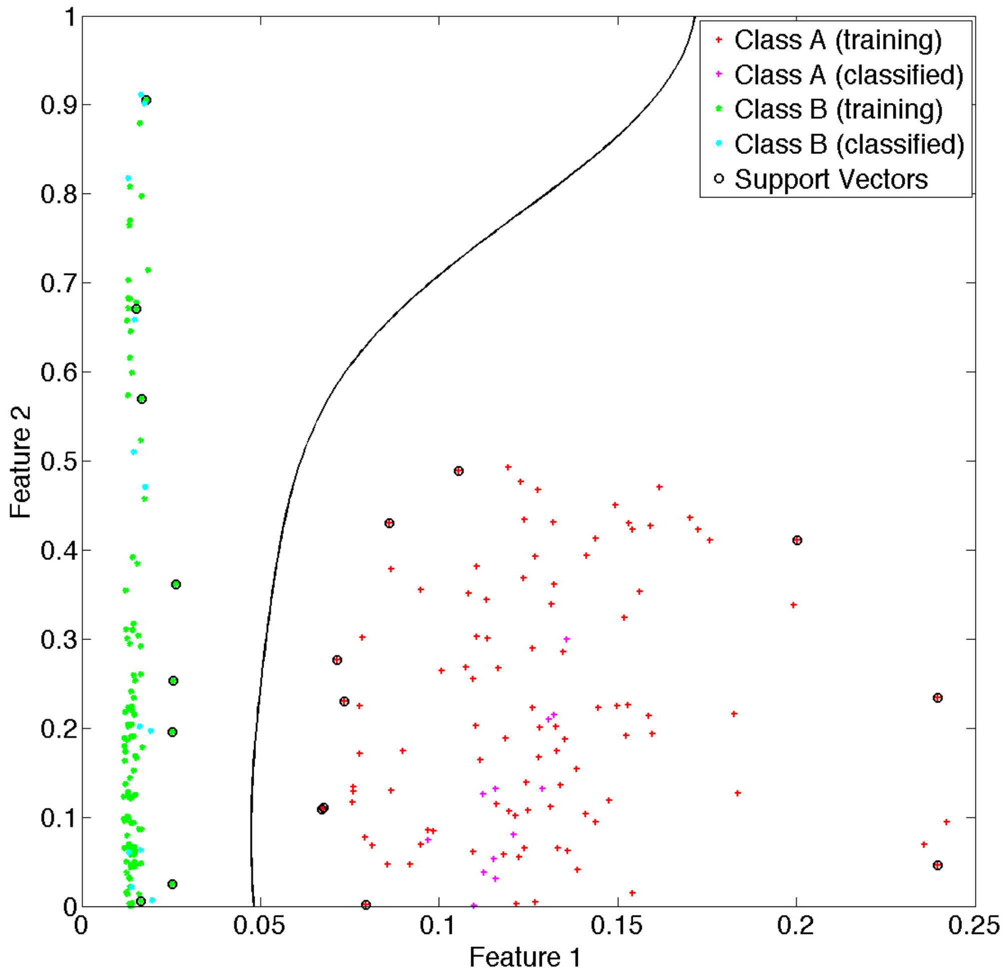


Figure 1.2. Support vector machine on 2 classes

which is when trying to classify some activities whose features are easily affected by noise. For some features of passive activities, such as energy of standing and sitting, values of such features are very low and easily affected by any external noise or unexpected movements. Therefore, the means and standard deviations may not be very representative. In this study, we used the Gaussian radial basis as the kernel function, with scaling factor $s = 1$. SVMs were mainly used to classify stationary activities, such as standing, lying, and sitting, since the training data are concentrated, and few outliers occur, which is suitable for the use of SVM.

The main drawback of using SVM in our applications is the established theories and methods are mainly targeted on classifying 2 classes, while in our research there may be more than 2 classes to be dealt with. Also, as for the Naïve Bayes classifier, the curse of dimensionality forces one to collect more ground truth to be able to make decisions in a higher dimensional space when more activities must be separated.

1.3 Motion Reconstruction

In this section, we introduce two major components that are used in motion reconstruction, which are the use of complementary filtering and human biomechanical models for motion decomposition.

1.3.1 Complementary Filter

A complementary filter is often used in motion reconstruction and tracking, due to its simple structure and fast computation, which is more suitable for real-time applications [20][21]. It is often used in flight navigation or robotics to estimate the orientation of the sensor. It fuses multiple estimation measurements that have noise of complementary spectral characteristics [22]. For instance, in estimating the vertical velocity using accelerometers and a barometric sensor, one would suffer from long-term drift when integrating the acceleration, and instantaneous noise when using barometric sensor to estimate velocities. The complementary filter applies a low-pass filter to the integrated signal of acceleration, and a low-pass filter to velocity estimated by barometric sensors, then sums up the filtered signals to have a better estimate of velocity.

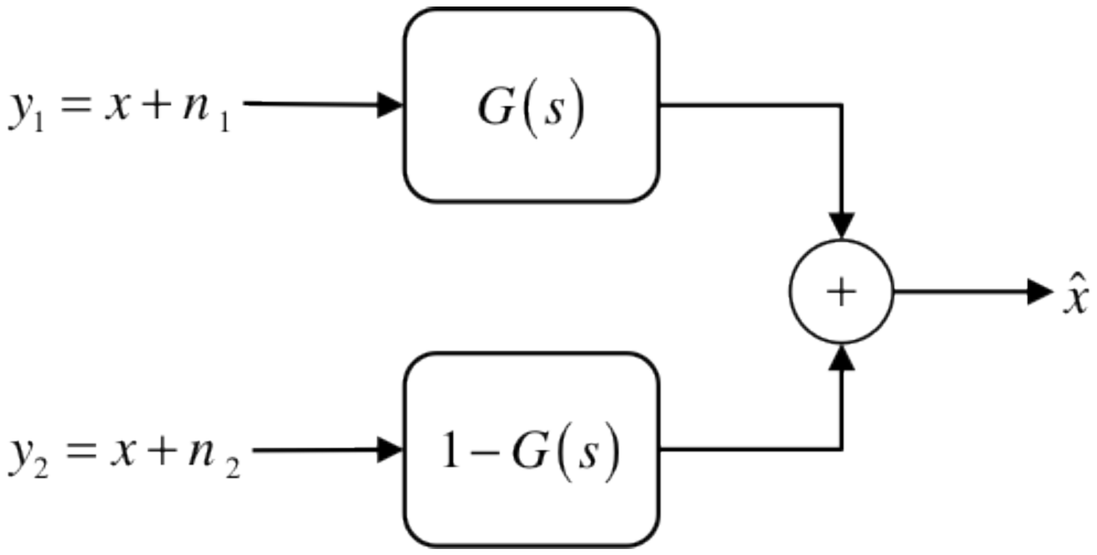


Figure 1.3 Complementary filter example

More formally, as shown in Figure 1.3, suppose that we have two methods to estimate x,

$$\begin{aligned} y_1 &= x + n_1 \\ y_2 &= x + n_2 \end{aligned} \tag{1.14}$$

where n_1 contains mostly low-frequency noise and n_2 mostly high-frequency noise. The complementary filter then applies a low-pass filter $G(s)$ to y_1, and another complementary high-pass filter $(1 - G(s))$ to y_2. The resulting signal can then be written as

$$\begin{aligned} \hat{x} &= G(s)y_1 + (1 - G(s))y_2 \\ &= G(s)(x + n_1) + (1 - G(s))(x + n_2) \\ &= x + G(s)n_1 + (1 - G(s))n_2 \end{aligned} \tag{1.15}$$

The signal x is then all pass filtered, while noises n_1 and n_2 are high and low pass filtered respectively.

In real-time motion tracking applications using inertial measurement units (IMU) such as accelerometers, gyros and magnetometers, we can estimate the orientation of the sensor either using accelerometers and magnetometers, or integrating the gyros. However, both methods have drawbacks. In the accelerometer and magnetometer method, one suffer from measurement noise of MEMS sensors; in the gyro method, the integration of angular velocity causes serious drift as time goes by. The complementary filter then filters the orientation estimation using accelerometers and magnetometers with a high-pass filter, and the orientation estimation using gyro with a complementary low-pass filter, hoping both filters can help to remove the corresponding noises. The remaining issue becomes how the filters should be designed and how the result can be verified, as can be seen in the following chapters.

1.3.2 Motion Decomposition

In this research, we use the biomechanical models for the human limbs to represent human motions. Based on [23], we model human joints by using the hierarchy as shown in Figure 1.4. This hierarchy has the root at hip center, and extends to the feet, hands and head. In this model, parent joints are those closer to the center of the body, while child joints are those connecting to their parents and away from the center. The bones are defined by surrounding parent and child joints, their own orientations, and their lengths. Motions are then just a set of rotations within this hierarchy.

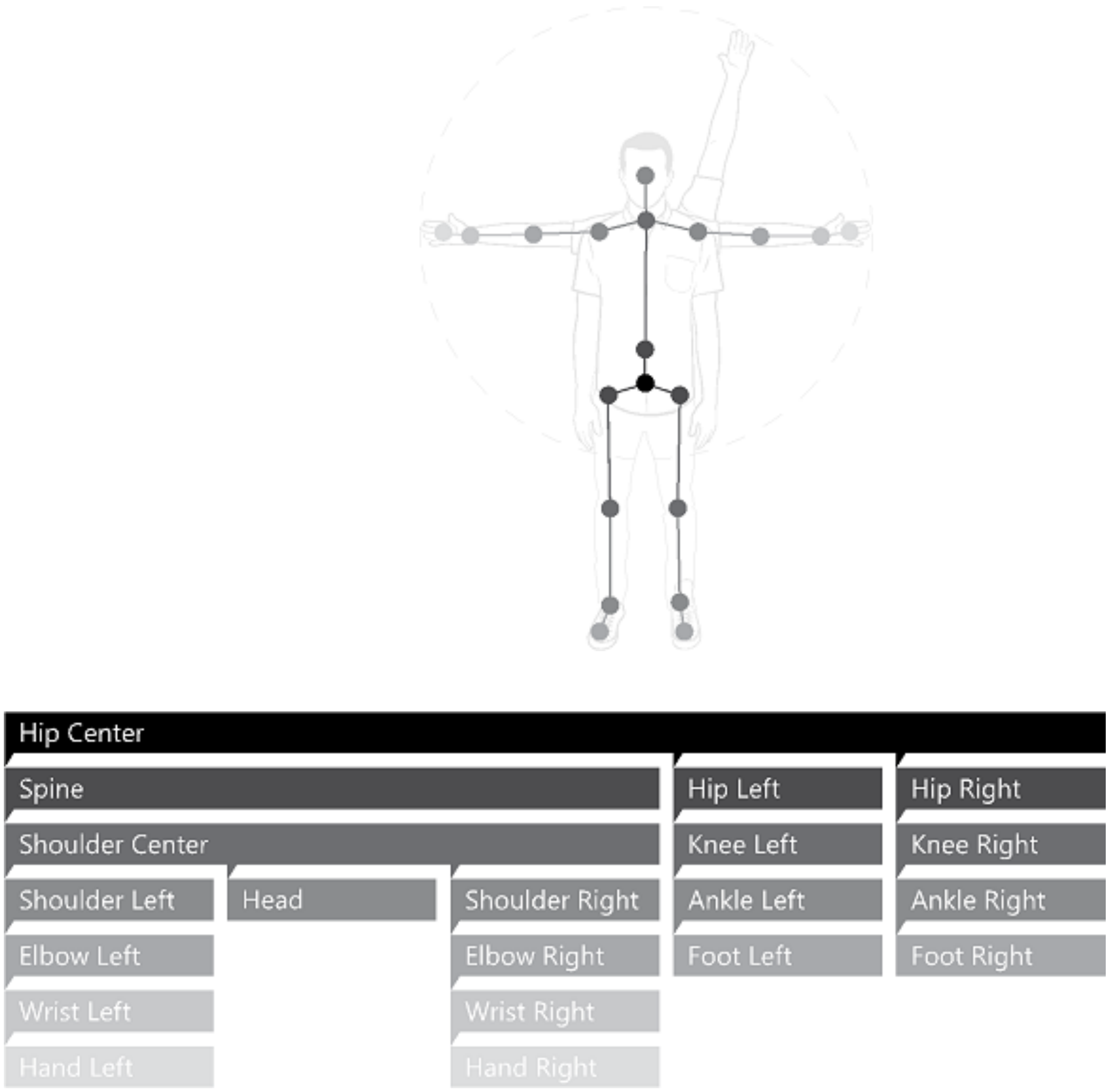


Figure 1.4 Joint hierarchy

1.4 Devices Used in this Research

1.4.1 Micro-electromechanical Systems (MEMS)

Micro-electromechanical systems refer to the technology of making mechanical devices very small, for which the size of such devices traditionally relying on classical physics are large. Because of the advance of semiconductor technologies, their sizes generally range from 20 to

1000 micrometers, with structures usually consisting of a microprocessor together with various microsensors [24][25]. There are many applications benefiting from MEMS technology, such as accelerometers in gaming devices, cars and cellphones, or gyroscope in modern aircraft and cars, microphones in cellphones, etc.

In this research, we mainly use three kinds of MEMS devices: accelerometers, gyros, and magnetometers. MEMS accelerometers, which sense the external force (including the gravity and force applied to the sensor), usually consist of a mass and a cantilever beam. When the mass deviates from its natural position, the capacitance between the beam and the mass is measured and converted digitally, which relates the measured capacitance to the force applied to the mass. Other kinds of accelerometers include piezoelectric material in the spring, and convert the mechanical deformation of the spring into voltages. The MEMS gyro uses a piezoelectric material to produce a constant oscillation to pick up the Coriolis effect when rotation occurs. The torque induced by rotation is then transformed into electric signals and the angular velocity can be measured. The magnetometer is a magnetic field sensor that senses magnetic fields from the Earth and other sources. It relies on the Hall effect to sense the magnetic field: when a magnetic field is present, the current exposed to this field gets deflected and thus the magnetic field force can be calculated.

This research utilizes sensor fusion to combine signals from these sensors. Sensor fusion is a way to combine data from multiple types of sensors to produce a more reliable result than using the sensor sources separately and individually [26][27]. There are many applications. For example, indoor navigation combines Wi-Fi signals with accelerometer measurements to help positioning when indoors [28], while GPS/INS uses GPS signals to help calibrating measurements from inertial navigation systems (INS) [29]. In this research, we try to combine signals from accelerometers, gyros and magnetometers to provide a better estimate of object orientation. To combine sensor signals from diverse sources, generally one has to rely on some algorithms for more accurate results and less noise. Common techniques used for estimating the orientation include Kalman filters and complementary filters.

1.4.2 Kinect and Software Development Kit (SDK)

The Kinect was released by Microsoft Corporation in November 2010. It consists of an RGB camera with 1280x960 resolution, an infrared (IR) camera that captures depth with 640x480 resolution, a multi-array microphone, and a triaxial accelerometer of range 2g. The Kinect is a motion sensing devices that can capture human motion by recognizing various joint positions in the space relative to it. In June 2011, Microsoft further released the Kinect SDK [23][30], which assists developers tracking human joint and skeleton positions in real-time, building 3D models for objects using Kinect Fusion, recognizing speech with its API, performing face tracking, etc. With this SDK, other than its original gaming purposes, people started using the Kinect for many other purposes [31][32][33][34][35]. In this research, we use the joint and skeleton tracking SDK to help us locate the positions of human body parts, and therefore we can track the trajectories.

Although the Kinect SDK provides an easy and powerful way to track the trajectories in real-time, there are a few drawbacks in using it. First, it is pose and gesture significant. Since it was originally designed for players facing the television to play with it, one has to face the Kinect before it can recognize human joints. The Kinect’s tracking ability is seriously degraded when not facing the sensor. Second, the Kinect is color and background sensitive. For the best recognition, there should be a solid background, and it should be clean with less colors, which suggests that Kinect has limitations in real-life environments of various backgrounds. Third, the Kinect requires a power outlet to operate, and this means that it is not suitable for outdoor recognition. Finally, the joint and skeleton tracking result using Kinect SDK is not accurate and fast enough for our application. The refreshing frame rate is not high enough (at a maximum of 29 frames per second), nor does its accuracy meet our need. Therefore, given the circumstances that Kinect might not always be available, and with an inaccurate tracking result, some signal processing algorithms and techniques are necessary to meet our goal.

Chapter 2

A Universal Hybrid Decision Tree Classifier Design for Human Activity Classification and Its Applications

2.1 Introduction

In this chapter we discuss the development of a tree classifier for human motions. Previous systems [1][2] have classified daily activities using naïve Bayes classifiers with accuracy ranging from 30% to 90%. However, these systems classified only 5 groups of activities based on their intensity levels. In people's daily lives, they may walk slowly or fast, they may rush for buses, or they may just sit on the couch with different gestures. As the number of activities of interest grow, to classify them with a single-stage classifier that separates all activities at the same time becomes difficult at many levels, not least in the large volume of training data required. Additionally, as researchers in different fields may care about different levels of details of activities, the number of classes would grow even more. A decision tree is a better tool in these situations. Decision tree classifiers [3][4][36][37][10] handle complex decision regions by partitioning them into sets of simpler low dimensional regions. This "divide and conquer" nature also helps to avoid the curse of dimensionality compared to single-stage classifiers. In multivariate analysis, where one usually needs to estimate a large number of classes from many features with only a small training data set, one is forced to go to high-dimension if using single-stage classifiers. However, a decision tree classifier helps to ease this problem by selecting only a few features at each internal node; if these features are carefully selected, there can be little performance degradation. Other advantages of tree-structured approaches include robustness to outliers in training data, flexibility and extensibility of target classes, and invariance under monotone transformations.

However, to classify daily-life activities, traditional decision tree methodologies are not enough. For example [6] has classified 7 real-life activities with a custom tree classifier using a comprehensive system containing various kinds of sensors including accelerometers, electrocardiogram (EKG), global positioning system (GPS), etc. The system collected complete information of activities and achieved overall 82% accuracy on classifying 7 activities. Another study [4] classified 20 activities with decision tree classifiers, with data collected by accelerometers on hip, wrist, ankle, arm and thigh. It verified the testing data with various classifiers and had the highest accuracy of 84% using the C4.5 tree classifier. In these papers, there are some activities that can be easily determined, especially classes related to motion activities. However, classifiers were confused by activities without simple characteristics, or those that share some common features with other activities. Therefore, there exists a need for a tree classifier with more flexibility, which selects its separation criteria and thresholds of internal nodes individually, and thereby applies different rules in drawing the decision boundaries, so as to separate these confusing activities easily.

Another important issue is the generalizability of a model. In a clinical trial, due to very high logistical costs, one can only control a small group of people to have good training data (i.e., with reliable ground truth); for the rest of the people, one might end up with something incomplete or with a small amount of training time. However, large classifier accuracy gains result when models are adapted to individuals. Thus, to make the model generalizable, we need a decision tree that fits the general public or significant subpopulation categories, whereby the structure and features of internal nodes are determined, and one only tunes the thresholds of each node based on shorter training sequences from individuals. In [3] similar work was done, in that they classified 7 activities with a custom fixed tree. However, it did not point out the importance of using a fixed structure, and neither did it have high enough accuracy for medical use (82%).

In this chapter, our goal is to create a fixed structured tree capable of classifying daily life activities daily with satisfactory accuracy. We took an empirical approach by collecting a large amount of data. We report a complete procedure for daily life activity classification, from data collection, feature extraction, tree structure and feature selection, to testing. The resulting classifier is generalizable and has high accuracy. Using leave-one-out cross validation, it produced average classification accuracy of 91.5%. In contrast, the MATLAB personalized tree classifiers using Gini's diversity index as the split criterion followed by optimally tuning the thresholds for each subject yielded 69.2%.

The remainder of the chapter is organized as follows. We provide the system setup, including data collection and tree formation in Section 2.2. In Section 2.3, we introduce two classifiers that are used in this chapter, and describe how to form a hybrid tree classifier. In section 2.4, we move from hybrid tree classifiers to the proposed universal hybrid tree classifier. Simulations and results are given in Section 2.5, followed by result in Section 2.6, and a short discussion of the universal tree in 2.7. We discuss three related issues associated with the universal hybrid tree classifier in the next three sections: monitoring workspace activities in section 2.8; estimation of accelerometer orientation in section 2.9; and the feature selection problem in section 2.10. Conclusions are drawn in Section 2.11.

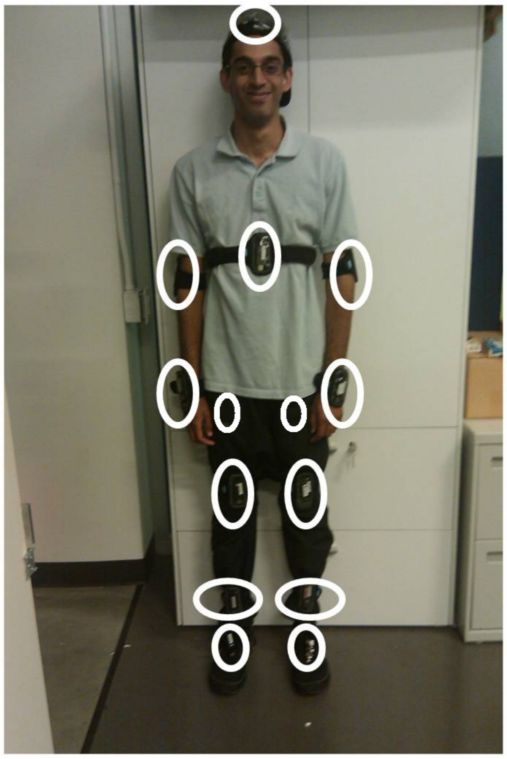


Figure 2.1. Locations for 14 sensors

2.2 System

We now explain how the whole system works, from data collection to activity classification.

2.2.1 Data Collection

We used the Gulf Coast Data Concept USB Accelerometer X6-2mini with a built-in tri-axial accelerometer [38] to collect the data. It is a small device, which can be easily worn on any part of the body. The accelerometers collect data at the sample rate 160 Hz, resolution 16 bits, and gain ±6g. We put accelerometers on 14 parts of the body, as described in Figure 2.1 and Table 2.1. In this research, we over-instrumented the test subjects in order to get a complete data set. However, we found that only few sensors are needed and crucial for activity classification for particular activities, as will be described later. In the training and testing processes, each

Table 2.1. Sensor placements

Upper limb and head	Lower limb
Forehead	Left and right pockets
Chest	Left and right knees
Right and left elbows	Left and right ankles
Right and left wrists	Left and right toes

Table 2.2. Collected activities

Motion	Stationary
Walk slowly	Stand
Walk fast	Sit upright
Run	Sit while slouching
Walk up slope	Sit while hunching
Walk down slope	Lie on back
Walk upstairs	Lie on stomach
Walk downstairs	Lie on side

sensor collected x, y, and z directions of acceleration, thus producing in total 42 channels of data. Seven people took part in data collection, with 2 hours of measurement for each person. The subjects were asked to perform the series of activities listed in Table 2.2, as being representative of some activities from daily living.

In order to build the ground truth, when the test subject was doing assigned activities, an annotator followed him/her to label the activities. The annotator used an Android phone to mark changes in contexts for reference purposes. In this program, users can edit the list of activities and their order to fulfill their needs for experiments. During the experiment, start and end markers were manually added. The annotation program was a considerable advance over previous pen and paper methods, particularly in providing consistent time stamps. Since there was no communication mechanism between accelerometers, or between accelerometers and the Android phone, before starting any measurement we simply tied all the sensors together and shook them sharply, while at the same time pressing the synchronize button of the annotation program. This simple action gave distinct signatures in all the signals, and thus provided the reference for signal synchronization.

2.2.2 Signal Processing Toolbox

After measurements, a comprehensive custom MATLAB toolbox was employed, that enabled data merging and synchronization, activity labeling, feature extraction, hybrid tree classifier formation, feature selection, and finally activity classification. The program includes a graphical user interface (GUI), with which users can easily click and load files, input parameters, and visualize data spreading and decision boundaries for classes. First, acceleration data was loaded and merged to a MATLAB variable. Then the merged data and its annotation were synchronized based on signatures made at the beginnings of the measurement sessions, tagging the data with the ground truth.

2.2.3 Feature Extraction

After synchronizing the data and the ground truth annotations, we converted the measured acceleration into various features. Feature extraction was done by using a moving window and extracting different features. The moving window was of length four seconds and of step size one second. The window size of four seconds ensured that we captured more than a complete cycle for every activity, therefore having similar features for each class. The step size of a second makes the activity classification in the resolution of one prediction per second; this is enough for the purpose of daily activity classification, where changes between activities are not rapid.

In each window, thirty-one features for each accelerometer were calculated. These thirty-one features can be clustered into three categories, which are the spatial, time, and frequency domains. Some instances for each category and their uses follow.

2.2.3.1 Spatial Domain

"Correlations" between x, y, and z directions were calculated. These features showed how the posture changes during some classes, or emphasized the transition during activities.

2.2.3.2 Time Domain

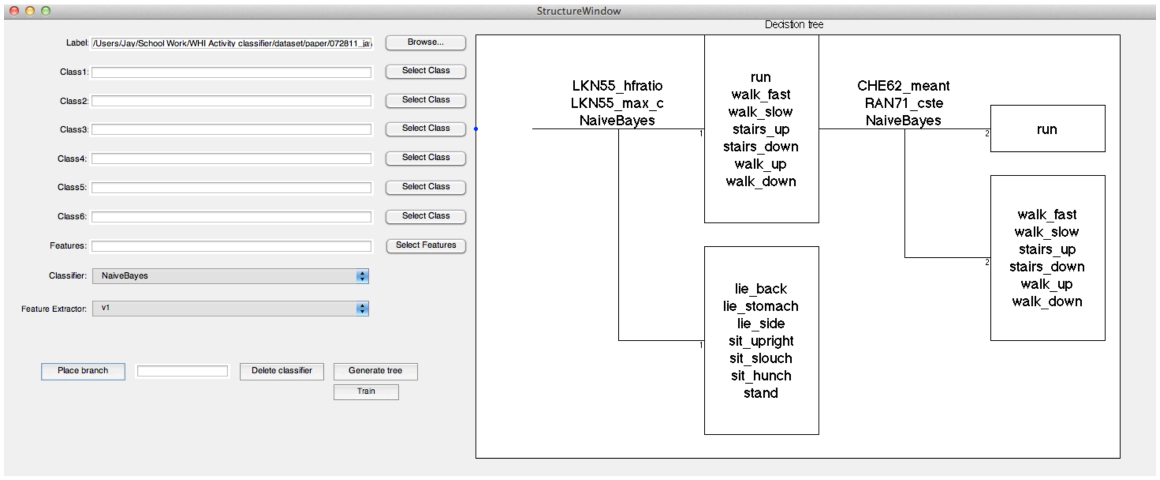
"Standard deviation" and "short-term energy (total energy of the windowed signal)" of a window were calculated. These features had strong correlations to how intense the movements were, which were suitable in distinguishing motion and stationary activities. The feature "maximum values" of a frame for each direction indicated the range of different motions.

2.2.3.3 Frequency Domain

The features "sidelobe location" and "DC value to sidelobe location ratio" indicated what was the dominant frequency of each activity. These features were helpful in distinguishing among periodic activities with different periods, such as run, walk fast, and walk slowly. The feature "f-ratio" which was the ratio of the energy of frequency band above a certain threshold to the total energy of the whole signal in the frame, indicates whether the energy was concentrated on certain frequency bands or spread through the entire frequency spectrum.

2.2.4 Tree Formation

The hybrid decision tree classifier was built using the structure toolbox, which is a sub-toolbox of the entire system as shown in Figure 2.2. The decision tree was built manually based on the knowledge of various domains of the signal. In the manner of Figure 2.2, we first grouped motion activities to the upper node, and stationary ones to the lower node, selected "horizontal f-ratio" and "maximum value of feature vector length" of the left knee sensor as the separating features, and then applied the naïve Bayes classifier for the root node as the separating classifier.



Label: /Users/Jay/School Work/WHI Activity classifier/dataset/paper/072811_ja

Classifier: NaiveBayes

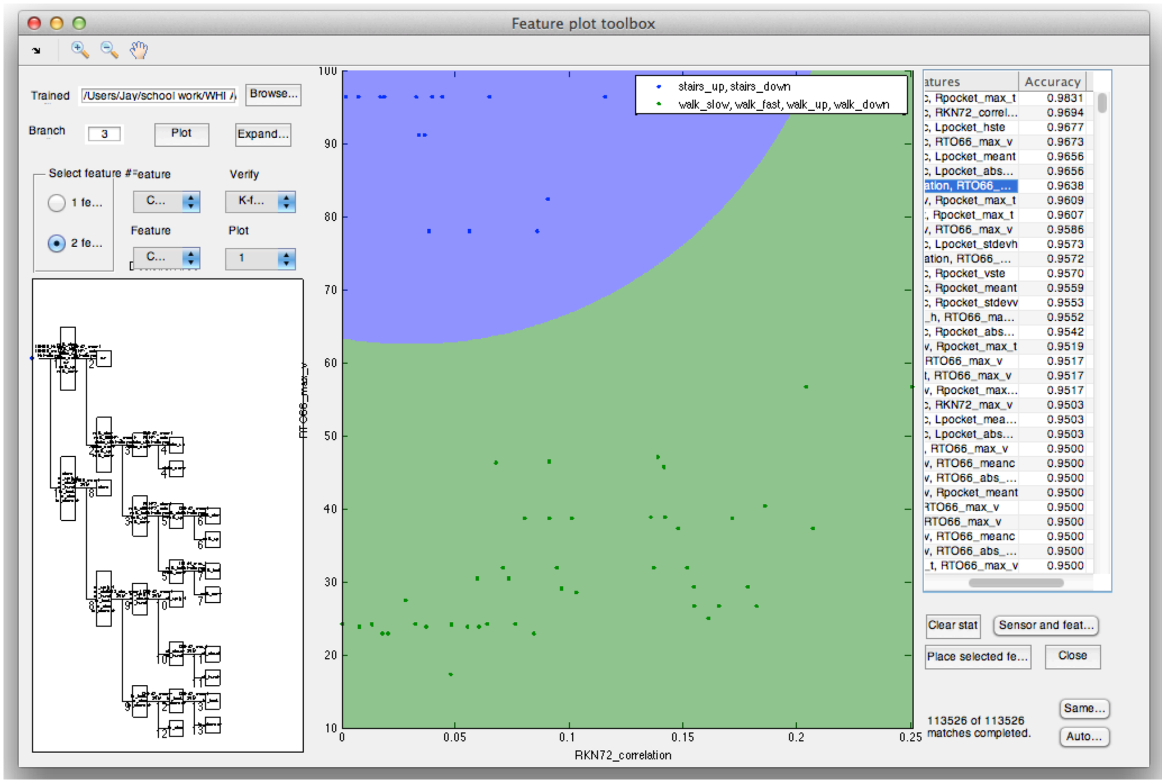
Feature Extractor: v1

Figure 2.2 Structure toolbox for building a decision tree

These two groups of activities were distinct and clearly separated in the space composed by these two features. Continuing in this fashion, we ended up with several complete hybrid decision tree classifiers. We then used a feature selection tool to visualize the feature vectors in their spaces to further determine the separating classifiers and features of each internal node, which will be described next.

2.2.5 Feature Selection

Once the tree structure was formed, we used the feature selection toolbox to help us determine feature sets and the type of single-stage classifiers that were good in separating classes. This toolbox, which was another sub-tool of the complete toolbox, was able to test trained features of a specified node of the tree against itself (resubstitution error), or with k-fold cross-validation. This toolbox also tested any combination of features on various types of classifiers automatically. Therefore it speeded up the feature selection process and covered some set of features that were not easily imagined by only observing the signals. Furthermore, it enables visualization of the feature vectors of activities in their feature spaces, and draws the decision



atures	Accuracy
c, Rpocket_max_t	0.9831
c, RKN72_correl...	0.9694
c, Lpocket_hste	0.9677
c, RTO66_max_v	0.9673
c, Lpocket_meant	0.9656
c, Lpocket_abs...	0.9656
ation, RTO66_...	0.9638
v, Rpocket_max_t	0.9609
, Rpocket_max_t	0.9607
v, RTO66_max_v	0.9586
c, Lpocket_stdevh	0.9573
ation, RTO66_...	0.9572
c, Rpocket_vste	0.9570
c, Rpocket_meant	0.9559
c, Rpocket_stdevv	0.9553
_h, RTO66_ma...	0.9552
c, Rpocket_abs...	0.9542
v, Rpocket_max_t	0.9519
RTO66_max_v	0.9517
t, RTO66_max_v	0.9517
v, Rpocket_max...	0.9517
c, RKN72_max_v	0.9503
c, Lpocket_mea...	0.9503
c, Lpocket_abs...	0.9503
, RTO66_max_v	0.9500
v, RTO66_meanc	0.9500
v, RTO66_abs_...	0.9500
v, Rpocket_meant	0.9500
RTO66_max_v	0.9500
RTO66_max_v	0.9500
v, RTO66_meanc	0.9500
v, RTO66_abs_...	0.9500
_t, RTO66_max_v	0.9500

113526 of 113526 matches completed.

Figure 2.3 Feature selection tool

boundary; therefore one can see and select the feature set for a node that was most robust to outliers. Figure 2.3 shows how the feature selection toolbox operates. In this toolbox, we could test all kinds of feature combinations for all nodes of the tree, and the decision boundary (based on naïve Bayes or SVM) was drawn. By observing the spread of feature points, one can even calculate a metric to rank the feature combination to have the best feature set separating classes of each node. Given the fast search time and simple feature vector visualization, we could easily verify if each node of the hybrid decision tree classifier had good classification performance.

2.2.6 Testing / Classification

Once the decision tree, including its separating features, was formed, we tested it using the testing toolbox. This tool loaded the testing set label and the tree classifier, and classified activities in a real-time fashion.

2.3 Hybrid Tree Formation

From chapter 1, we know that in designing a decision tree classifier T, where

$$T = \left\{C\left(t\right), F\left(t\right), D\left(t\right)\right\} , \tag{2.1}$$

the procedure can be viewed as an optimization problem in two steps:

$$\begin{gathered}\text{Step 1: For a given } C \text{ and } F\text{, find } D = D\left(C,F\right) \text{ such that} \\ P_{\mathrm{e}}\left(C,F,D^{*}\left(C,F\right)\right) = \min_{D} P_{\mathrm{e}}\left(C,F,D\right)\end{gathered} \tag{2.2}$$

$$\begin{gathered}\text{Step 2: Find } C^{*}, F^{*} \text{ such that} \\ P_{e}\left(C^{*}, F^{*}, D^{*}\left(C,F\right)\right) = \min_{C,F}\left(C, F, D^{*}\left(C,F\right)\right)\end{gathered} \tag{2.3}$$

where C, F, D are defined earlier. As mentioned in chapter 1, in most designs a single decision rule is used to find the final tree classifier. However, in this research we tried a more flexible design, which can adapt to the varying natures of the activity classes, and thus make the model expandable. This is achieved by introducing a new type of tree classifier called a hybrid tree classifier, which allows different types of decision rules available when designing the tree classifier. We do so by manually determining the decision tree structure, and found feature sets and decision rules for each internal node. Therefore, in the tree design procedure, we fix the set of classes C for all nodes and try to find the optimal feature set F^* and decision rules D^* that minimize the total probability of error. Thus, we have

$$\begin{gathered}\text{Given } C\text{, find } F^* \text{ and } D^* \text{ such that}\\ P_{\mathrm{e}}\left(C,F^*,D^*\right)=\min_{F,D} P_{\mathrm{e}}\left(C,F,D\right)\end{gathered} \tag{2.4}$$

Among many possibilities, we used two kinds of classifiers for the decision rules for internal nodes, namely, the naïve Bayes classifier and support vector machine (SVM).

2.4 Universal Hybrid Decision Trees

After creating a hybrid tree classifier for classifying various activities, we then tried to find a tree that can classify multiple sets of testing data from many subjects. As mentioned previously, this is important since with this tree we can specialize this classifier to individuals with minimal additional training, therefore making the model more easily applied to the general public. Suppose we have in total *M* sets of training data defined by (2.1), each of them from a carefully monitored test subject, and let TD_j be the training data for the subject *j*. Also, we have *N* manually structured trees, each with $l(i)$ internal nodes for $i=1,2,\cdots,N$. Then each tree T_i can be written as

$$\begin{gathered}T=\left\{F(t),D(t),C(t)\right\}\\ t=1,2,\cdots,l \;\; i=1,2,\cdots,N\end{gathered} \tag{2.5}$$

with $l(i)$ internal nodes. In every tree, the class subset for each node $C(t)$ is determined for every internal node. Let $TD_{j,t}$ be part of the training data TD_j whose classes that are involved in node t of tree T . $P_{\mathrm{e}|j,t}\left(F(t),D(t),TD_{j,t}\right)$ is the probability of error of node t when applying feature set $F(t)$ and decision rule $D(t)$ on training data $TD_{j,t}$. The procedure can be summarized by the following algorithm:

```
Begin
    1. Given a set of possible decision trees, randomly pick a tree $T$ with $l$ internal nodes.
    2. For $t = 1$ to $l$
       Find the optimal set $\left(F^*(t), D^*(t)\right)$ that minimizes the weighted probability of error
       $\left(F^*(t), D^*(t)\right) = \arg\min_{(F(t),D(t))} \sum_{j=1}^{M} w_j \cdot P_{e|j,t}\left(F(t), D(t), TD_{j,t}\right)$
       where $w_j$ is the weighting function for the subject $j$, indicating the weighting of that
       type of people to the general public.
    3. If $\sum_{j=1}^{M} w_j \cdot P_{e|j,t}\left(F^*(t), D^*(t), TD_{j,t}\right) > th_{err}$
       Terminate the for loop, go to step 1 and try the next tree $T$, where $th_{err}$ is the predefined
       error threshold
       End If
       End For
    4. Output the tree classifier
       $T^* = \left\{F^*(t), D^*(t), C^*(t)\right\} \quad t = 1, 2, \cdots, l$
End Begin
```

The above algorithm provides a means to find a compromise tree that accounts for the differences among people, while maintaining a satisfactory error rate. After creating this universal hybrid decision tree classifier, when there is a test subject with only small amount of training, we can then apply the tree classifier, include the tree structure, its separating features and decision rules, to the test subject. The only thing that is changed is the decision threshold for each internal node. The threshold is determined specifically for each subject, while maintaining the decision tree structure.

2.5 Simulation

2.5.1 Methods

Three different kinds of decision tree classifier mechanisms were used, namely the custom universal hybrid decision tree, automatically generated trees for each subject, and automated trees but with tuned thresholds for individuals. All of them were provided with full data and extracted features. The common structure and features of the custom decision tree were formed based on the algorithm described earlier, then for each person, we applied different thresholds for its internal nodes. The personalized automatically generated trees were used in this report to compare with our custom tree. For each test subject's automated tree, we kept the structure and separating features but calculated thresholds for other subjects. This showed how well the personalized automated tree could perform when applied to different people. The classification results for these three classifiers were calculated using testing on the training set (resubstitution error), training on 40% of data and testing on the rest 60% (40%-60% partition error), and leave-one-out cross-validation (LOOCV).

2.5.2 Custom Universal Hybrid Decision Tree

The custom decision tree (Figure 2.4) consisted of 27 nodes, where 13 were internal nodes with binary separation. We first used all data from 7 subjects to determine the tree structure and

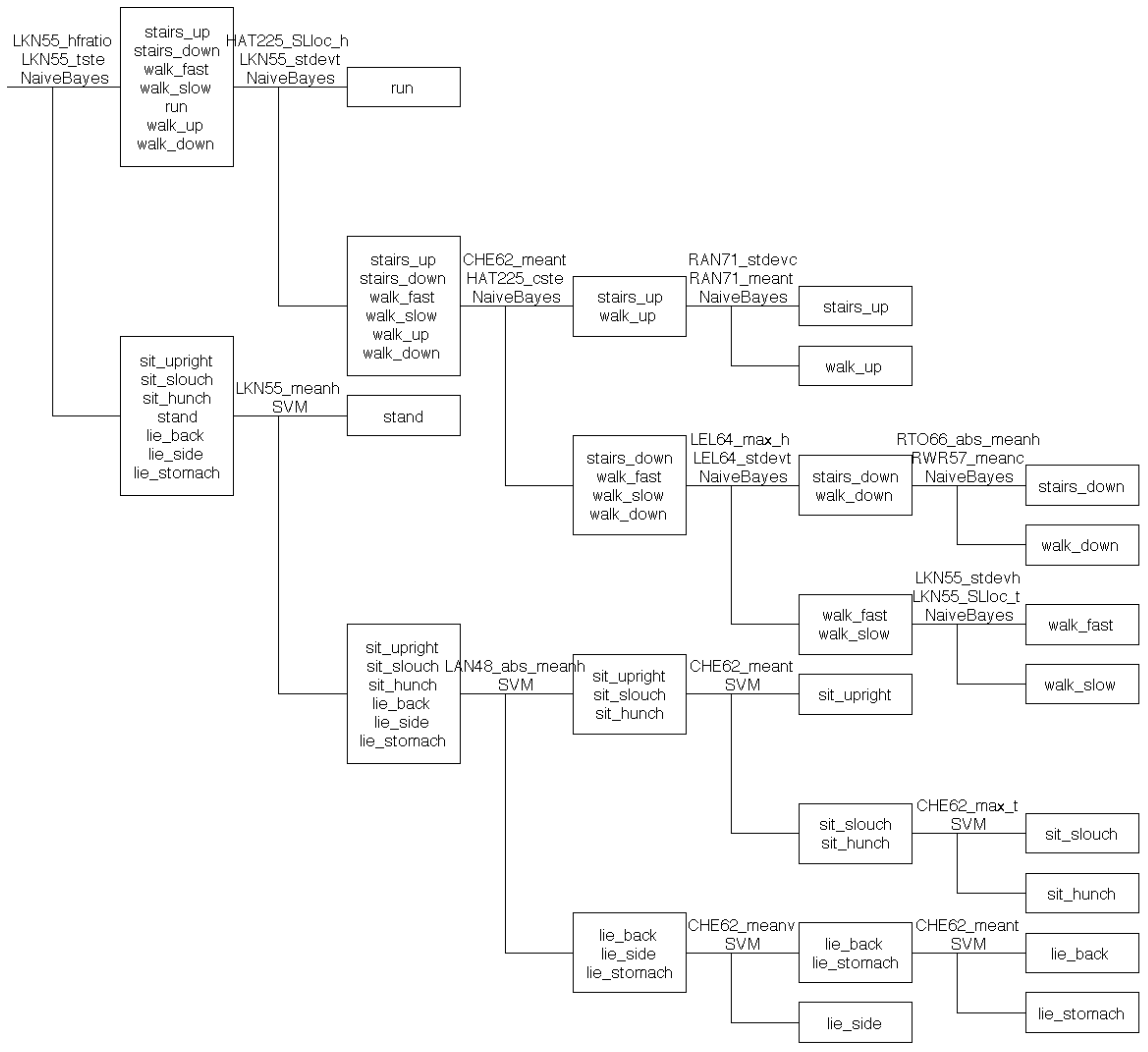


Figure 2.4 Universal hybrid decision tree classifier

features giving the highest accuracies. Afterward, for each subject, we determined individual thresholds for internal nodes of the tree. In this tree, we first separate motion activities (stairs up and down, walking fast and slow, walking up and down ramp, and running) from stationary activities (sitting upright, sitting slouch, sitting hunch, standing, lying back, side and stomach). In the upper part of the tree (motion activities), we used naïve Bayes classifiers on each branch, and assumed equal prior probabilities; in the lower part (stationary activities), we used SVMs with the Gaussian radial basis function kernel. For nodes using naïve Bayes classifiers, we selected two features that gave the highest weighted accuracy in separating classes; for nodes with SVM classifiers, we only selected one feature due to computational concerns. The threshold values were determined specifically for each individual. After creating this universal tree classifier, we just changed the separation thresholds for each individual, and the structure of the tree remained unchanged during the simulation.

Resubstitution error was calculated by finding the thresholds from the whole raining data, and then testing on the same data. 40%-60% error was estimated by finding the thresholds from 40% of the data, and then testing on the remaining data. LOOCV was done by finding a universal hybrid tree structure using data from all people except one subject. Thresholds were then calculated using 40% of the data from the subject left out of the tree structure creation process, and then tested on the remaining 60% of the data.

2.5.3 Automatically Generated Tree

Automatically generated decision trees were created using the MATLAB built-in function "classregtree." This function used Gini's diversity index [12] as the separation criterion. The rule of thumb of this function is to find the largest class first, and then separate it from other classes. It should be noted that the automated trees were specific to the target training data. Therefore each subject has a unique automated decision tree. The acquired data varied from person to person, even from different parts when we chopped it. Thus, the size of the automatically generated tree was different; on average the tree had 19.9 internal nodes ranging from 16 to 26, and on average 20.9 leaf nodes with a range from 31 to 53. For the resubstitution error estimation, decision trees for each person were generated and tested against themselves. In 40%-60% error estimation, specific trees were generated for each person on his/her 40% of data, and tested on the remaining 60%. In LOOCV, we found the automated decision trees by using data of all except one subject, and then tested on the targeted subject.

2.5.4 Automatically Generated Tree with Tuned Thresholds

In order to compare to the universal tree structure, we kept the same structure of automatically generated trees from the previous section and tuned their threshold values according to testing data. In this test, we called MATLAB to generate the automated decision tree for subject A, then kept the tree structure and selected features but changed the separation thresholds of internal nodes of the tree using testing data of subject B, C, etc. The resubstitution error was estimated by keeping the structure from one person, then changing separation thresholds with other people's data, and then tested against their own data. The 40%-60% error was acquired by keeping the tree structure but changing thresholds using 40% of data from one subject, then tested on the remaining 60% of data. In LOOCV, we found the decision trees using all but one subject's data, then determined the thresholds using 40% of the last subject's data, and then tested on the remaining 60%.

2.6 Result

Table 2.3 shows the classification error rates for universal hybrid trees, automatically generated trees and automatically generated trees with tuned thresholds. Table 2.4 to Table 2.6 (next page) show the classification results for three different classifiers, using LOOCV. The universal hybrid tree shows considerably better results than the others.

Table 2.3 Classification result summary (mean ± standard deviation)

Classifier	Resub. Error	40%-60% Error	LOOCV
Universal Hybrid Tree	93.5%±0.6%	92.5%±2.6%	91.5%±3.2%
Auto Tree	97.7%±4.4%	92.5%±6.0%	73.0%±13.5%
Auto Tree with Tuned Threshold	54.4%±13.3%	47.7%±14.3%	69.2%±12.2%

Table 2.4 Confusion matrix of universal hybrid decision trees, tested using LOOCV

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	
a	**1518**	0	0	5	4	572	0	7	0	0	0	0	0	0	a = lie_back
b	0	**1808**	0	0	0	292	11	0	0	0	0	0	0	0	b = lie_side
c	0	0	**1794**	2	302	0	0	0	4	0	0	0	0	0	c = lie_stomach
d	0	0	0	**1980**	0	0	0	0	1	0	0	0	0	0	d = run
e	0	0	0	3	**1774**	0	303	0	18	0	0	0	0	0	e = sit_hunch
f	0	0	0	3	0	**1480**	594	5	0	0	0	0	0	0	f = sit_slouch
g	0	0	0	0	300	273	**1510**	7	7	0	0	0	4	1	g = sit_upright
h	0	0	0	37	0	0	0	**1202**	22	0	198	16	300	26	h = stairs_down
i	0	0	0	1	0	0	0	0	**942**	0	0	0	13	693	i = stairs_up
j	0	0	0	0	0	0	0	0	0	**2070**	0	0	18	0	j = stand
k	0	0	0	68	0	0	0	268	103	0	**735**	130	227	151	k = walk_down
l	0	0	0	144	0	0	0	19	203	0	345	**851**	204	338	l = walk_fast
m	0	0	0	45	0	0	0	0	5	0	7	283	**1654**	112	m = walk_slow
n	0	0	0	5	0	0	0	2	298	0	0	19	316	**1353**	n = walk_up

Table 2.5 Confusion matrix of automatically generated tree, tested using LOOCV

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	
a	**1085**	0	0	0	0	167	0	0	0	0	0	0	0	0	a = lie_back
b	0	**1128**	128	0	0	0	0	0	0	0	0	0	0	0	b = lie_side
c	184	0	**1067**	0	0	0	0	0	0	0	0	0	0	0	c = lie_stomach
d	0	0	1	**1164**	0	0	0	0	0	0	5	0	0	0	d = run
e	0	0	0	0	**876**	0	373	0	0	0	0	0	0	0	e = sit_hunch
f	0	0	0	0	0	**517**	719	0	0	0	0	0	0	0	f = sit_slouch
g	0	0	0	0	309	168	**774**	0	0	0	0	0	0	0	g = sit_upright
h	0	0	0	0	0	0	0	**512**	29	0	347	21	10	60	h = stairs_down
i	0	0	0	3	0	0	0	3	**552**	0	35	4	12	279	i = stairs_up
j	0	0	0	0	0	0	0	0	0	**1242**	0	0	0	0	j = stand
k	0	0	0	0	0	0	0	165	54	0	**480**	84	80	82	k = walk_down
l	0	0	0	0	0	0	0	183	78	0	439	**410**	117	24	l = walk_fast
m	0	0	0	0	0	0	0	5	9	9	44	6	**665**	517	m = walk_slow
n	0	0	0	0	0	0	0	16	143	0	79	14	31	**850**	n = walk_up

Table 2.6 Confusion matrix of automatically generated trees, tested using LOOCV

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	
a	**1085**	0	0	0	0	167	0	0	0	0	0	0	0	0	a = lie_back
b	0	**1128**	128	0	0	0	0	0	0	0	0	0	0	0	b = lie_side
c	184	0	**1067**	0	0	0	0	0	0	0	0	0	0	0	c = lie_stomach
d	0	0	1	**1164**	0	0	0	0	0	0	5	0	0	0	d = run
e	0	0	0	0	**876**	0	373	0	0	0	0	0	0	0	e = sit_hunch
f	0	0	0	0	0	**517**	719	0	0	0	0	0	0	0	f = sit_slouch
g	0	0	0	0	309	168	**774**	0	0	0	0	0	0	0	g = sit_upright
h	0	0	0	0	0	0	0	**512**	29	0	347	21	10	60	h = stairs_down
i	0	0	0	3	0	0	0	3	**552**	0	35	4	12	279	i = stairs_up
j	0	0	0	0	0	0	0	0	0	**1242**	0	0	0	0	j = stand
k	0	0	0	0	0	0	0	165	54	0	**480**	84	80	82	k = walk_down
l	0	0	0	0	0	0	0	183	78	0	439	**410**	117	24	l = walk_fast
m	0	0	0	0	0	0	0	5	9	9	44	6	**665**	517	m = walk_slow
n	0	0	0	0	0	0	0	16	143	0	79	14	31	**850**	n = walk_up

2.7 Discussion

2.7.1 Universal Hybrid Tree Structure

From the structure and chosen features of internal nodes (Figure 2.4), we first used naïve Bayes classifiers to distinguish motion and stationary activities using energy as the separating feature, based on physical intuition. We separated motion activities using naïve Bayes classifiers by grouping similar activities at different nodes of the tree. By similar we mean that there existed a common Gaussian model describing all activities contained in every node. For example, we separated running from walking and stairs using energy (standard deviation) and frequency (sidelobe location) since we believed that walking and stairs feature values are similar in this space, and were distinct from feature values for running. However, there were some difficulties in distinguishing different walking types and stairs, and this will be discussed quantitatively in section 2.10.3. In distinguishing stationary motions, we used SVMs for the separation rules. SVMs are good in classifying activities, especially when the values are not easily described using parameterized models. When we grouped several stationary activities, in the feature space these activities were centered at their own regions, which made it improper to characterize each node using single-peak Gaussian models. On the other hand, SVMs drew clear boundaries between stationary activities, especially in the early stages of the tree, when different stationary activities were grouped together. Thus, SVMs outperformed naïve Bayes classifiers at the early stages of the tree, and had similar classification accuracy near the leaf nodes.

2.7.2 Automatically Generated Trees

When comparing automatically generated trees versus universal hybrid decision trees, we found out that auto-generated trees were too specific to the training data to classify other subject's data.

As seen in the second row of Table 2.3, auto-generated trees are very accurate when trained and tested on the same set of data (97.7%). However, this accuracy dropped when we separated the data into two (92.5%) since they were too fine-tuned and overfit the training data. For LOOCV, classification accuracy dropped dramatically and displayed a large variance (73.0% ± 13.5%), indicating that the automatically generated trees were not easily generalized even we have a large amount of training data. Additionally, the average number of internal nodes of the LOOCV trees was 136.7 and 274.4 for total average number of nodes. All of the above clearly indicates that the personalized trees depend largely on the training subject, even the timing when the data is generated, thus making it hard to generalize the model to the general public.

We further illustrated this overfitting phenomenon of auto-generated trees by keeping the structures and features of the tree and tuning only separation thresholds. In the third row of Table 2.3, the classification accuracies were never satisfying (54.4%, 47.7%, and 69.2% for resubstitution error, 46%-60% error and LOOCV error respectively). The overfit tree structure of personalized automatically generated decision trees makes it hard to find a universal tree structure that classifies daily activities with acceptable accuracy.

On the other hand, the classification accuracies of the universal hybrid decision tree dropped only a little when we separate the subjects' data (from 93.5% to 92.5%), and when we generalized the result (from 93.5% to 91.5%); this can be seen from the first row of Table 2.3. In the first two tests, the tree structure and separating features were given in Figure 2.4 and the accuracy remained high for the two error estimations. In LOOCV, we fixed the tree structure and changed the separating features based on only part of the subject test data each time, and determined the thresholds for each subjects. The result was still above 90% with smaller variations compared to automatically generated decision trees. Therefore, we showed that universal hybrid decision trees can classify daily-life activities with acceptable accuracy.

2.7.3 Confusion Matrices

We discuss individual classification accuracy of each activity in this section. The universal hybrid decision tree successfully classified all stationary activities and running, but did not do as well in classifying walking and stairs activities. On the one hand, stationary activities have few variations and thus SVMs easily classified them. The high intensity and frequency of running made it distinct from other motion activities in feature spaces. On the other hand, 77% of stairs down were classified correctly and 12% of them were classified as ramp down; similarly, 76% of walk up-ramp were correctly classified, where 20% of them were classified as stairs up. Since the feature values of walking on ramps and stairs are very similar, it was difficult to distinguish among them. Additionally, only 76% of walk down-ramp and 68% of walk fast were classified correctly, and the rest of them were misclassified as other walking and stairs.

Automatically generated trees classified almost every activity when trained and tested on the same set of data, and had similar classification accuracy compared to universal hybrid decision trees. However, when estimating LOOCV error as in Table 5, only lie back (86%), lie side (93%) lie stomach (86%), running (100%) and standing (100%) have accuracies above 85%; all other activities have of accuracies below 80% (77% for sitting hunch, 71% for sit slouch, 62% for sit upright, 71% for stairs down, 58% for stairs up, 38% for walking down-ramp, 41% for walking fast, 61% for walking slow, and 70% for walking up-ramp). We get similar results from the use of automatically generated trees with tuned thresholds. Therefore, this showed that a united set of thresholds is improper for generalizing the model to other untrained data. A better way was to learn the tree structure and separating features from existing data, and then determine the thresholds for internal nodes with a short amount of observation.

2.8 Monitoring Workspace Activities using Accelerometers

2.8.1 Introduction

Physical activity monitoring (PAM) systems comprised of on-the-body accelerometers are effective tools for monitoring physical activity with medical, athletic training, and general health applications. In this section we used a PAM system for monitoring people at their workplace. Accelerometer systems have already proven their effectiveness for the physical activity classification necessary for health monitoring, successfully classifying basic physical activities including walking, jogging, and going upstairs and downstairs [40]. In this project we demonstrated that our PAM system is capable of utilizing a personalized training set easily acquired by the user in a clinical setting. We also investigated the effect of training set duration on overall classification accuracy. Previous research has used pressure sensors embedded in a worker's chair for seated posture classification [41][42], and video surveillance has been used to classify standing, sitting and lying down postures [43]. However, there has been little research concentrated on workspace activity and seated posture classification with accelerometers. These sensors are less costly for mass production than chairs equipped with embedded sensors and less invasive than video surveillance. Workplace activities that are of interest for classification in our system include walking, standing, sitting, sitting posture, and seated movements such as shaking ones leg or twisting in a chair. Studies have shown that too much sedentary behavior, such as sitting at a computer during the work day, is detrimental to health so much so that it leads to increased risks of cardiovascular disease, despite regular participation in moderate to intense exercise [44]. It has also been shown that bad workplace posture can result in widespread physical pain [45]. Fidgeting and restlessness, such as shaking ones leg and twisting in a chair, are both symptoms of anxiety, and thus the monitoring of such activities could potentially provide insight into the stress level of workers [44][46]. With the tri-axial accelerometers used in this work, employers and employees will be able to monitor exactly how much time they are spending in sedentary positions, whether or not they have proper posture when sitting, and to what extent they are exhibiting anxious physical behaviors. This data will be able to be used as a guideline for altering their behaviors for the preservation and improvement of their health.

2.8.2 System Architecture

2.8.2.1 System Components

As before, our system consists of tri-axial wireless and wearable Gulf Coast Data Concept X6-2mini accelerometer sensors. The sensors continually collect data in the X, Y, and Z directions once removed from a USB port, at a selected rate of 160Hz.

2.8.2.2 Training Data Collection

Sensors are placed on the user in specified locations (knees, chest). When collecting a training set, an easily recognizable signature (such as jumping or leaning back and forth five times) is performed before each activity. Each activity is performed for the same amount of time and its occurrence and duration is recorded, facilitating the ease of labeling the data. The orientation and location of each sensor is also recorded in order to remain consistent between data collection sessions. The workspace activities performed are walking, standing, sitting, sitting with the back reclined 85-95, sitting with the back reclined 115-125, sitting while slouching, shaking one leg while sitting, crossing one leg while sitting, swiveling in a chair, and sliding in a chair away from and towards a desk.

2.8.2.3 Scripted/Testing Data Collection

Sensors are placed in the same locations and with the same orientations as they are in training data collection. The same activities are performed but in a natural, un-planned manner. An observer (or the user) records the activities that the user performs so that the data can be labeled appropriately, creating ground truth for classification. In a deployed system, this recording would not be necessary as we would rely on the classifications, presuming they yield sufficiently accurate statistics.

2.8.2.4 Data Analysis

For the classification, we used a Naïve Bayes classifier [47] over a feature space. The features were calculated over a window of 4 seconds. The features were: mean, maximum value, and frequency energy. This was done on several levels first by classifying sitting, walking, and standing, and then within sitting, classifying the postures and movements.

2.8.3 Experiments and Results

The activities classification performed were structured into three levels. The first classification level comprised of walking, sitting, and standing, the second of seated posture, and the third of seated movements: twisting in the chair, shaking the leg, sliding to and from the desk, and crossing legs. From these data the frequency, max, min, mean, and standard deviation values were extracted on the three levels of classification. Splitting the classification into several levels has advantages for (1) producing a model that is physically understandable, and thus leading to information that is more useful in advising subjects on how to change their behavior (e.g., degree of slouch, take more breaks, etc.), and (2) reducing training time, since each decision is low-dimensional. We chose the naïve Bayes classifier [47] because it is simple and it worked well.

On the first level (walking, sitting, and standing) an accuracy of 99.5% was achieved as seen in Figure 2.5. For the posture, where the angle of the back was measured, the accuracy was 99.6%, as seen in Figure 2.6. On this level, two types of proper posture 2.6[48] were classified; reclined at 120 degrees, and upright at 105 degrees and many types of slouching were classified as improper. For the various movements while sitting, an accuracy of 96.5% was obtained, as seen in Figure 2.7.

Classification

True Class	Unknown	Walk	Stand	Sit
Unknown	5.2632	40.3509	15.7895	38.5965
Walk	0	100	0	0
Stand	0	1.0929	98.9071	0
Sit	0	0.4651	0	99.5349

Figure 2.5. Walking, siting, and standing classification statics

Classification

True Class	Unknown	Proper Reclined	Proper Upright	Improper
Unknown	0.5464	25.1366	59.5628	14.7541
Proper Reclined	0	98.4962	1.5038	0
Proper Upright	0	0	100	0
Improper	0	0	0	100

Figure 2.6. Sitting posture classification statistics

Classification

True Class	Unknown	Twist	Shake	Cross	Slide
Unknown	0.6342	1.0571	20.9302	67.6533	9.7252
Twist	0	100	0	0	0
Shake	0	0	97.3684	0	2.6316
Cross	0	0	0	92.6829	7.3171
Slide	0	2.7778	0	0	97.2222

Figure 2.7. Sitting motions classification statistics

2.8.3.1 Training Data Set Duration

An integral part of our PAM system is the training data set. In order to determine the optimal amount of time for which each activity should be performed in the training data set, a single set of data was broken up into subsets of time intervals. One time interval was dedicated as the testing data set, while the other time intervals length was varied and dedicated as the training data sets. The same testing data set was tested against each of these training data sets, and the overall accuracy for each training data set length was recorded. For walking, standing, and sitting classification, it was found that 2 minutes was an adequate training interval, as seen in Figure 2.8. For seated posture classification it was found to be 2 minutes, as seen in Figure 2.9. For seated motions it was found to be 15 seconds, as seen in Figure 2.10.

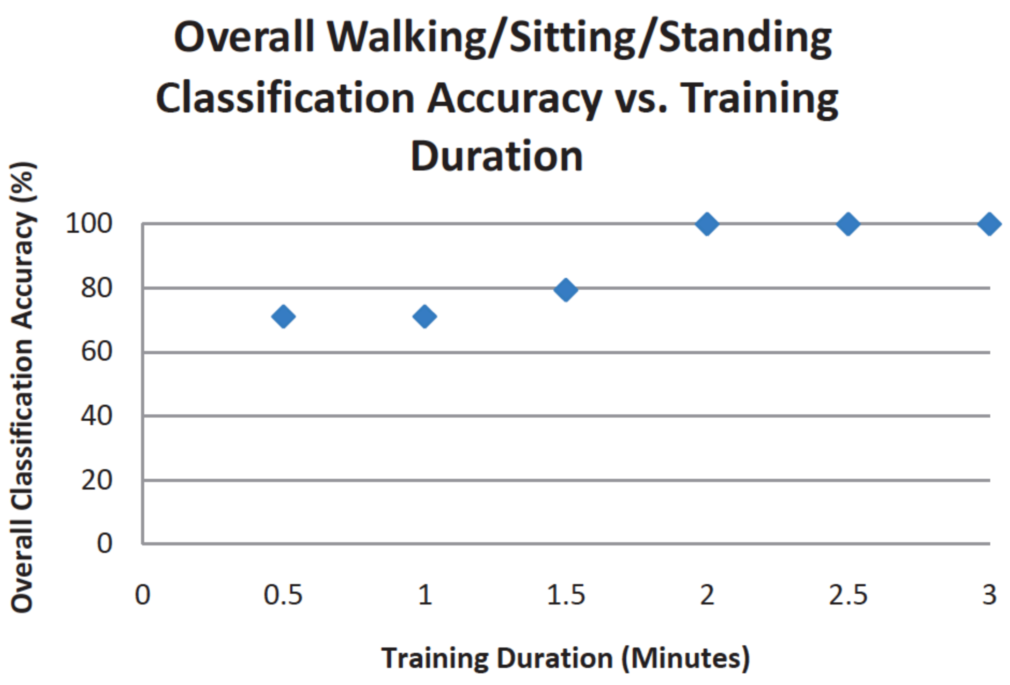


Figure 2.8. Overall classification accuracy of walking, sitting, and standing as a function of training duration

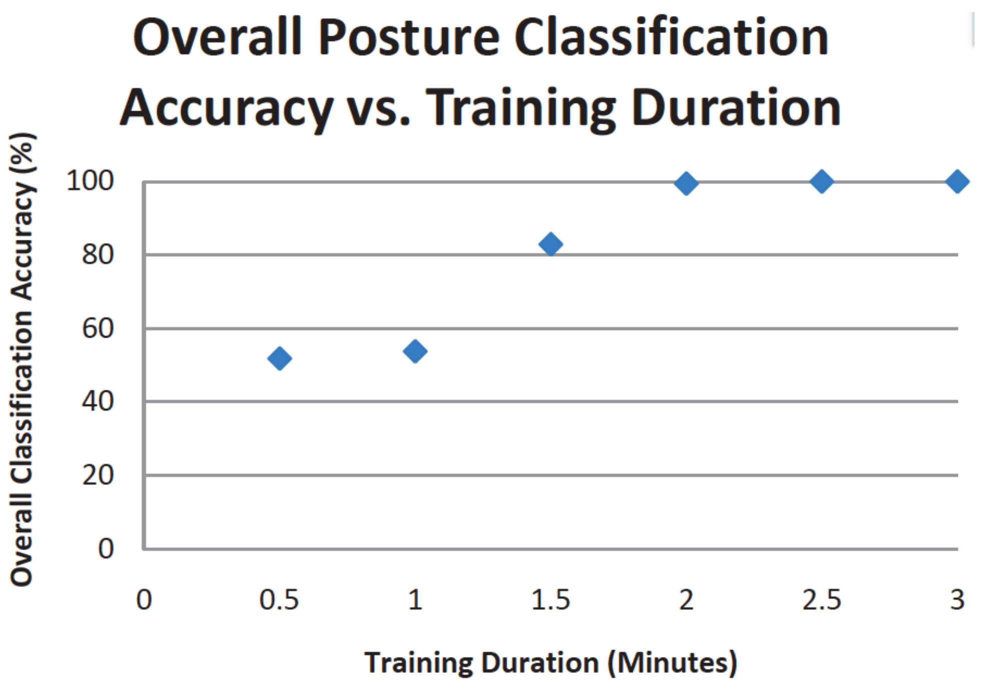


Figure 2.9. Overall classification accuracy of seated posture as a function of training duration

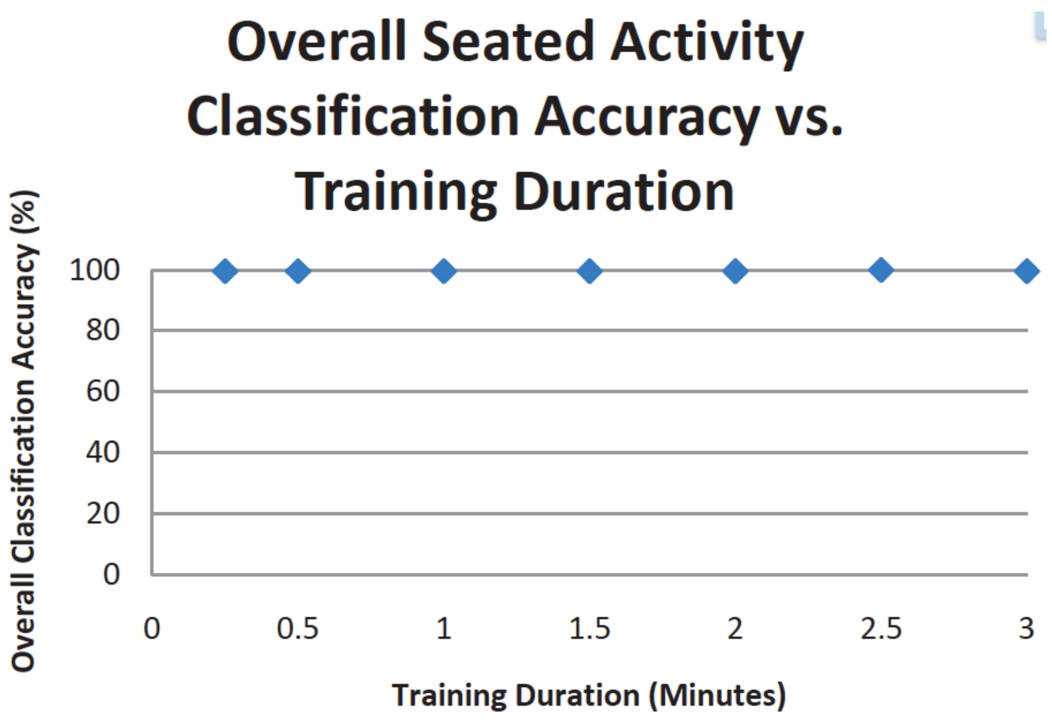


Figure 2.10. Overall seated motion classification accuracy as a function of training duration

2.8.4 Summary

We have presented a system that can accurately classify daily life activities in the work place. The systematic and simple method of training that has been developed is key. The procedures developed and results obtained allow, with research on basic activities as a foundation, for the monitoring of work place physical activity. Subjects who have expressed pain or discomfort in their body would be enabled with such a system to track their daily movement and posture without any disruption to their daily life.

Further work in two main areas is desired, expanding on posture and feature selection. We have only studied the posture of the back; whether the subject is sitting at a proper angle or slouching. However, there are many aspects to posture and proper physical motion in the workplace, such as how long subjects stare at a computer monitor, the angle of the knees, whether the subject's feet are flat on the floor, the level of the arm rest, etc. With more data collected in these areas, the applications of the system would be even greater. Another interest is in feature selection. There are many possible features to compute, and many possible sensor positions. Further analysis on the best features and combination of features is needed. There are also more feature selection algorithms that could be investigated, such as giving features weights that reflect their ability to differentiate between activities. In the future, we would like to test this system on various subjects for longer periods of time to gain a wider data pool for pattern analysis.

2.9 Estimation of Accelerometer Orientation for Activity Recognition

2.9.1 Introduction

In real world applications, many activity classification algorithms are not robust due to issues related to sensor orientation. In this section we discuss the use of personalized and supervised learning methods where a training set is used to build an activity classifier for each user. For these methods, a classifier would be built using accelerometers placed in specific orientations. The robustness issue comes into play, when there is a mismatch in the accelerometer orientation between the training, and the testing or subsequent use of the system. This is a very practical problem since the users will wear their accelerometers at different times and use their trained classifier built in a previous time. Since activity recognition algorithms are executed on training under known sensor orientations, subsequently the classifications are sensitive to those orientations as well. In order to make the systems more robust, calibration algorithms must be in place to manipulate and correct data produced by incorrectly oriented sensors. There are two traditional means for dealing with problems concerning the orientation of sensors. The first is to find orientation-invariant features, using mathematical manipulations such as power spectral density or a Fourier Transform [49]. The other is calibration through a series of movements. In this section we propose and evaluate a method to calibrate a system of sensors through a series of simple pre-defined movements. Additionally we propose an algorithm to automate the system of sensors calibration using orientation invariant motion recognition methods. This method is then tested on real data for human motion recognition.

Very few researchers have considered this problem. In [51], the authors use a similar approach but do not report the improvement one could get from such a method. The contribution of this section is that it shows the effectiveness of accelerometer orientation calibration using pre-defined movements on real data. Our results are based on real experiments using three sensors, for seven daily-life activities.

2.9.2 Methodology

2.9.2.1 System Description

We again employ the GCDC Miniature 3-axis Accelerometer Data Logger X6-2mini [53]. Our accelerometers samples at 160Hz, with a range of ±6g, recording at 16 bits of resolution. For classification purposes, the algorithms include a naïve Bayes classifier, combined with a decision tree. At each node of the decision tree, one or two mathematical features are extracted from each sensor. Features include mean values, standard deviations, and energy, among others. The specific features and activities used for experimentation purposes are discussed in the experiments and evaluation section.

2.9.2.2 Rotation Matrix Estimation Method

We use rotation matrices to calibrate the misoriented data measured by a misoriented sensor.

Each sensor measures the acceleration in a 3-D space relative to the sensor orientation. We refer to that space by sensor space. We use a reference 3-D space that corresponds to gravity, and we call it hand space. In this space gravity is aligned with the y-axis.

For each sensor, a 3x3 rotation matrix is constructed to calibrate the misorientated data. Orientations in three dimensions can be used to represent one system's orientation relative to another [52]. In this method we use a fixed system where gravity is aligned with the y-axis. In this section, it will be referred to as hand space, as we use gravity as a reference to align with the hand. The sensor has its own orientation however, which can be represented relative to the hand. Thus if we have a rotation matrix that represents the sensor in hand space, it will make the sensor data appear to come from a sensor that is aligned with the hand as shown in Figure 2.11. This is doable because all sensor data is related by an absolute, the gravity vector, as shown in Figure 2.12.

Using a feature of rotation matrices, if an inverse is performed on the 3x3 matrix of the hand in sensor space, it becomes the sensor in hand space. This rotation matrix can then be multiplied by the data being recorded by the sensor, and the sensor data can be manipulated to look as though it is being produced from a correctly oriented sensor.

2.9.2.3 Estimating the Orientation

An algorithm was developed to automate sensor calibration for systems of sensors simultaneously. Having the user perform movements shown in Figure 2.13 and Figure 2.14, an algorithm (described in detail below) recognizes those movements, records the acceleration signatures, and applies rotation matrices to correct the data. The correction motion is easy for the naïve users to perform, so that the rotation matrix can be automatically built and applied on the

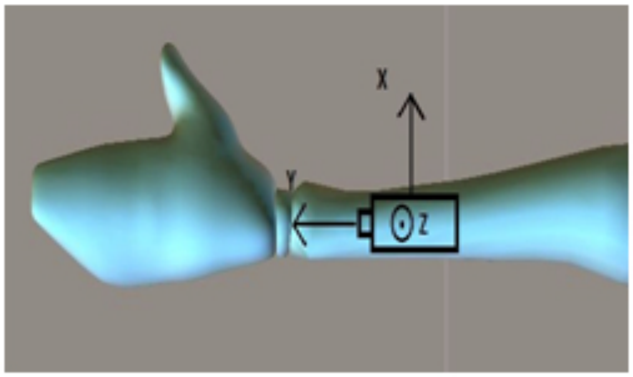


Figure 2.11. Three dimensional acceleration signals

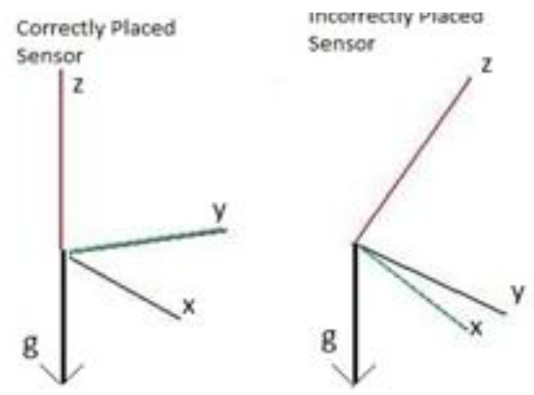


Figure 2.12. The figure on the left shows the system for a correctly placed sensor. The figure on the right shows the system for an incorrectly placed sensor

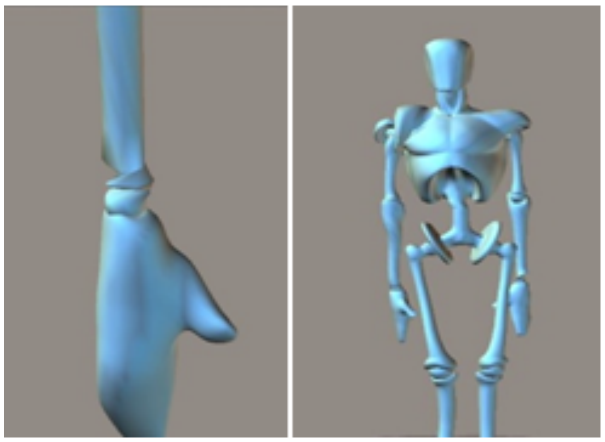


Figure 2.13. Calibration action 1

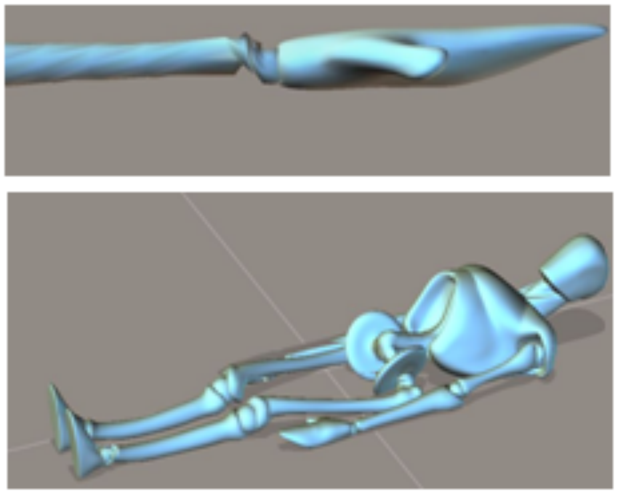


Figure 2.14. Calibration action 2

subjects' data for researchers to utilize without difficulty.

The first step is aligning the signals. To do this, the sensors are all held together in the same orientation and violently shaken. Once the time signature on all the sensors is clear, the signals are time shifted to make all movements recorded from the individual in sync. The data are also put through a low pass filter prior to processing. This ensures that shaking dynamics are kept at a minimum and tilt is emphasized. This also makes the method more robust to deal with individuals that have trouble holding still, such as Parkinson's disease. The sensors would otherwise produce sudden spikes in the data, creating a high standard deviation, giving the illusion of movement indication.

The second stage consists of finding the time period when the individual was standing upright. Regardless of orientation, the sensors must be worn flat against the skin. This ensures that if the individual is standing upright, the sensor's z-axis will always be perpendicular to gravity and read zero. The other indication that the individual is standing upright and still, is the data produced by the accelerometers will have minimal movement, indicated by a low standard deviation. Within the signal, a time frame of 10 seconds is searched for, where the accelerometers z-axis is parallel with the ground, and the individual is holding still. This is marked by an average z-reading of less than 0.2g, and a low standard deviation in **x** and **y** indicating stillness in the subject. These values are then recorded and placed into the second column of the rotation matrix.

The transitional period from standing to lying down is marked by a very high average standard deviation on the three sensors attached to the individual. The individual lying down is found by a period following the transitional period with a low standard deviation on all three axes. This ensures that as long as the individual stands upright, and then subsequently lies down, all of the needed signals will be found for rotation matrices processing.

Once these time periods are found, average values over 10 seconds are now available for each of the sensors in each of the needed axes. The values are put into a rotation matrix, inverted and then multiplied by the sensor data as described in the previous sub-section. The method is robust and user friendly, as it can automatically calibrate data, rather than having individuals finding time signals visually or recording them from an external device.

2.9.3 Results and Evaluations

2.9.3.1 Single Sensor Experiments

An initial experiment was conducted to test the effectiveness of this calibration method on the subject's wrist. One sensor was correctly oriented, while two sensors were placed in incorrect orientations on various parts of the wrist, as well as tilted to different angles. Control indicates the sensor that was correctly oriented. Experiments 1 and 2 denote the sensors that were

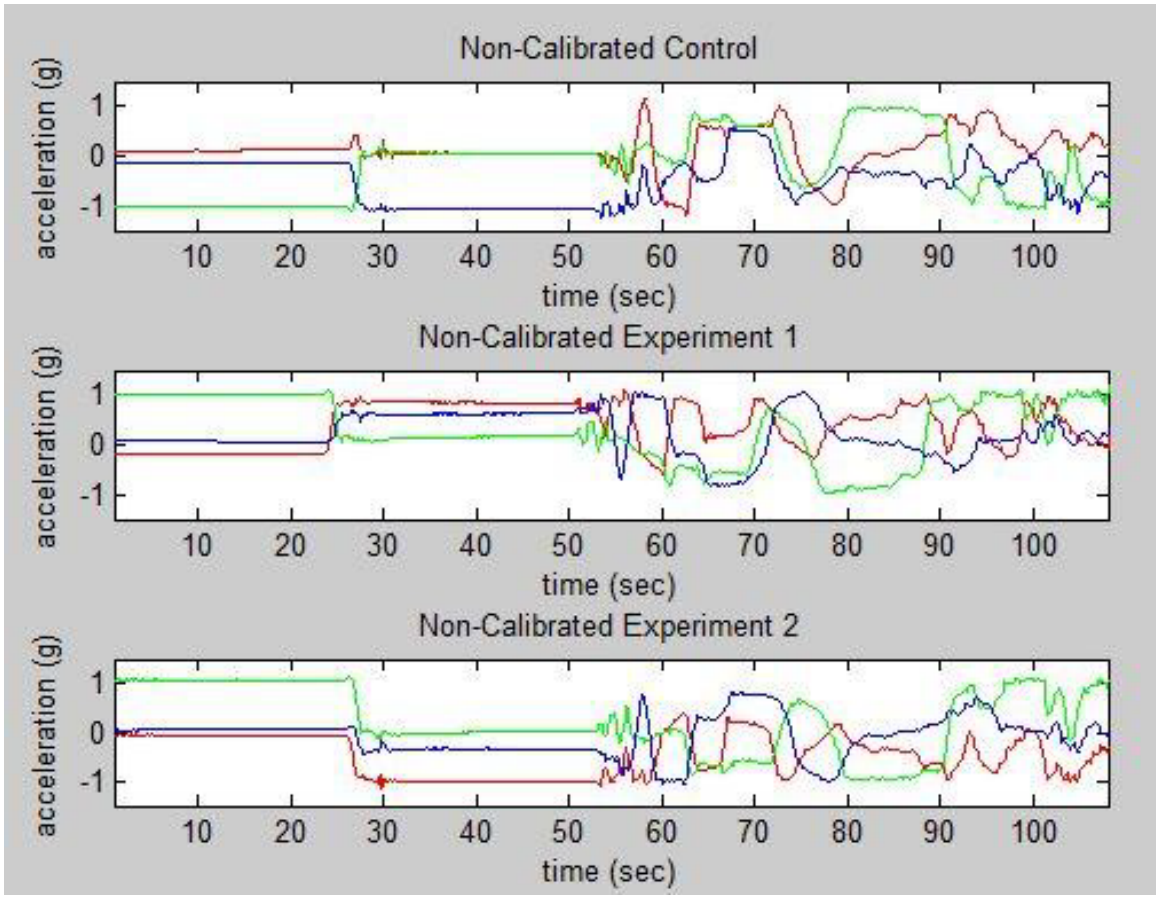


Figure 2.15. Data from non-calibrated sensors

incorrectly oriented. The x, y, and z axes are denoted as, red, green, and blue lines in that order.

A series of movements were performed, and the rotation matrices were applied via the calibration algorithm. In Figure 2.15, it is seen that the signals from the sensors are related, but yield vastly different results. In Figure 2.16 however, the signals all look almost indistinguishable from one another aside from a time delay, and the control is unchanged. These early experiments were an indication that the algorithm was successful.

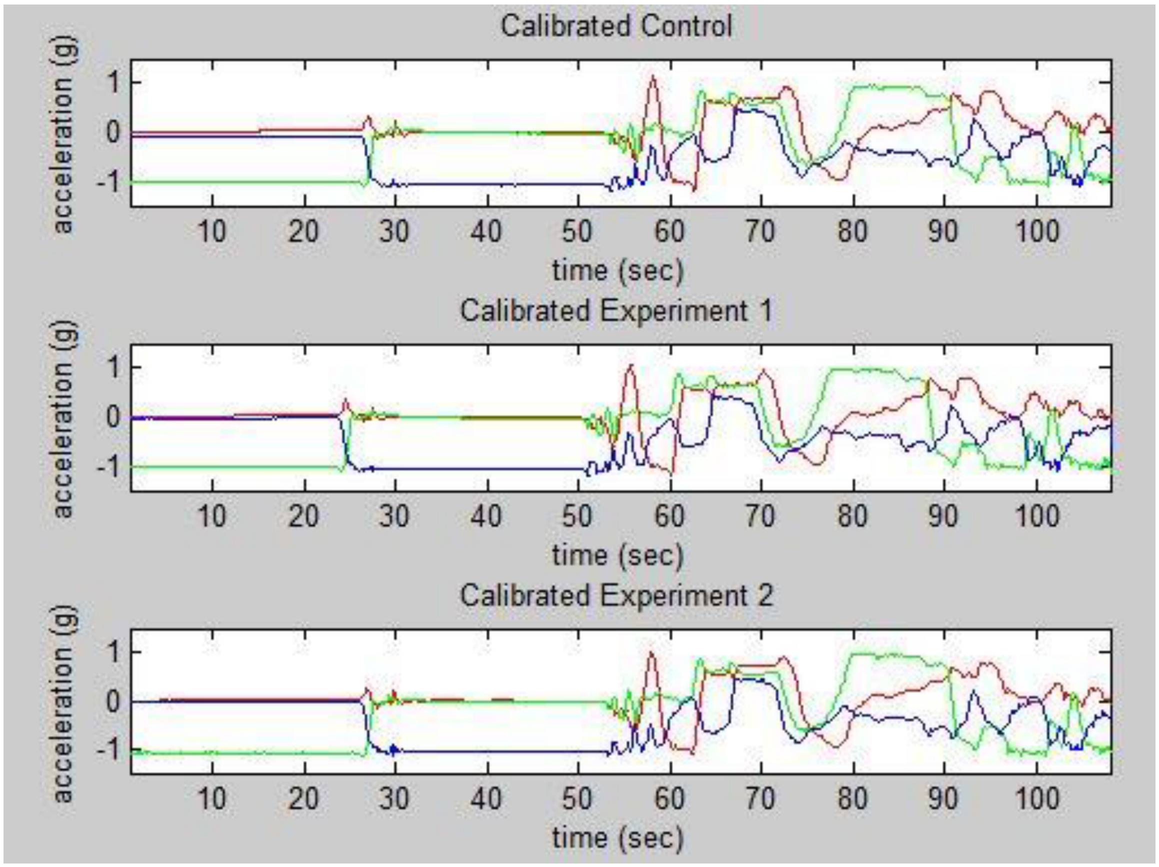


Figure 2.16. Data from calibrated sensors

2.9.3.2 Multiple Sensor Activity Classification

In this experiment, the calibration algorithm was tested on two systems of 3 sensors attached to different locations on the subject's body. Three sensors represented the control, as well as the base of the training data, and the other three are the experiment, placed at identical locations with different orientations. These locations were the right ankle, the right wrist, and the chest of the test subject. The activities being trained and classified were slow walking, running, walking up stairs, walking down stairs, sitting, lying down, and standing upright.

Our naïve Bayes decision tree classifier is shown in Figure 2.17. The first distinction between motion activities and still poses was made in the first branch. This distinction is of

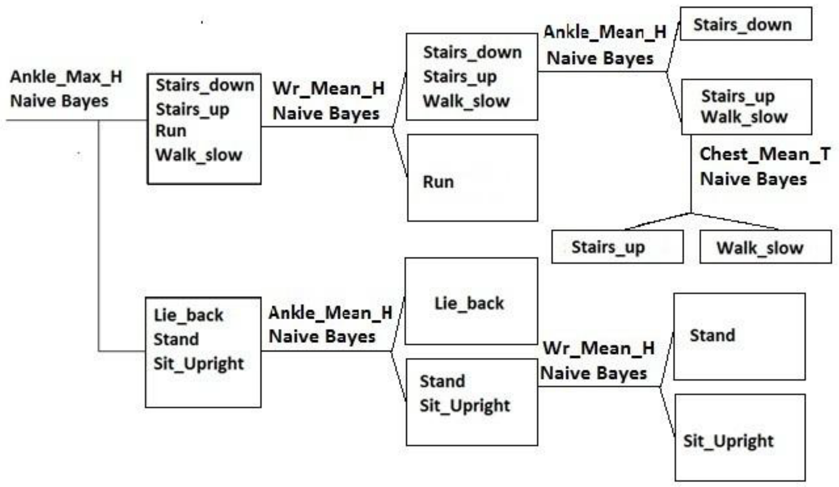


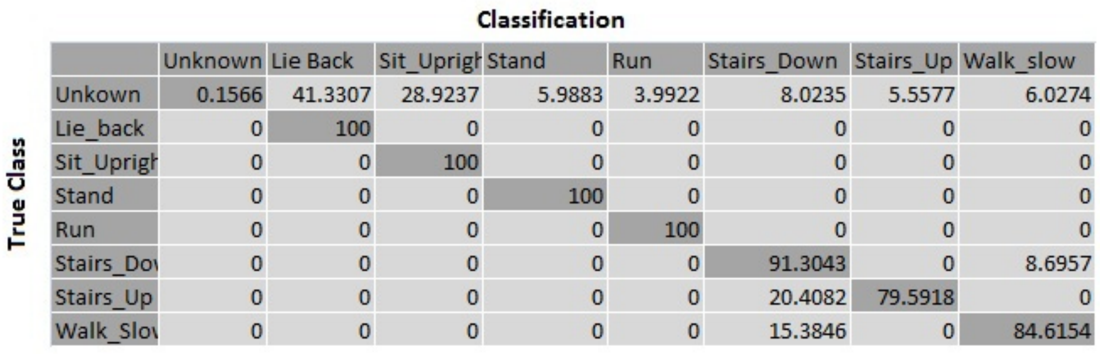
Figure 2.17. The decision tree used, the features used are shown on every node

importance to us, because still motion activities can be determined only through tilt, and are subsequently much more dependent on the orientation of sensors. Motion activities can be determined often times through motion invariant features, such as average standard deviation of the x, y, and z axes.

In this experiment the calibration algorithm was applied to two systems of 3 sensors attached to different locations on the subject's body. Three sensors represent the control, and the other three represent the experiment, as incorrectly oriented sensors. The individual wearing the sensors underwent 7 activities to be classified: slow walking, running, walking up stairs, walking down stairs, sitting, lying down, and standing. The three experimental sensors were tested for accuracy both before and after the calibration algorithms, and compared to the control experiment. The sensors were located on the right ankle, the right wrist, and the chest of the test subject.

Our naïve Bayes decision tree classifier is shown in Figure 2.17. For example, the first distinction made was between motion activities and still poses. It was found that the maximum value of the y-axis was the most accurate feature for separating these sets using cross validation. Subsequently nodes are added to the tree until all 7 activities have their distinguished sets of features.

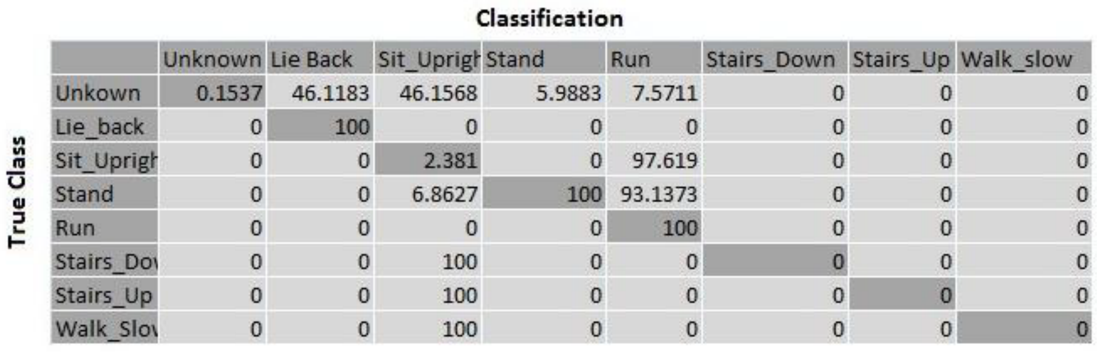
Figure 2.18 represents the control of the experiment using correctly oriented sensors. The activities were classified correctly with an accuracy of 96%. The incorrectly oriented sensors in Figure 2.19 had only 38% accuracy. Once the algorithm was run on the data, the data was again tested for activity classification and an accuracy of 93% was achieved. Also, it is clear that some activities are accurately classified regardless of orientation. The reason is that still activities are entirely orientation dependent, while mobile activities can be classified on a range of features, some being more orientation dependent than others. For example high average standard deviation can mark running, which is also rotation invariant. The data indicated that with very poor placement, the algorithm could make sensor data on average, accurate within 3% of the correctly oriented sensor data. This indicates a successful method to be used and developed further in the future.



Classification

True Class	Unknown	Lie Back	Sit_Uprigh	Stand	Run	Stairs_Down	Stairs_Up	Walk_slow
Unkown	0.1566	41.3307	28.9237	5.9883	3.9922	8.0235	5.5577	6.0274
Lie_back	0	100	0	0	0	0	0	0
Sit_Uprigh	0	0	100	0	0	0	0	0
Stand	0	0	0	100	0	0	0	0
Run	0	0	0	0	100	0	0	0
Stairs_Dov	0	0	0	0	0	91.3043	0	8.6957
Stairs_Up	0	0	0	0	0	20.4082	79.5918	0
Walk_Slov	0	0	0	0	0	15.3846	0	84.6154

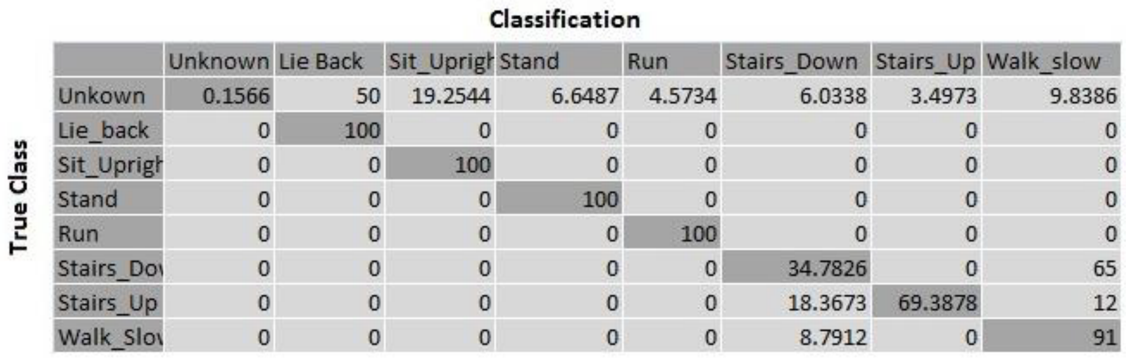
Figure 2.18. Confusion matrix for correctly oriented sensors



Classification

True Class	Unknown	Lie Back	Sit_Uprigh	Stand	Run	Stairs_Down	Stairs_Up	Walk_slow
Unkown	0.1537	46.1183	46.1568	5.9883	7.5711	0	0	0
Lie_back	0	100	0	0	0	0	0	0
Sit_Uprigh	0	0	2.381	0	97.619	0	0	0
Stand	0	0	6.8627	100	93.1373	0	0	0
Run	0	0	0	0	100	0	0	0
Stairs_Dov	0	0	100	0	0	0	0	0
Stairs_Up	0	0	100	0	0	0	0	0
Walk_Slov	0	0	100	0	0	0	0	0

Figure 2.19. Confusion matrix for non-calibrated incorrectly oriented sensors



Classification

True Class	Unknown	Lie Back	Sit_Uprigh	Stand	Run	Stairs_Down	Stairs_Up	Walk_slow
Unkown	0.1566	50	19.2544	6.6487	4.5734	6.0338	3.4973	9.8386
Lie_back	0	100	0	0	0	0	0	0
Sit_Uprigh	0	0	100	0	0	0	0	0
Stand	0	0	0	100	0	0	0	0
Run	0	0	0	0	100	0	0	0
Stairs_Dov	0	0	0	0	0	34.7826	0	65
Stairs_Up	0	0	0	0	0	18.3673	69.3878	12
Walk_Slov	0	0	0	0	0	8.7912	0	91

Figure 2.20. Confusion matrix for calibrated incorrectly oriented sensors

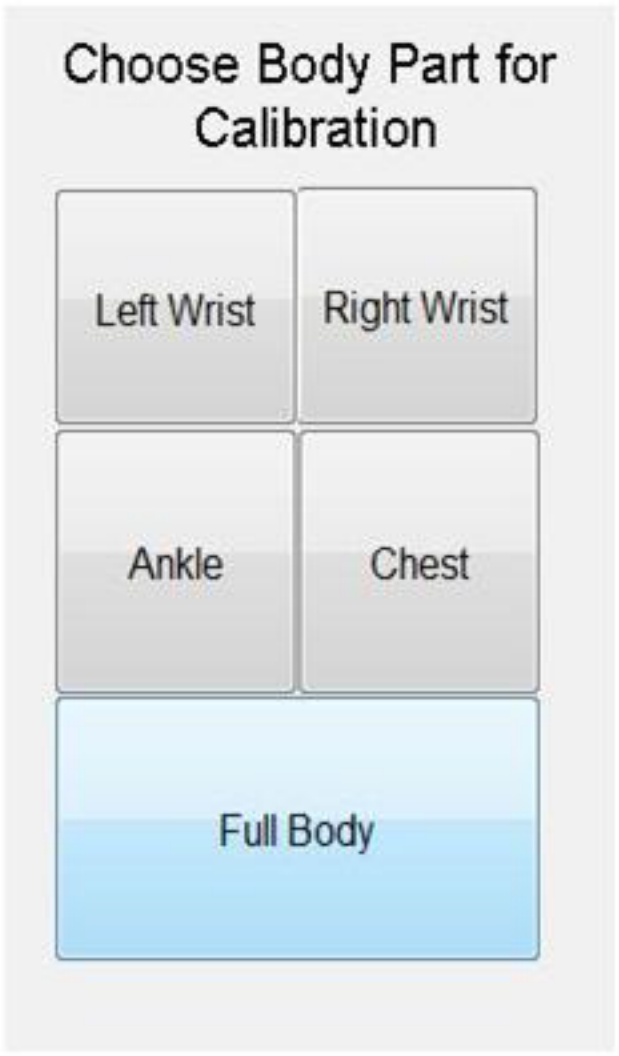


Figure 2.21. GUI home screen

2.9.3.3 Graphical User Interface

A GUI was created so that researchers could choose to calibrate individual body parts, or a system of three sensors simultaneously. After choosing the body part(s) that need calibrating, the individual selects the data file that needs to be preprocessed, and a message will appear indicating its success. Figure 2.21 is the first screen, and after selecting Full Body, Figure 2.22 is the second screen showing the positions needed for calibration and a sample to show a successful calibration.

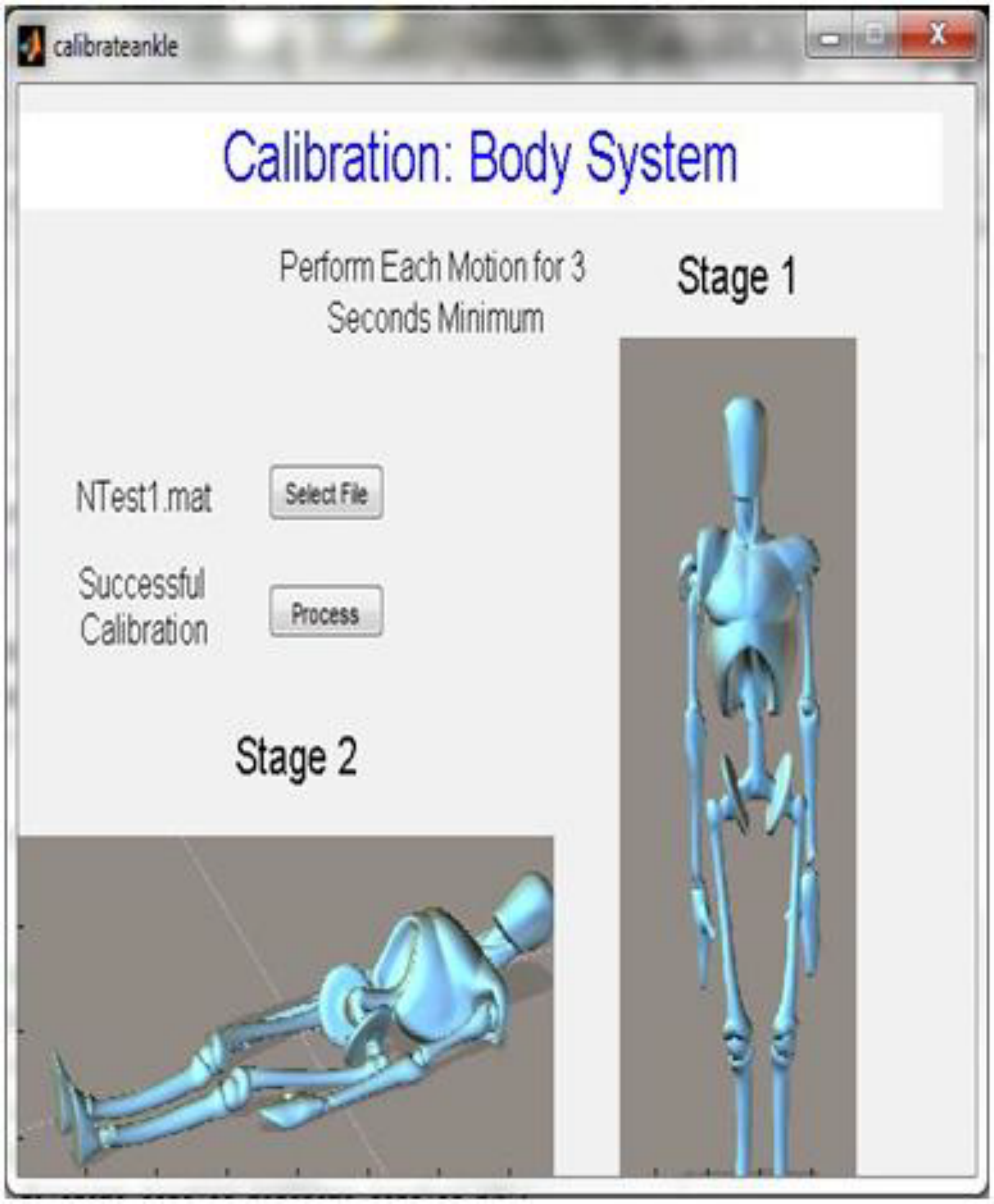


Figure 2.22. GUI final screen

2.9.4 Summary

In this section, we presented an approach for correcting the data recorded by misoriented accelerometers used for activity recognition purposes. This approach uses rotation matrices estimated from pre-defined activities done by the user at the initialization of the system. We show that it improves the accuracy from 38% to 92% for a real data set of 7 activities. Based on these promising results we are pursuing an extension to this work. This involves automatic recognition of activities that may be used for calibration of sensor orientation, rather than requiring the subject to engage in a set of prescribed activities, which may themselves be subject to error. This requires collection of large training sets over multiple subjects that include many orientation errors so that the classifier may be self-calibrating through recognition of error states. While the work involved in model creation is larger, methods that further reduce what is demanded of users may ease scaling to very large numbers.

2.10 Feature Selection Based on Mutual Information for Human Activity Recognition

2.10.1 Introduction

In prior sections we have outlined means of constructing decision trees when the number of activities and sensors is relatively small. This is the usual situation reported in the literature. For example, in [40], multi-modal sensor systems were used to classify basic physical activities, including walking, jogging, and going up and down stairs. In [4] and [56], sensor systems using only accelerometers were used for activity classification; [4] used biaxial accelerometers to monitor both ambulatory and sedentary motions, while [56] used tri-axial accelerometers to monitor workspace activities. Smart phone based accelerometers were also used for activity recognition as in [57]. A representative sampling of previous research is presented in Table 2.7.

In the research reported in this section, we aim at capturing the motions of all the parts of the body for a thorough study of the activity recognition problem. We over-instrument the subjects with 14 tri-axial accelerometers placed on various parts of the body, and we consider the

Table 2.7. Summary of previous research

Ref	Accuracy	No. Activities	No. Subjects	No. Sensors
[40]	95%	10	2	7
[4]	84%	20	20	5
[57]	85%	5	10	1
[1]	N/A	5 groups	11	6
[58]	90%	8	12	7
[59]	95%	8	7	1
[60]	65%-95%	8	1	12
[61]	90%	5	5	2

classification of 14 common daily activities. We take a supervised learning approach, using a binary decision-tree with a naïve Bayes classifier at every internal node and a large feature set of 31 features per accelerometer (total of 434 features). This is a high-dimensional problem where brute force is not possible, and a feature selection algorithm is needed to find the best features for every naïve Bayes classifier (present at every internal node). Feature selection is a problem that has been studied many times before in other contexts. Different types include margin-based algorithms such as RELIEF [62] and mutual information based algorithms such as MIFS [63]. We use a mutual information-based algorithm because it is computationally capable of handling the large amount of data captured by 14 accelerometers.

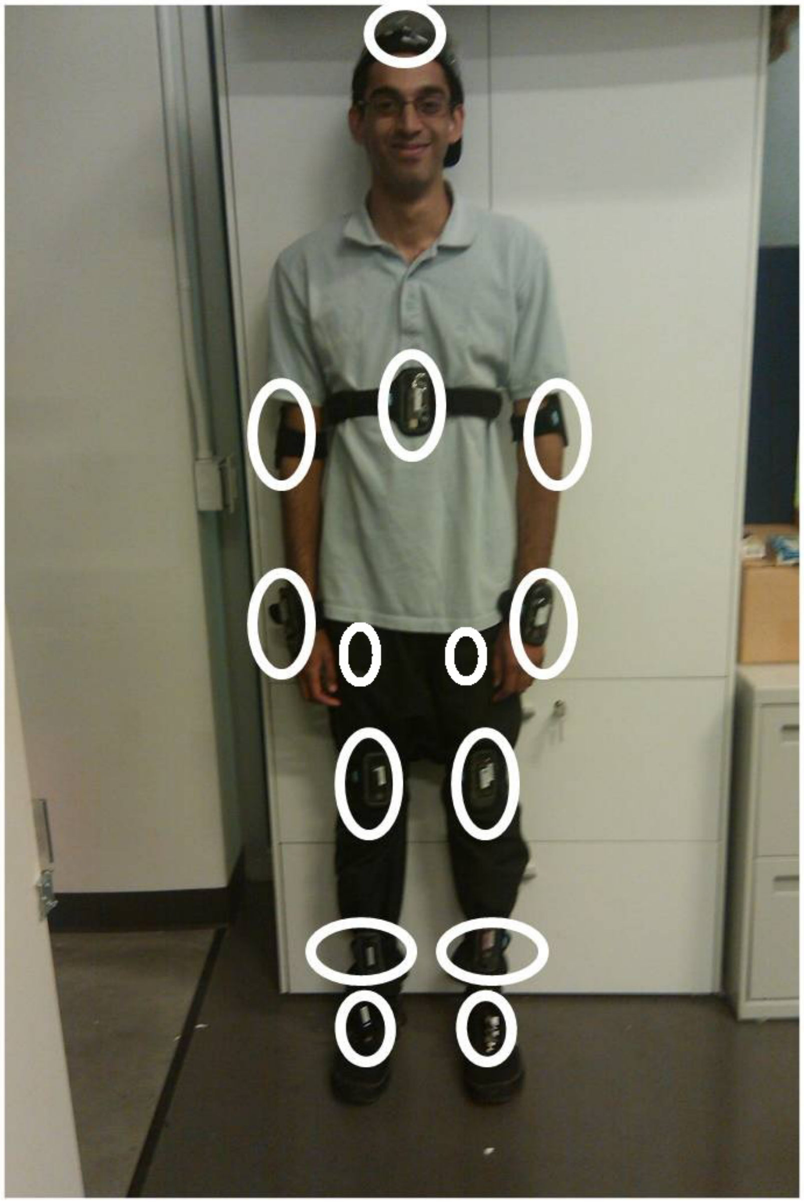


Figure 2.23. Location of the 14 accelerometers

2.10.2 Methodology

2.10.2.1 Training Data Collection

Accelerometers are placed on an individual at fourteen locations, as shown in Figure 2.23. The accelerometers we used were tri-axial wireless Gulf Coast Data Concept X6-2mini

Table 2.8. The 14 activities that were classified

Active	**Stationary**
Slow walk	Stand
Fast walk	Sit (upright)
Walk (up-slope)	Sit (hunch)
Walk (down-slope)	Sit (slouch)
Walk (up stairs)	Lie down (on back)
Walk (down stairs)	Lie down (on stomach)
Run	Lie down (on side)

accelerometers (±6g) [53], which continually collected data at a rate of 160Hz. Fourteen different activities are performed, as described in Table 2.8. To collect labels for ground truth, we used an Android phone application. The application has a list of the activities to choose from and a start/stop button to record the time the subject started the activity, and the time he/she stopped. Eight different data sets were collected from eight different healthy individuals for a length of five minutes per activity.

2.10.2.2 Features Computation

Features were computed on 4-second windows of acceleration data with 3 second overlapping between consecutive windows. We compute 31 different features for each sensor, shown in Table 2.9. Since we used 14 different sensors, this meant a total of 434 features from which to choose.

Table 2.9. Features used (m refers to the magnitude of the 3D acceleration vector)

Feature
Standard deviation of x,y,z axes and m
Mean of x,y,z axes and m
Absolute mean of x,y,z axes and m
Energy ratio of x,y,z axes and m
Ratio of DC to sidelobe of x,y,z axes
First sidelobe location of x,y,z axes
Max value of x,y,z axes and m
Short time energy in x,y,z axes and m
Correlation between x and y axes

2.10.2.3 Classification

We used the binary decision tree shown in Figure 2.24, with a naïve Bayes classifier at each node. The naïve Bayes classifier is a probabilistic method given by the function (2.6), where $\mathcal{C}$ is the set of classes and $\mathcal{F}$ is the set of features.

$$\max_{C \in \mathcal{C}} \left\{ p(C) \prod_{f \in \mathcal{F}} p(f|C) \right\} \tag{2.6}$$

This classification was performed offline. A tree was used so that the classifier would not have to distinguish between all 14 of the classes using the same set of features. Instead, classifiers are used to partition the data into smaller and smaller categories of classes until the categories

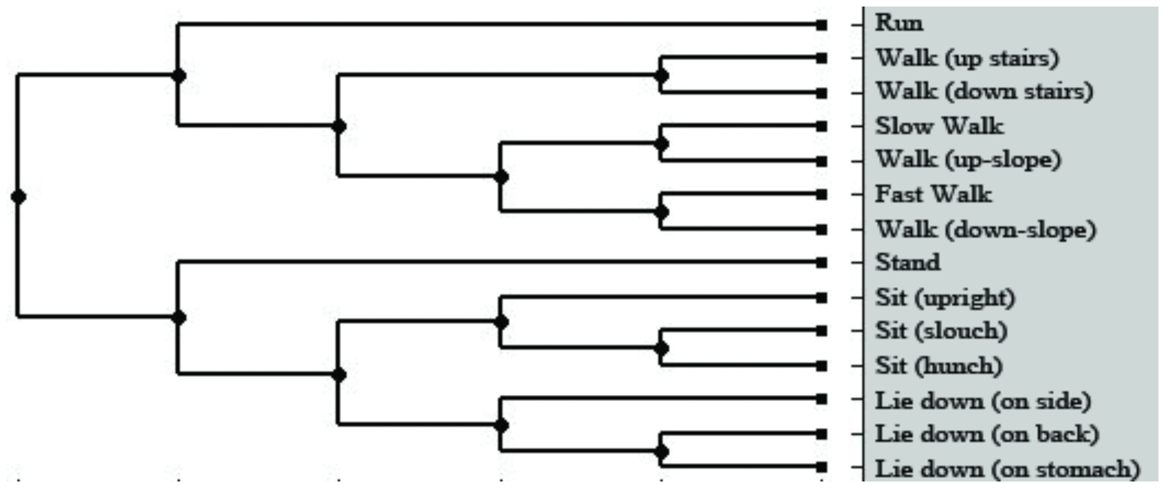


Figure 2.24. Decision tree used

consist of a single class, at which point the data is fully classified. In the probability calculations (given by Bayes' rule), the features were assumed to be independent with a Gaussian distribution, as required by the naïve Bayes classifier. For every subject, the naïve Bayes classifiers (at the internal nodes of the tree) were trained on his/her training data; this is often called a user dependent procedure. The feature selection was also personalized to every subject.

2.10.2.4 Feature Selection Algorithm

The high-dimensionality of the problem requires a good feature selection algorithm to find the best features for the naïve Bayes classifier at every internal node. In order to minimize computational complexity while maximizing accuracy, this algorithm employs a 'filter' solution first, and then a 'wrapper.' The algorithm works as follows:

1. We determine the Gaussianity of each feature by calculating the negentropy of each feature given each class using the approximation given in equation (2.7), where J is the negentropy, x is a random variable, E is the expected value, and $kurt$ is the kurtosis, the fourth central moment of the [64][65].

$$J\left(x\right) \approx \frac{1}{12}E\left\{x^3\right\}^2 + \frac{1}{48}kurt\left(x\right)^2 \tag{2.7}$$

We remove all features with negentropy values that are higher than an a priori threshold.

2. Using the mRMR algorithm, we ranked the features. [66] The term this algorithm wishes to maximize is given by formula (2.8).

$$I\left(C;f_i\right) - \frac{1}{\left|S\right|}\sum_{f_s \in S} I\left(f_s;f_i\right) \tag{2.8}$$

I is the mutual information, C is the class variable, f_i is the feature under consideration, and S is the set of features already selected and ranked. Calculating mutual information requires calculating the entropy of a feature or set of features, a computationally expensive process because each feature is a mixture of Gaussians. Hence a Taylor series approximation of the entropy was employed[3].

3. By now, there are a few parameters that can be changed: the threshold for the negentropy values and the degree of the Taylor series approximation. In addition, there are really two different possible algorithms, using only the first term of (3) (Max-Relevance), or both (Max-Relevance and Min-Redundancy) [66]. Instead of choosing one algorithm, or just one set of parameters, a range of parameters are used over both algorithms, and the sets of features returned by these algorithms are captured. Because we wish to minimize the number of features, we use the first **k** features in each ranking, where **k** ranges from **1** to the full set.

4. This gives us a list of feature sets. We pick the feature set that minimizes the training error[1].
5. The above steps are repeated for each node in the tree. Then for each node, the highest ranking set of features are chosen, and the total number of sensors used so far is updated.

2.10.3 Results

We collected sets of data from eight different individuals where the participant did five minutes of each of the 14 activities while wearing all 14 accelerometers. For these data sets, we trained on half the data, 2.5 minutes per activity, and tested on the other half, a time suggested by [40]. We got an average overall accuracy of 96.5%, as seen in Table 2.10. These results show that a high activity recognition rate is achievable for a large set of activities. Table 2.11 shows that a large number of sensors was used for every subject. This is due to the fact that the feature selection algorithm does not take into consideration from which sensor the features were selected. It would be interesting to change the feature selection algorithm to a sensor selection algorithm, while maintaining a relatively high accuracy. This could be done by adding a term to favor features from the same sensors. It is also worth noting that for different subjects, different features were selected. This is due to the variation in the acceleration data belonging to different subjects (e.g. different subjects walk differently, sit differently, and lie differently.).

[1] This corresponds to choosing the feature set that gives the highest discrimination between the two branches of the tree at the corresponding node. Training error is the percentage of misclassified training data

Table 2.10. Average accuracy for each of the activities

Activity	**Percent Correct**
Run	100%
Walk (up stairs)	97.67%
Walk (down stairs)	94.54%
Slow walk	92.77%
Walk (up-slope)	95.95%
Fast walk	96.81%
Walk (down-slope)	95.32%
Stand	99.41%
Sit (upright)	89.90%
Sit (slouch)	94.62%
Sit (hunch)	99.24%
Lie down (on side)	100%
Lie down (on back)	94.83%
Lie down (on stomach)	99.66%

Table 2.11. Average accuracy for each of the test subjects

Test subject	**Our algorithm**	**Number of sensors**
1	93%	10
2	98%	12
3	96%	9
4	97%	10
5	98%	12
6	99%	10
7	96%	13
8	95%	11
Average	96.5%	10.875

2.10.4 Summary

This section presents a combination of a tree-based classification and a feature selection algorithm for human activity recognition, and shows that a high activity recognition rate is achievable for a large set of common daily activities. More than just a specific algorithm, this section presents a framework that maximizes the accuracy that can be garnered from the results of specific algorithms, like the mRMR algorithm that we used. This work shows that different sensors (at different locations on the body) are the best for discriminating between subset of the activities. The algorithm presented could be changed to minimize for the number of sensor used. This is a step forward towards understanding human activities and towards finding the best placements of sensors on the body for the recognition of a large set of activities.

2.11 Conclusion

In this chapter we have presented a variety of applications of decision trees to the problem of activity classification. In some cases, the problem addressed was simple enough that naïve Bayes classifiers were sufficient, but in general more sophisticated approaches are required to enable good classification accuracy with limited human effort in providing ground truth. The hybrid tree we have introduced provides exactly this functionality so that the overall effort in developing models that can be tuned to individuals is reduced.

Chapter 3

MOTION TRAJECTORY FUSION USING DIVERSE SENSOR TYPES

3.1 Introduction

Inertial measurement units (IMU) are widely used due to their low cost, low weight and small size. They are now implemented in numerous fields including aviation, robotics, gaming, sports and others to measure orientations or directions[68][69][70][71]. Some studies also utilized inertial sensing to classify human activities or reconstruct human motions [72][73][74][75][76][77].

Generally used IMUs include accelerometers, gyros, magnetometers, GPS and other devices. Due to their physical characteristics and numerical data manipulating procedures, estimation results using these devices suffer from high measurement noise, incorrect scaling and biasing. Therefore, there are many studies discussing how to model measurement errors and drift using various filters and algorithms [78][79][80].

Our goal is to estimate the orientations of upper limbs at any given moment to find the motion trajectories of the arm. This will benefit medical-field studies, which focus on long-term and detailed movement monitoring. For conditions such as Parkinson's disease or rehabilitation from injuries, doctors and therapists usually need to watch tiny changes of patients' motions for a period of time. If there exists a system composed of IMUs, which can tell them of any significant changes of the motions at patients' home environments, it would greatly benefit doctors' diagnosis and save a huge amount of medical resources through timely interventions.

Much research has been conducted to reconstruct trajectories, or to estimate sensor orientations. In [76] kinematic models were combined with unscented Kalman filters to estimate orientations of joints under slow and fast motions. However only simple arm movements were evaluated. In [77], a continuous-wavelet-transform based method was used to integrate accelerometer data analytically to avoid numerical integration drifts, in which subjects only performed motions slowly, and some reconstructed patterns are only recognizable but not accurate.

In [81], biomechanical models and non-linear complementary filters were combined to estimate upper body motion. However, all experiments were done outdoors where there is less magnetometer interference.

In this chapter, we show how to estimate motion trajectories by combining non-linear complementary filter designs, which estimate orientations and gyro bias [70][82], with biomechanical models of upper limbs, including limb decomposition and human motion limitations. From different studies, we have seen that different environments require different motion reconstruction strategies. For example, we know that in an indoor environment, the magnetometer suffers from large amounts of magnetic interference, which will affect its measurements. In addition, in an outdoor environment, though the magnetometer suffers from less interference, there are also some limitations which affect the robustness of our methods: there may be limited wireless internet connection which is necessary to upload and transfer recorded data between recording devices, and the devices need to be less dependent on a power source. For these reasons, in this chapter we formulated a system, which can track human motion trajectories under various environments. Experiments were conducted covering arm movements, pattern drawing and daily life activities. This method not only applies to upper limb motion reconstruction, but can be also used to estimate orientation of lower limbs with the appropriate kinematic model.

In section 3.2 we introduce the system for estimating motion trajectories using various types of sensors. Section 3.3 introduces the experimental and simulation setups. The results are presented in section 3.4, followed by the discussions of the results in section 3.5, and we draw the conclusions in section 3.6.

3.2 Algorithms

3.2.1 Definitions and building blocks

3.2.1.1 Notation and measurement modeling

3.2.1.1.1 Special rotation group

In geometry, any orientation and rotation can be expressed as a vector $\upsilon \in \mathbb{R}^3$, where it contains the yaw, pitch and roll angles of that rotation. We can also define orientations and rotations in matrix form. $R \in \mathbb{R}^{3\times3}$ belongs to the special orthogonal group $\mathrm{SO}(3)$ such that

$$RR^T = I, \qquad R \in \mathbb{R}^{3\times3} \tag{3.1}$$

We then define the operator $\vee : \mathbb{R}^3 \to \mathrm{SO}(3)$ such that

$$\upsilon_\vee = \begin{pmatrix} 0 & -\upsilon_3 & \upsilon_2 \\ \upsilon_3 & 0 & -\upsilon_1 \\ -\upsilon_2 & \upsilon_1 & 0 \end{pmatrix} \qquad \upsilon \in \mathbb{R}^3, \upsilon_\vee \in \mathrm{SO}(3) \tag{3.2}$$

Then for a rotation ω and any vector $v \in \mathbb{R}^3$, we have

$$\omega_{\vee} v = \omega \times v \tag{3.3}$$

where $\times$ is the vector cross product. The following identity is also commonly used in this chapter:

$$\left(Ra\right)_{\vee} = Ra_{\vee}R^{T}, \qquad R \in \mathrm{SO}\left(3\right), a \in \mathbb{R}^{3} \tag{3.4}$$

where R is a rotation matrix, and a represents an orientation in 3-D space.

We then define $\mathrm{vex} : \mathrm{SO}\left(3\right) \rightarrow \mathbb{R}^{3}$ being the inverse of $\vee$. That is to say, we have

$$\begin{aligned} \left(\mathrm{vex}\left(R\right)\right)_{\vee} &= R, \qquad && R \in \mathrm{SO}\left(3\right) \\ \mathrm{vex}\left(v_{\vee}\right) &= v, \qquad && v \in \mathbb{R}^{3} \end{aligned} \tag{3.5}$$

3.2.1.1.2 Coordinate systems (Frames of reference)

We use the following notation to represent different frames of reference

$\left\{A\right\}$: Earth frame of reference (Earth coordinate).

$\left\{B\right\}$: Body frame of reference (Body coordinate)

$\left\{E\right\}$: Estimator frame of reference (Estimator coordinate).

$\left\{K\right\}$: Kinect frame of reference (Kinect coordinate).

We use x, y, and z to represent the axie of north, east, and down (NED) coordinate system in $\left\{A\right\}$. $\left\{B\right\}$ is the body coordinate, and is used to denote the original data of the IMU.

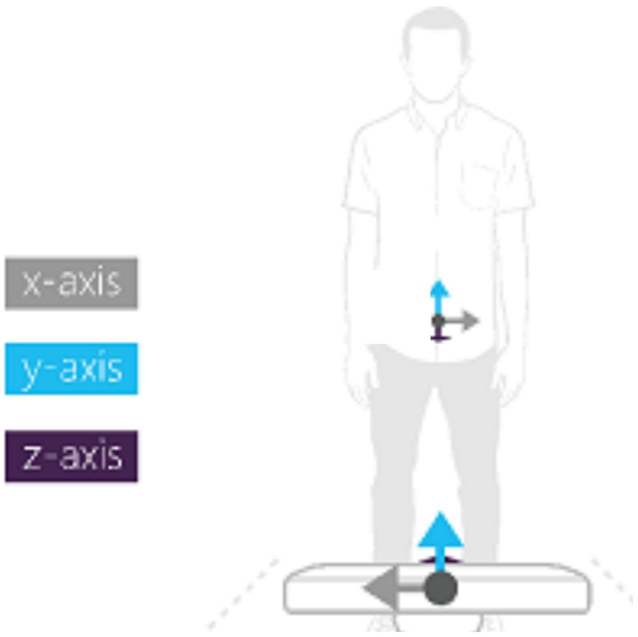


Figure 3.1. Kinect axes definition

$\{E\}$ is the estimator coordinate, which is used to represent the estimation of the IMU orientation. $\{K\}$ is the Kinect frame of reference, whose x, y and z axes are defined from the view of the camera of the Kinect as shown in Figure 3.1.

Unless otherwise specified, we use left superscripts and subscripts to describe orientations and rotations expressed in different frames of reference; we use subscripts to describe the origin frame of reference, and superscript to represent the destination frame of reference. For instance, ${}_{B}v$ represents the orientation v relative to the body frame; ${}^{A}_{B}R:\{B\}\rightarrow\{A\}$ represents the rotation matrix R from the Earth frame to the body frame.

3.2.1.1.3 Measurement models

In this section we model measurements from MEMS sensors and the Kinect in the following ways.

3.2.1.1.3.1 Accelerometers

Let the noisy measurements of accelerometers measured in the body frame be denoted by $\tilde{a} \in \mathbb{R}^3$, and the true measurement by $a \in \mathbb{R}^3$. We model the relation of collected signals to their true values by

$$_B\tilde{a} = M_a\, {}_B a + b_a + n_a \tag{3.6}$$

where $M_a \in \mathbb{R}^{3\times 3}$ is the combination of sensitivity and misalignment of the accelerometer axes, $b_a \in \mathbb{R}^3$ is the constant bias of the measurements, and n_a is the zero-mean additive white Gaussian noise (AWGN).

It is important that we distinguish gravity and the force applied to the sensor. Let the Earth's gravitational acceleration field in the Earth frame be denoted by ${}_A g_0$, and the instantaneous acceleration applied to the sensor in Earth frame by ${}_A\dot{v}$. We can then describe the ideal accelerometer measurements ${}_B\tilde{a}$ by

$$_B\tilde{a} = M_a \cdot R\left({}_A\dot{v} - {}_A g_0\right) + b_a + n_a \ . \tag{3.7}$$

When the sensor moves slowly, we have $\left\|{}_A\dot{v}\right\| \approx 0$; then the accelerometer measurements are roughly equal to the gravitational acceleration. Therefore, we can express the normalized ideal measurements by

$$\frac{{}_B a}{\left\|{}_B a\right\|} = {}^A_B R \left\|{}_A g_0\right\| e_3 \tag{3.8}$$

where $e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ in NED coordinates.

3.2.1.1.3.2 Gyros

We describe the noisy gyro measurements ${}_B\tilde{w} \in \mathbb{R}^3$ by

$$ {}_B\tilde{w} = M_g \, {}_B w + b_g + n_g \tag{3.9} $$

where $M_a \in \mathbb{R}^{3\times3}$ is the combination of sensitivity and misalignment of the gyro, ${}_B w \in \mathbb{R}^3$ is the true gyro rotation along its axes, $b_g \in \mathbb{R}^3$ is the constant bias of the measurements, and $n_g \in \mathbb{R}^3$ is the zero-mean AWGN. If we want to integrate the gyro measurements of angular velocity to find the orientation, we have to notice the bias term b_g will accumulate as time goes by.

3.2.1.1.3.3 Magnetometers

Similar to previous sections, we describe the noisy magnetometer measurements ${}_B\tilde{m} \in \mathbb{R}^3$ by

$$ {}_B\tilde{m} = M_m \, {}_B m + b_m + n_m \tag{3.10} $$

where $M_m \in \mathbb{R}^{3\times3}$ is the combination of sensitivity and misalignment of the magnetometer, ${}_B m \in \mathbb{R}^3$ is the Earth magnetic field being projected on the magnetometer axes, $b_m \in \mathbb{R}^3$ is the constant bias of the measurements, and $n_m \in \mathbb{R}^3$ is the zero-mean AWGN.

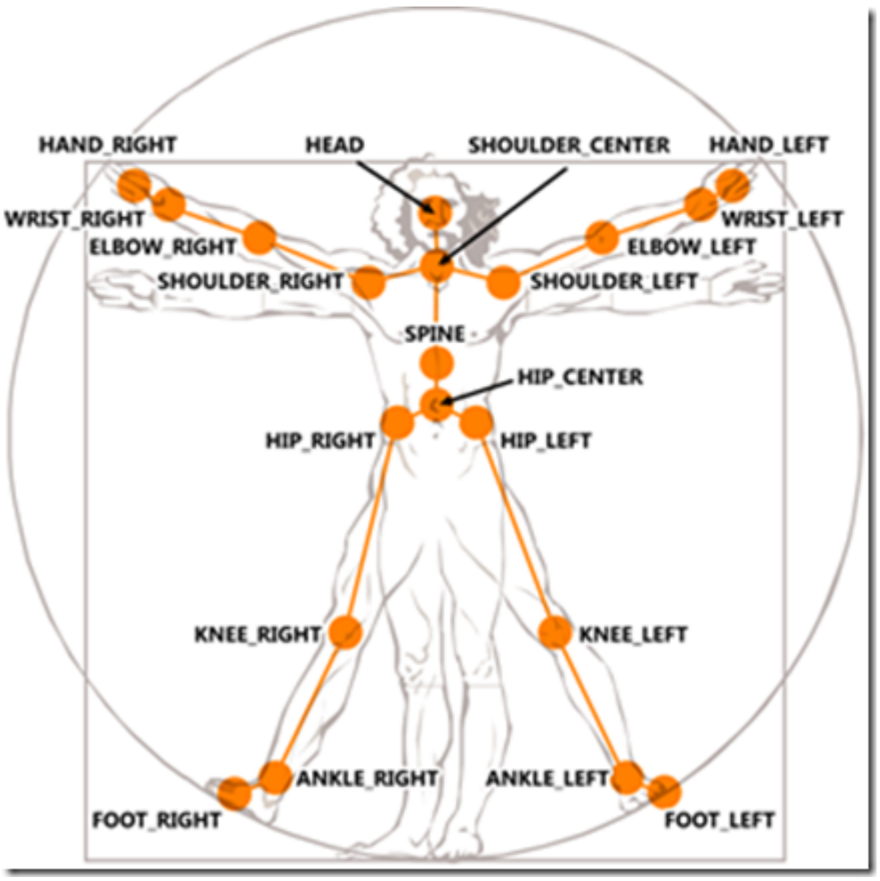


Figure 3.2. Joint information provided by Kinect

3.2.1.1.3.4 Kinect

The released Kinect for Windows Software Development Kit (SDK) [23] provides the position information of 20 human joints as shown in Figure 3.2, and Table 3.1 shows the joint numbers. We model the i^{th} Kinect position measurement in the Kinect frame of reference ${}_K\tilde{p}_i \in \mathbb{R}^3$ by

$$ {}_K\tilde{p}_i = {}_K p_i + n_i \tag{3.11} $$

where ${}_K p_i \in \mathbb{R}^3$ is the real distances of the joint to Kinect camera, and $n_i \in \mathbb{R}^3$ is the zero-mean AWGN.

Table 3.1. Number of joints

#	Part	#	Part	#	Part	#	Part	#	Part
1	Hip center	5	Shoulder left	9	Shoulder right	13	Hip left	17	Hip right
2	Spine	6	Elbow left	10	Elbow right	14	Knee left	18	Knee right
3	Shoulder center	7	Wrist left	11	Wrist right	15	Ankle left	19	Ankle right
4	Head	8	Hand left	12	Hand right	16	Foot left	20	Foot right

3.2.1.2 Non-linear Complementary Filters with Bias Estimation

In this section, we introduce our method for finding the orientation of the sensor, namely, non-linear complementary filters with bias estimation. From [70][82], this filter applies a low pass filter and a high pass filter to two signals and then fuses them to get better estimates. In our situation, we have a static estimate of the orientation which is accurate when the subject moves slowly, but inaccurate when moving fast; we also have a dynamic estimate of the orientation which tells instant changes of the orientation, but accumulates errors when integrating the dynamics. The complementary filter will apply a low pass filter to the static estimate, and a high pass filter to the dynamic estimate to fuse both estimates to acquire a better orientation estimate. We start by defining the error measurement of an orientation, and then introduce the static and the dynamic estimation processes for the orientation, and finally explain how we fuse these measurements using non-linear complementary filters.

3.2.1.2.1 *Estimate of Sensor Orientations and Error Measurement*

We define

$$ {}^{A}_{E}\hat{R} : \{E\} \to \{A\}, \qquad {}^{A}_{E}\hat{R} \in \mathbb{R}^{3\times 3} $$

to be the estimated orientation of the sensors from the estimator frame to the Earth frame. This orientation should be close to the true orientation from the body frame to the Earth frame, which is ${}^{A}_{B}R$. We define the relative rotation of ${}^{A}_{E}\hat{R}$ and ${}^{A}_{B}R$ by

$$ {}^{E}_{B}\tilde{R} = {}^{A}_{E}\hat{R}^{T}\, {}^{A}_{B}R $$

With Lyapunov stability analysis, we can define the estimation error by

$$ \begin{aligned} err &= \frac{1}{4}\left\| I - {}^{E}_{B}\tilde{R} \right\|^{2} \\ &= \frac{1}{2}\operatorname{tr}\left(I - {}^{E}_{B}\tilde{R} \right) \\ &= 1 - \cos(\theta) \end{aligned} \tag{3.12} $$

where θ is the angle of rotation from the $\{B\}$ frame to $\{E\}$ frame. Once the estimate of orientation is close to the true value, we have two frames overlapped and thus $\theta = 0$, and therefore from (3.12) the estimation error goes to zero.

3.2.1.2.2 *Static and Dynamic Estimation of the Orientation*

Ideally, any two nonparallel measurements ${}_{B}\upsilon_1, {}_{B}\upsilon_2$ measured in the body frame and their corresponding value in the Earth frame ${}_{A}\upsilon_1, {}_{A}\upsilon_2$ can be used to calculate the orientation of a rigid body. We call this estimate the static orientation ${}_{E}^{A}\hat{R}_{s}$. From [70], this static estimate can be acquired using the following optimization formula,

$${}_{E}^{A}\hat{R}_{s} = \underset{R\in \mathrm{SO}(3)}{\arg\min}\left(\lambda_1 \left\| \frac{R\cdot {}_{B}v_1}{{}_{B}v_1} - \frac{{}_{A}v_1}{{}_{A}v_1} \right\|^2 + \lambda_2 \left\| \frac{R\cdot {}_{B}v_2}{{}_{B}v_2} - \frac{{}_{A}v_2}{{}_{A}v_2} \right\|^2 \right) \tag{3.13}$$

where the weightings λ_1, λ_2 are chosen depending on the confidence of the sensor outputs.

Also, if we have measurements of the angular velocities of a rigid body $\omega \in \mathbb{R}^3$, we can estimate the dynamic orientations ${}_{E}^{A}\hat{R}_{d}$ from the rotational kinematics by solving the following differential equation

$$\begin{aligned} \frac{\partial}{\partial t}\, {}_{E}^{A}\hat{R}_{d} &= {}_{E}^{A}\hat{R}_{d} \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix} \\ &= {}_{E}^{A}\hat{R}_{d} \cdot \omega_{\vee} \end{aligned} \tag{3.14}$$

3.2.1.2.3 *Direct and Passive Complementary Filters*

The rotation error between estimation ${}_{E}^{A}\hat{R}$ and ground truth ${}_{B}^{A}R$ can be expressed as $\tilde{R} = {}_{E}^{A}\hat{R}^{T}\, {}_{B}^{A}R$. Based on [82][83], we define the correction term

$$\sigma = \mathrm{vex}\left(\frac{1}{2}\left(\tilde{R}^{T} - \tilde{R} \right) \right) \in \mathbb{R}^3 \tag{3.15}$$

We use σ to represent error between the estimated orientation and the true orientation. When the estimate is equal to the truth, we have $\tilde{R} = I$, and thus $\sigma = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{T}$. With the above definitions, we then fuse static and dynamic estimates to derive the final estimate of the orientation.

From (3.4) and (3.14) we have

$$\frac{\partial}{\partial t} R = R\omega_{\vee} = \left(R\omega\right)_{\vee} R \tag{3.16}$$

where ω is the angular velocity. By modifying the last term of the equation and based on [70][83], we define two types of non-linear filters.

Direct complementary filter with bias correction

$$\frac{\partial}{\partial t}\, {}^{A}_{E}\hat{R}^{D} = \left({}^{A}_{E}\hat{R}_{s} \left({}_{B}\tilde{\omega} - \hat{b}^{D}_{g} \right) + k_{p}\, {}^{A}_{E}\hat{R}^{D} \sigma \right)_{\vee} {}^{A}_{E}\hat{R}^{D}, \qquad {}^{A}_{E}\hat{R}^{D}\left(0\right) = {}^{A}_{E}\hat{R}_{s0} \tag{3.17}$$

$$\frac{\partial}{\partial t}\hat{b}^{D}_{g} = -k_{I}\sigma, \qquad \hat{b}^{D}_{g}\left(0\right) = 0 \tag{3.18}$$

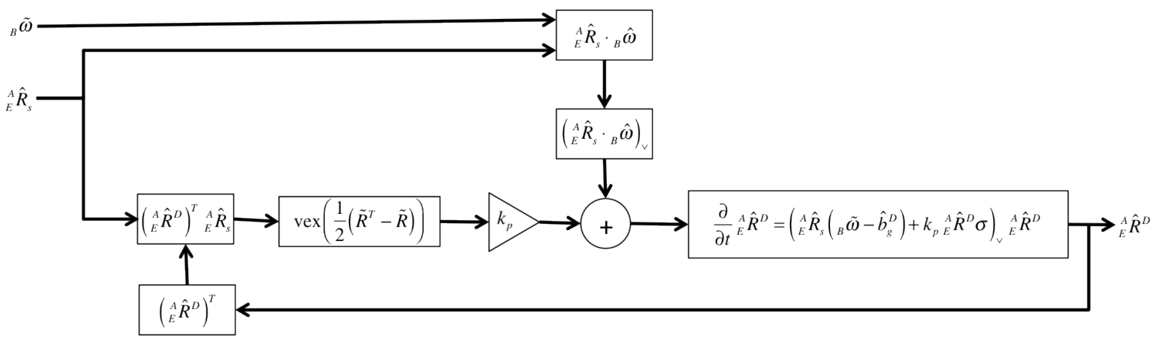
$$\sigma = \operatorname{vex}\left(\frac{1}{2}\left(\tilde{R}^{T} - \tilde{R} \right) \right), \qquad \tilde{R} = \left({}^{A}_{E}\hat{R}^{D} \right)^{T} {}^{A}_{E}\hat{R}_{s} \tag{3.19}$$

Passive complementary filter with bias correction

$$\frac{\partial}{\partial t}\, {}^{A}_{E}\hat{R}^{P} = \left({}^{A}_{E}\hat{R}^{P} \left({}_{B}\tilde{\omega} - \hat{b}^{P}_{g} \right) + k_{p}\, {}^{A}_{E}\hat{R}^{P} \sigma \right)_{\vee} {}^{A}_{E}\hat{R}^{P}, \qquad {}^{A}_{E}\hat{R}^{P}\left(0\right) = {}^{A}_{E}\hat{R}_{s0} \tag{3.20}$$

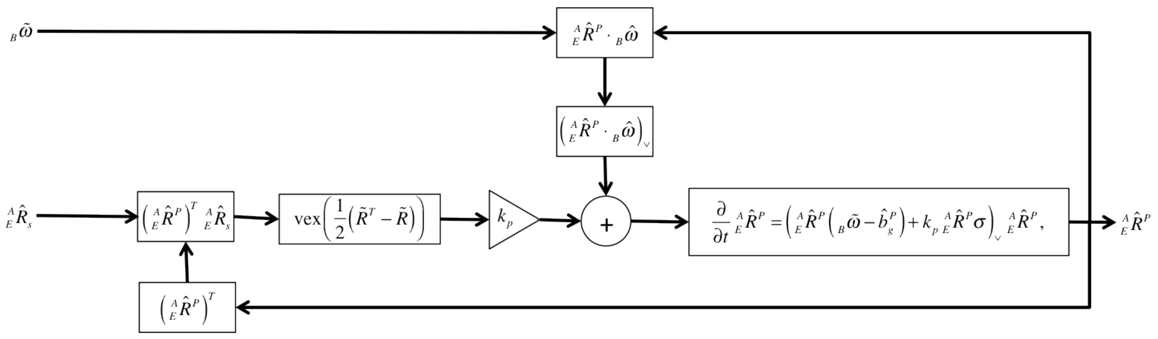
$$\frac{\partial}{\partial t}\hat{b}^{P}_{g} = -k_{I}\sigma, \qquad \hat{b}^{P}_{g}\left(0\right) = 0 \tag{3.21}$$

$$\sigma = \operatorname{vex}\left(\frac{1}{2}\left(\tilde{R}^{T} - \tilde{R} \right) \right), \qquad \tilde{R} = \left({}^{A}_{E}\hat{R}^{P} \right)^{T} {}^{A}_{E}\hat{R}_{s} \tag{3.22}$$



$_B\tilde{\omega}$
$^A_E\hat{R}_s$
$^A_E\hat{R}_s \cdot {}_B\hat{\omega}$
$\left(^A_E\hat{R}_s \cdot {}_B\hat{\omega}\right)_\vee$
$\left(^A_E\hat{R}^D\right)^T {}^A_E\hat{R}_s$
$\mathrm{vex}\left(\frac{1}{2}\left(\tilde{R}^T - \tilde{R}\right)\right)$
k_p
$+$
$\frac{\partial}{\partial t}{}^A_E\hat{R}^D = \left(^A_E\hat{R}_s\left({}_B\tilde{\omega} - \hat{b}^D_g\right) + k_p {}^A_E\hat{R}^D\sigma\right)_\vee {}^A_E\hat{R}^D$
$^A_E\hat{R}^D$
$\left(^A_E\hat{R}^D\right)^T$

Figure 3.3. Block diagram of direct complementary filter



$_B\tilde{\omega}$
$^A_E\hat{R}^P \cdot {}_B\hat{\omega}$
$\left(^A_E\hat{R}^P \cdot {}_B\hat{\omega}\right)_\vee$
$^A_E\hat{R}_s$
$\left(^A_E\hat{R}^P\right)^T {}^A_E\hat{R}_s$
$\mathrm{vex}\left(\frac{1}{2}\left(\tilde{R}^T - \tilde{R}\right)\right)$
k_p
$+$
$\frac{\partial}{\partial t}{}^A_E\hat{R}^P = \left(^A_E\hat{R}^P\left({}_B\tilde{\omega} - \hat{b}^P_g\right) + k_p {}^A_E\hat{R}^P\sigma\right)_\vee {}^A_E\hat{R}^P,$
$^A_E\hat{R}^P$
$\left(^A_E\hat{R}^P\right)^T$

Figure 3.4. Block diagram of passive complementary filter

where $^A_E\hat{R}^D$ and $^A_E\hat{R}^P$ are direct and passive estimates of the orientation from the estimator frame to the Earth frame, $\hat{b}^D_g$ and $\hat{b}^P_g$ are the estimated bias of gyros, k_p and k_I are positive gains. Figure 3.3 and Figure 3.4 show the block diagram of the direct and passive complementary filters respectively.

In both figures, we can realize non-linear complementary filter models as feedback systems, where static estimation $^A_E\hat{R}_s$ and angular velocity $_B\tilde{\omega}$ serve as inputs to the system, which is used to find the initial estimation using rotational kinematic in (3.14). This initial estimate is then transposed and post multiplied by ${}^{A}_{E}\hat{R}_{s}$ to form the error rotation $\tilde{R} = {}^{A}_{E}\hat{R}_{s}\left({}^{A}_{E}\hat{R}\right)^{T}$, i.e., the transpose in $\mathrm{SO}(3)$ is equivalent to subtraction in linear domain, and so $\tilde{R}$ defines the error between the initial and static orientation estimates. This error is then multiplied by a gain k_{p} and adds to the initial estimate, which forms the final estimate of orientation.

In [70] it was shown that the estimates of the orientations ${}^{A}_{E}\hat{R}^{D}$ and ${}^{A}_{E}\hat{R}^{P}$, as well as gyro bias estimations , $\hat{b}_{g}^{D}$ and $\hat{b}_{g}^{P}$ will converge to the true values ${}^{A}_{B}R$ and b_{g} respectively, and for almost all initial conditions the trajectory $\left({}^{A}_{E}\hat{R}^{D}(t), \hat{b}_{g}^{D}(t)\right)$ and $\left({}^{A}_{E}\hat{R}^{P}(t), \hat{b}_{g}^{P}(t)\right)$ are locally exponentially stable to the trajectory $\left({}^{A}_{B}R(t), b_{g}(t)\right)$. Here, we use both methods to estimate the orientations of the sensors.

3.2.1.3 Upper Body Motion Decomposition

After finding the orientations of sensors using complementary filters, we then use the biomechanical models for the human limbs to reconstruct human motions. As mentioned in 1.3.2, we use a hierarchy of human joints as shown in Figure 3.5 to represent motions as a series of transitions of human limbs. In this model, let the joint m denotes a parent joint with its location in the Earth frame ${}_{A}P_{M} \in \mathbb{R}^{3}$. This parent is connected with its child $m+1$ by a bone of length l_{m} with estimated orientation ${}^{A}_{E}\hat{R}_{l_{m}}$. This estimated orientation ${}^{A}_{E}\hat{R}_{l_{m}}$ is the same as that of the sensor attached to the limb.

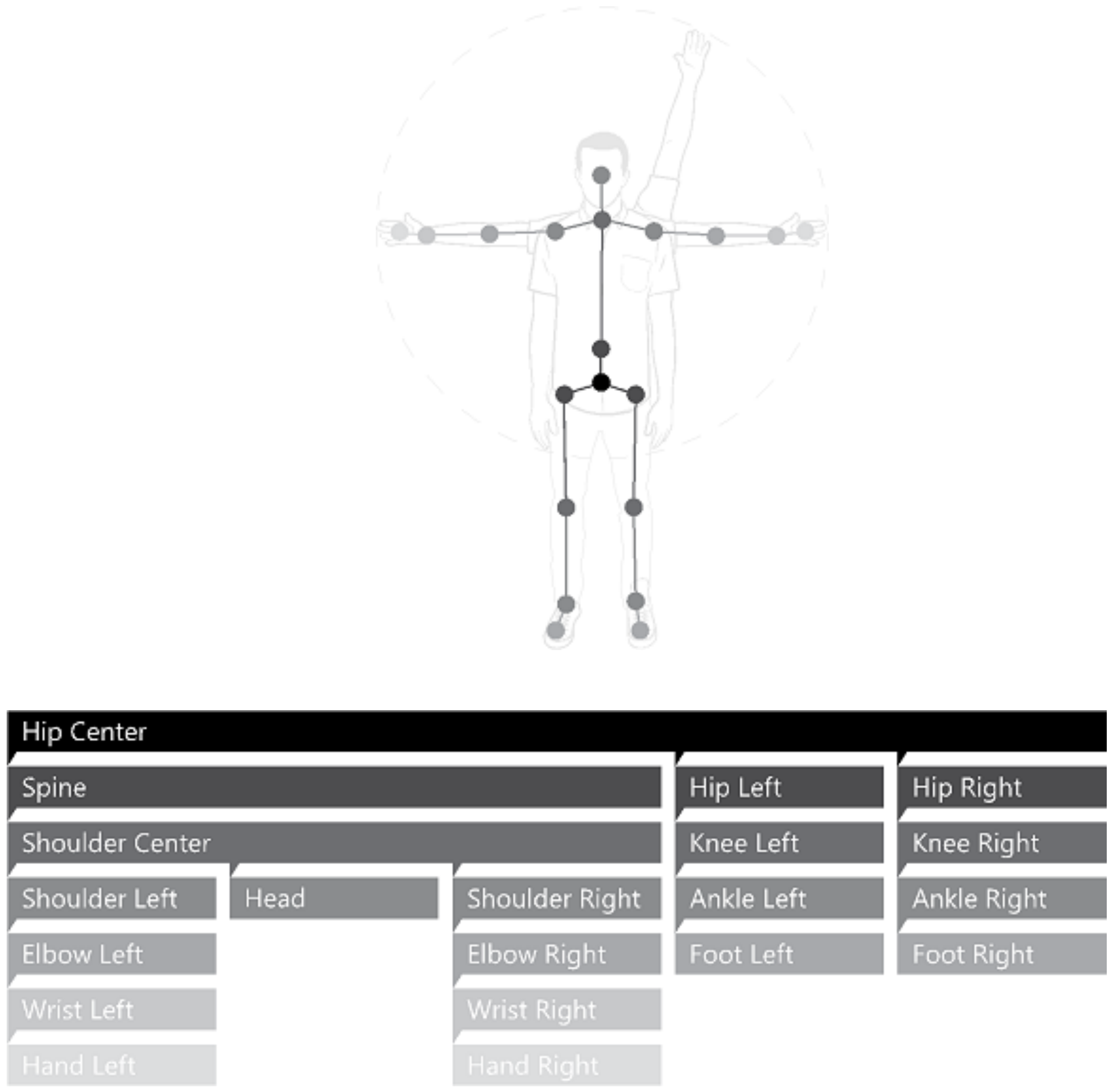


Figure 3.5. Joint hierarchy

Here, we put sensors in the middle of the limbs, and align the y-axis of the sensors with the bone, where the positive direction of the y-axis points outward from the human body. Since the y-axis of the sensor is aligned with the bone, we can define a relative vector ${}_E V = {}_E\begin{bmatrix} 0 & l_m & 0 \end{bmatrix}^T \in \mathbb{R}^3$ in the estimator frame. This vector represents the direction of the

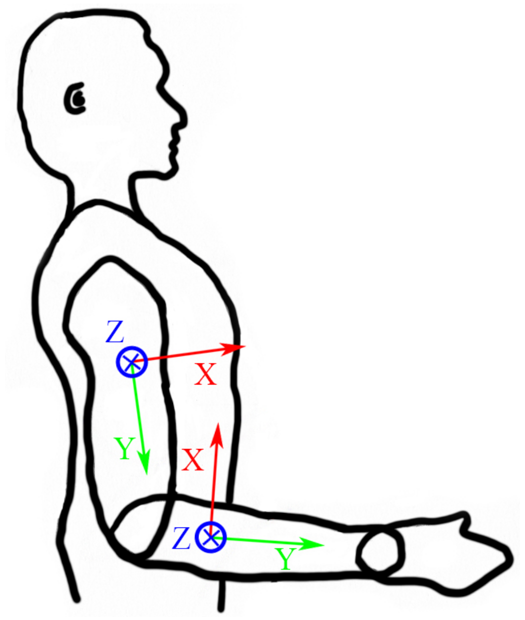


Figure 3.6. Upper limb decompositions and sensor placements

child joint $m+1$ seen by its parent joint m. We can then express the location of the child joint in the estimator frame as

$$_{E}P_{m+1} = {_{E}P_{m}} + {^{A}_{E}\hat{R}_{l_m}} \cdot {_{E}V} \tag{3.23}$$

This formula describes how we can find positions of child joints given their parents.

The hierarchy structure in the previous section describes how we model the motion of every human joint at any given moment. We use this model to estimate motions of human upper limbs. In this model, we assume that the upper limb motions can be decomposed into upper arm and forearm movements. We put sensors in the middle of these two limbs, and align the y-axis of the sensors with the bone, where the plus direction of the y-axis points outward from the human body. Figure 3.6 shows positions and orientations of sensor placements. We do not put sensors on the fist and consider the fist as a part that extends from the forearm. Also, we consider the shoulder to be a fixed joint in the space. Therefore, the whole model of the upper limb looks like a double pendulum.

We set the origin at the position of the shoulder joint. Let ${}^{A}_{E}\hat{R}_{U}$ and ${}^{A}_{E}\hat{R}_{F}$ represent the estimated orientations of the upper arm and the forearm respectively. Also, let l_u represent the length of the upper arm, and l_f be the length of the forearm and the fist. Then from equation (3.23) we can find the estimated positions of the elbow ${}_{A}\hat{P}_{W}$ and the fist ${}_{A}\hat{P}_{F}$ in the Earth frame by

$$\begin{aligned} {}_{A}\hat{P}_{W} &= {}^{A}_{E}\hat{R}_{U}\begin{bmatrix} 0 & l_u & 0 \end{bmatrix}^T \\ {}_{A}\hat{P}_{F} &= {}_{A}\hat{P}_{W} + {}^{A}_{E}\hat{R}_{F}\begin{bmatrix} 0 & l_f & 0 \end{bmatrix}^T \end{aligned} \tag{3.24}$$

By calculating ${}_{A}P_{W}$ and ${}_{A}P_{F}$, we can then estimate upper limb trajectories in NED coordinates.

3.2.1.4 Parameter Optimization

Since human limbs deform when twisting, they cannot be considered as ideal rigid bodies. Therefore, the above double pendulum model needs to be fixed. We remodel the problem into a supervised training procedure.

At first, we asked subjects to perform some designated motions, and recorded the ground truth. The training motions were designed to be easily followed and were repeated for several times. Later on, we compared the estimated results with the ground truth, and calculated estimation errors. We tuned the parameters $\left(k_p, k_I\right)$ in equations (3.17) and (3.20) and then recorded estimation errors. Finally, we found the optimal set of $\left(k_p^*, k_I^*\right)$ such that the estimation error is minimized.

In this study, subjects were asked to slowly draw a square of length l on a wall, and they stopped for a while at each vertex. We then rebuilt the training motions using the algorithms of 3.2.1.2 and 3.2.1.3 with different parameters $\left(k_p, k_I\right)$. We compared the reconstructed length to l and found the estimation error. The optimal set $\left(k_p^*, k_I^*\right)$ given the minimum error would be used in the testing experiments. In summary, we have

$$\left(k_p^*, k_I^*\right) = \underset{\left(k_p, k_I\right) \in \left(\mathbb{R}, \mathbb{R}\right)}{\arg\min} \sum_{k=1}^{N-1} \left| \left\| {}_A P_{F,k+1} - {}_A P_{F,k} \right\| - l \right| \tag{3.25}$$

where ${}_A P_{F,j} \in \mathbb{R}^3$ is the position of the fist for the j^{th} vertex of the square.

It turns out that equation (3.25) is a nonconvex problem, and therefore we exhaustively searched a certain range to find the optimal set. Figure 3.7 shows an example of the error versus different $\left(k_p, k_I\right)$ of the passive complementary filter for one subject. The minimum error of 2.72% happens when $\left(k_p^*, k_I^*\right) = \left(1.3, 0.8\right)$. For direct complementary filters we have similar results and they are also nonconvex problems. Once we found the optimal parameters, we then applied them to the testing set, and that completes the training process.

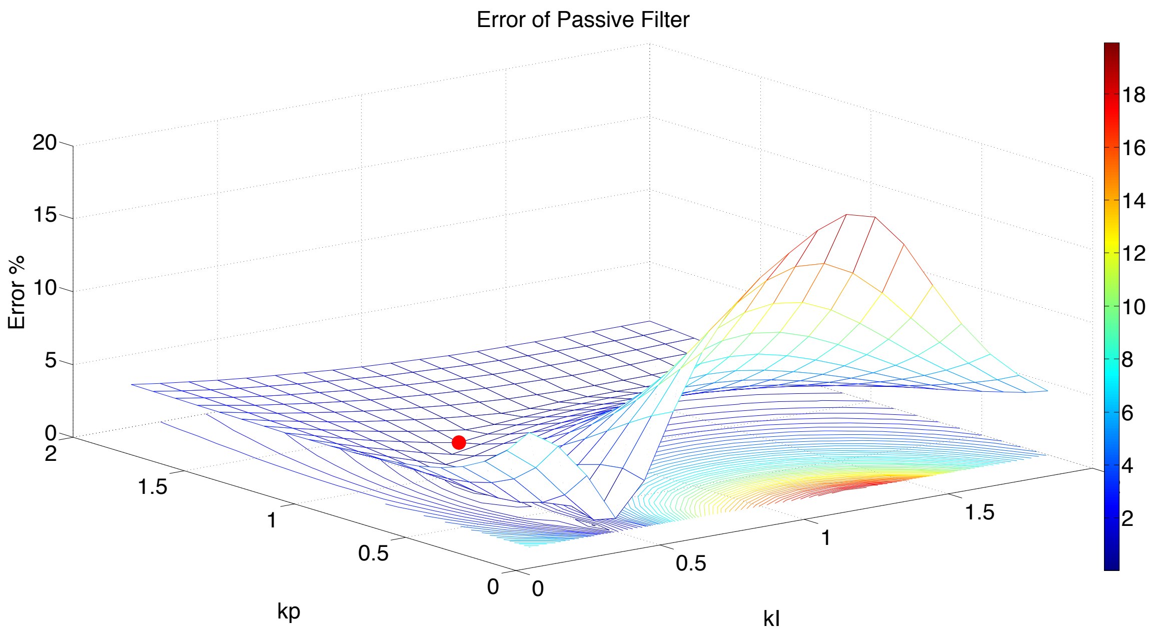


Figure 3.7. Training error using Passive complementary filters

3.2.2 Motion Reconstruction Strategies

In this section we describe how we can reconstruct human motion trajectories with different devices and algorithms described in 3.2.1.2, 3.2.1.3 and 3.2.1.4. In this study, we provide three methods to track human motions: Integration method, IMU method, and IMU with Kinect method.

3.2.2.1 Integration Model

The simplest way to reconstruct motions is through integration of gyro measurements, and by combining with human biomechanical model we can track trajectories in the space as described in Figure 3.8. This method, as will be shown in the experiment results, suffers from severe drift due to numerical integration. Recall from the noisy model of gyro signals (3.9) and the classical rotation kinematic (3.14) we can write the estimated orientations using the integration method ${}^{A}_{E}\hat{R}_{\text{Int}}$ as

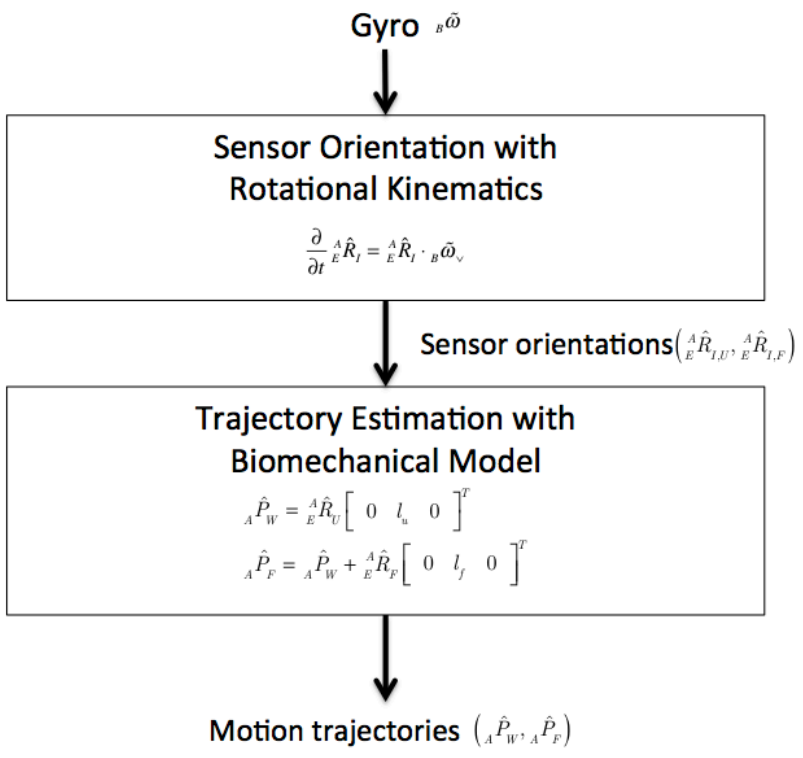

Gyro ${}_B\tilde{\omega}$
Sensor Orientation with Rotational Kinematics
$\frac{\partial}{\partial t}{}_E^A\hat{R}_I = {}_E^A\hat{R}_I \cdot {}_B\tilde{\omega}_\vee$
Sensor orientations$({}_E^A\hat{R}_{I,U}, {}_E^A\hat{R}_{I,F})$
Trajectory Estimation with Biomechanical Model
${}_A\hat{P}_W = {}_E^A\hat{R}_U\begin{bmatrix} 0 & l_u & 0 \end{bmatrix}^T$
${}_A\hat{P}_F = {}_A\hat{P}_W + {}_E^A\hat{R}_F\begin{bmatrix} 0 & l_f & 0 \end{bmatrix}^T$
Motion trajectories $({}_A\hat{P}_W, {}_A\hat{P}_F)$

Figure 3.8. Integration method

$$\frac{\partial}{\partial t}{}_E^V\hat{R}_{\text{Int}} = {}_E^V\hat{R}_{\text{Int}} \cdot {}_B\hat{\omega}_\vee = {}_E^V\hat{R}_{\text{Int}} \cdot \left(M_{g\,B}\boldsymbol{\omega} + b_g + n_g\right)_\vee, \qquad {}_E^V\hat{R}_{\text{Int}}(0) = {}_E^V\hat{R}_{\text{Int0}}$$

while the true value ${}_B^V R_{\text{Int}}$ estimated form the true gyro data without gyro bias b_g and zero-mean AWGN n_g is

$$\frac{\partial}{\partial t}{}_B^V R_{\text{Int}} = {}_B^V R_{\text{Int}} \cdot {}_B\boldsymbol{\omega}_\vee, \qquad {}_B^V R_{\text{Int}}(0) = {}_B^V R_{\text{Int0}} \tag{3.26}$$

Notice that here we create another virtual frame $\{V\}$, and ${}_E^A\hat{R}_{\text{Int}}$ tries to transpose vectors expressed in the body frame $\{B\}$ to this virtual frame $\{V\}$. This virtual frame is created since for the integration model using only gyro signals, we have no way to know where the true positions of the joints are either in the Earth or Kinect frames. Therefore, this virtual frame is created to aid the model to locate the joint positions. The x, y, and z axes of this frame are arbitrarily defined. We define the first orientation of ${}_{E}^{V}\hat{R}_{\mathrm{Int}}$ as the identity matrix, which means

After finding the sensor orientations of the forearm and upper arm $\left({}_{E}^{V}\hat{R}_{\mathrm{Int},U}, {}_{E}^{V}\hat{R}_{\mathrm{Int},F}\right)$, from the estimator frame to the Earth frame, we then use equation (3.24) to find the positions of the elbow and the fist $\left({}_{V}\hat{P}_{\mathrm{Int},W}, {}_{V}\hat{P}_{\mathrm{Int},F}\right)$ in the virtual frame, which serve as the final estimations of motion trajectories.

3.2.2.2 IMU Model

In this section, we reconstruct motion trajectories using the accelerometers, gyros and magnetometers of the IMU. We first introduce how to find the static orientation using accelerometers and magnetometers, and then introduce the whole process of acquiring motion trajectories.

3.2.2.2.1 Static Orientations from Accelerometers and Magnetometers

Ideally, we can acquire the static orientation given we have a set of two nonparallel vectors $\left(v_1, v_2\right)$. From (3.13) we know

$$
{}_{E}^{A}\hat{R}_{s} = \underset{R \in \mathrm{SO}(3)}{\arg\min} \left(\lambda_1 \left\| \frac{R \cdot {}_{B}v_1}{{}_{B}v_1} - \frac{{}_{A}v_1}{{}_{A}v_1} \right\|^2 + \lambda_2 \left\| \frac{R \cdot {}_{B}v_2}{{}_{B}v_2} - \frac{{}_{A}v_2}{{}_{A}v_2} \right\|^2 \right)
$$

However, due to the computational complexity of solving this equation usually a suboptimal solution is provided. In this section, we explain how we calculate the static orientation of the IMU given accelerometer and magnetometer measurements.

Recall the definitions of frames of reference in 3.2.1.1:

$\{E\}$: Earth frame of reference

$\{B\}$: Body frame of reference

${}_{A}g$: Normalized ideal gravity in Earth fame when the sensor is not moving

${}_{B}g$: Normalized ideal gravity in body frame when the sensor is not moving

${}_{A}m$: Normalized ideal magnet in Earth frame when the sensor is not moving

${}_{B}m$: Normalized ideal magnet in body frame when the sensor is not moving

We want to know ${}_{B}^{A}R$, which is the orientation from the body to Earth frames, such that

$$
\begin{aligned}
{}_{A}g &= {}_{B}^{A}R \, {}_{B}g \\
{}_{A}m &= {}_{B}^{A}R \, {}_{B}m
\end{aligned}
$$

Since we use the NED (north east down) coordinate system, in the Earth frame we have

$$
\begin{aligned}
{}_{A}g &= {}_{B}^{A}R \, {}_{B}g = e_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \\
{}_{A}m &= {}_{B}^{A}R \, {}_{B}m = e_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T
\end{aligned}
$$

where e_i is the standard basis vector of $\mathbb{R}^3$. Since $e_3 \times e_1 = e_2$, and from the knowledge of the cross product of vectors a, b under matrix transformation M

$$
(Ma)\times(Mb) = \det(M)\, M^{-T}(a \times b)
$$

we have

$$\begin{aligned} e_2 &= {}_A g \times {}_A m \\ &= {}^A_B R\, {}_B g \times {}^A_B R\, {}_B m \\ &= \det\left({}^A_B R\right) {}^A_B R^{-T} \left({}_B g \times {}_B m\right) \\ &= {}^A_B R\left({}_B g \times {}_B m\right) \end{aligned} \tag{3.27}$$

Finally, we have

$$\begin{aligned} I &= \begin{bmatrix} e_1 & e_2 & e_3 \end{bmatrix} \\ &= \begin{bmatrix} {}^A_B R\, {}_B m & {}^A_B R\left({}_B g \times {}_B m\right) & {}^A_B R\, {}_B g \end{bmatrix} \\ &= {}^A_B R \begin{bmatrix} {}_B m & {}_B g \times {}_B m & {}_B g \end{bmatrix} \end{aligned}$$

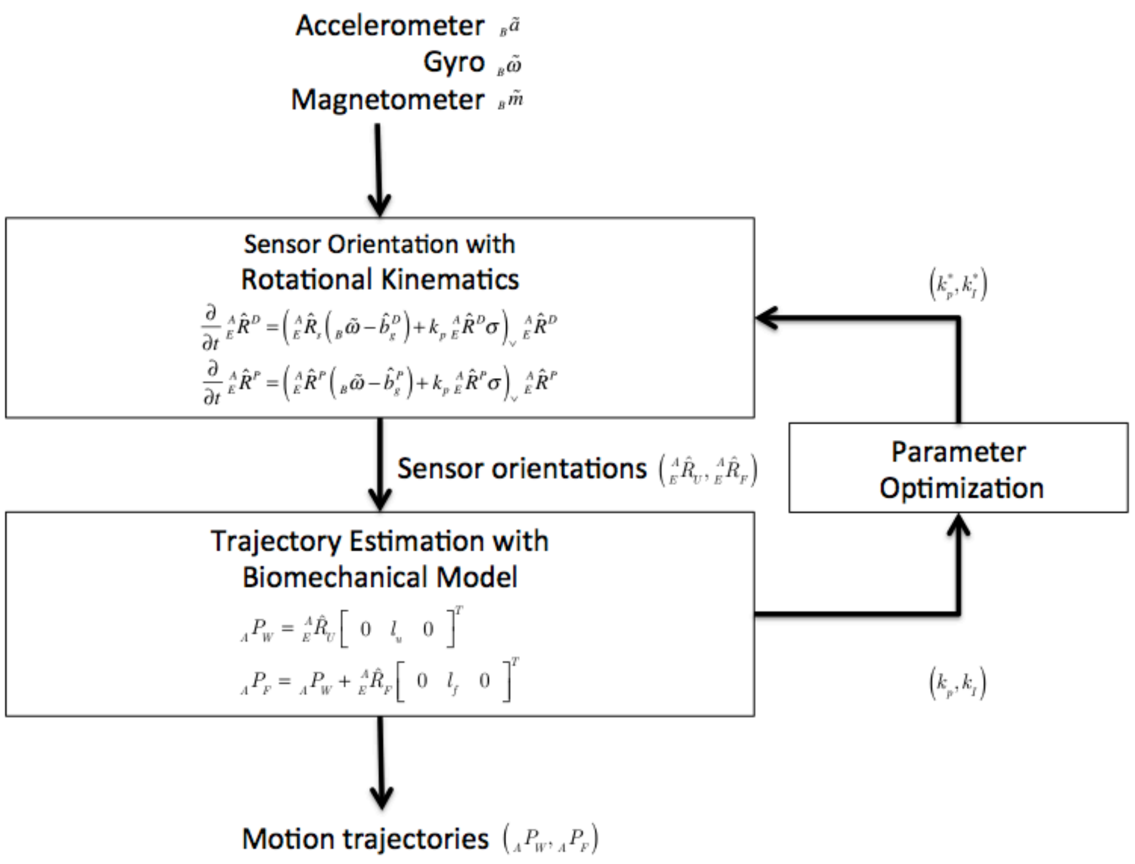
which indicates

$${}^A_B R = \begin{bmatrix} {}_B m & {}_B g \times {}_B m & {}_B g \end{bmatrix}^{-1} \tag{3.28}$$

Since the accelerometer and magnetometer measurements are noisy, we perform another cross product of the second and the third column vector of (3.28) to ensure the estimated rotation matrix is orthogonal. Therefore, we have our final estimate of the static orientation

$${}^A_E \hat{R}_{\mathrm{IMU},s} = \begin{bmatrix} \left(\dfrac{{}_B\tilde{a}}{\left\|{}_B\tilde{a}\right\|} \times \dfrac{{}_B\tilde{m}}{\left\|{}_B\tilde{m}\right\|}\right) \times \dfrac{{}_B\tilde{a}}{\left\|{}_B\tilde{a}\right\|} & \dfrac{{}_B\tilde{a}}{\left\|{}_B\tilde{a}\right\|} \times \dfrac{{}_B\tilde{m}}{\left\|{}_B\tilde{m}\right\|} & \dfrac{{}_B\tilde{a}}{\left\|{}_B\tilde{a}\right\|} \end{bmatrix}^T \tag{3.29}$$

The reason we modify the first vector of ${}^A_E \hat{R}_{\mathrm{IMU},s}$ is that generally the magnetometer measurement is more inaccurate compared to accelerometer readings due to disruption of the Earth's magnetic field due to structures, magnets, and the like.



Accelerometer $_B\tilde{a}$
Gyro $_B\tilde{\omega}$
Magnetometer $_B\tilde{m}$
Sensor Orientation with Rotational Kinematics
$\frac{\partial}{\partial t}\,{}^A_E\hat{R}^D = \left({}^A_E\hat{R}_s\left({}_B\tilde{\omega}-\hat{b}_g^D\right)+k_p\,{}^A_E\hat{R}^D\sigma\right)_\vee {}^A_E\hat{R}^D$
$\frac{\partial}{\partial t}\,{}^A_E\hat{R}^P = \left({}^A_E\hat{R}^P\left({}_B\tilde{\omega}-\hat{b}_g^P\right)+k_p\,{}^A_E\hat{R}^P\sigma\right)_\vee {}^A_E\hat{R}^P$
Sensor orientations $\left({}^A_E\hat{R}_U, {}^A_E\hat{R}_F\right)$
Trajectory Estimation with Biomechanical Model
${}_AP_W = {}^A_E\hat{R}_U\begin{bmatrix}0 & l_u & 0\end{bmatrix}^T$
${}_AP_F = {}_AP_W + {}^A_E\hat{R}_F\begin{bmatrix}0 & l_f & 0\end{bmatrix}^T$
Parameter Optimization
$\left(k_p^*, k_I^*\right)$
$\left(k_p, k_I\right)$
Motion trajectories $\left({}_AP_W, {}_AP_F\right)$

Figure 3.9. Motion reconstruction using IMU

This static estimate is accurate if the object moves slowly and the measurement error is small, that is, in low frequency conditions we have ${}^A_E\hat{R}_{\mathrm{IMU},s} \approx {}^A_B R$.

3.2.2.2.2 Motion Tracking Scheme using IMU Model

After obtaining the static estimate of sensor orientations ${}^A_E\hat{R}_{\mathrm{IMU},s}$, we then use the passive and direct non-linear complementary filters in 3.2.1.2 to find out the estimated rotation of the upper arm and forearm from the estimated frame to the Earth frame for the direct $\left({}^A_E\hat{R}^D_{\mathrm{IMU},U}, {}^A_E\hat{R}^D_{\mathrm{IMU},F}\right)$ and passive $\left({}^A_E\hat{R}^P_{\mathrm{IMU},U}, {}^A_E\hat{R}^P_{\mathrm{IMU},F}\right)$ complementary filters respectively. We then use the biomechanical model in 3.2.1.3, and parameter optimization in 3.2.1.4 to find the final elbow and fist position in the Earth frame $\left({}_{A}\hat{P}_{\text{IMU},W}, {}_{A}\hat{P}_{\text{IMU},F} \right)$ respectively. The whole model can is described in Figure 3.9.

3.2.2.3 Kinect and IMU Model

Since measurements form magnetometers get severely distorted when indoors, the idea of using such a model of heterogeneous sensors is to try to replace magnetometer readings by Kinect data, when available. In this section, we reconstruct motion trajectories using accelerometers, gyros and the Kinect. We first introduce how to find the static orientation using accelerometers and the Kinect, and then introduce the whole process of acquiring motion trajectories.

3.2.2.3.1 Static Orientations from Accelerometers and Kinect

In finding the rotation between the body frame of reference $\{B\}$ and the Kinect frame of reference $\{K\}$, from the definition of 3.2.1.1 recall:

$\{K\}$: Kinect frame of reference

$\{B\}$: Body frame of reference

${}_{K}g$: Normalized ideal gravity in Kinect frame when the sensor is not moving

${}_{B}g$: Normalized ideal gravity in body frame when the sensor is not moving

${}_{K}b$: Normalized bone (limb) pointing outward in Kinect frame

${}_{B}b$: Normalized bone (limb) pointing outward in body frame

Recall the Kinect axes definition in Figure 3.1 with $\{B\}$ frame and the bone vector b being marked. In this frame of reference we have ${}_{K}g = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^{T}$ pointing to the negative y

$$_{K}g = {}^{K}_{B}R\,_{B}g$$
$$_{K}b = {}^{K}_{B}R\,_{B}b$$

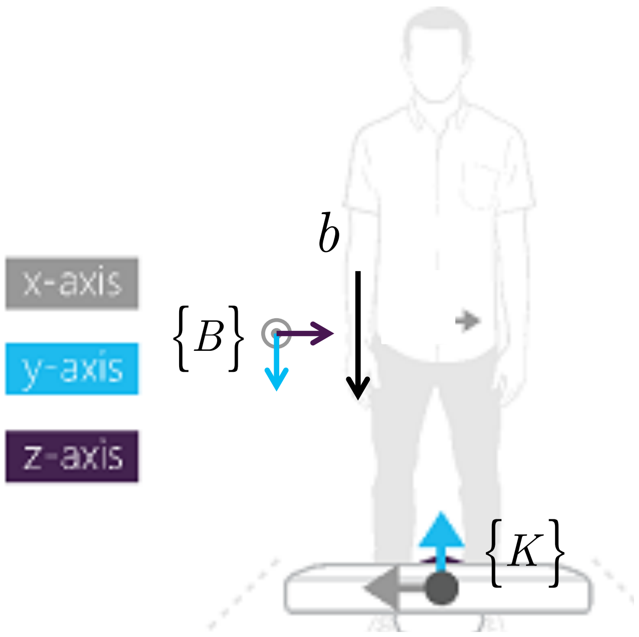


Figure 3.1. Kinect axes definition

direction of Kinect frame (notice difference between $_{A}g = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{T}$ and $_{K}g = \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^{T}$); and since the bone is aligned with the plus y-axis of body frame, we have $_{B}b = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^{T}$ in the body frame.

We want to find ${}^{K}_{B}R$, which is the relationship between frames of reference $\{K\}$ and $\{B\}$, such that

$$_{K}g = {}^{K}_{B}R\,_{B}g$$
$$_{K}b = {}^{K}_{B}R\,_{B}b$$

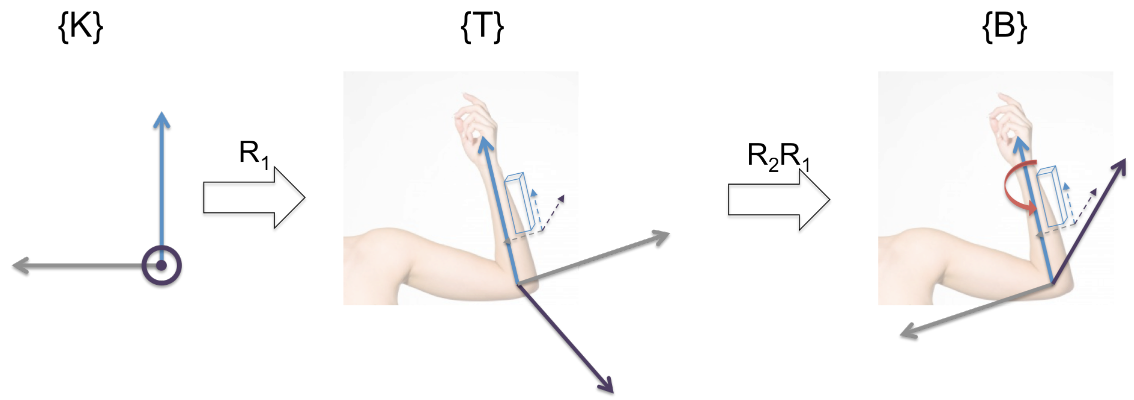


Figure 3.10. Relationship between frames of reference

In order to solve this problem, we introduce a virtual transition frame of reference, named $\{T\}$. This transition frame has the y-axis aligned with the y-axis of $\{B\}$ but the x and z-axis may not align with those of the body frame of reference. $\{T\}$ is transformed from $\{K\}$ with a rotation ${}^{T}_{K}R_1$. As shown in Figure 3.10, the frame $\{T\}$ can be transformed to the frame $\{B\}$ via another rotation ${}^{B}_{T}R_2$. Therefore, for any vector $a \in \mathbb{R}^3$, we have

$$ {}_{B}a = {}^{B}_{T}R_2 \, {}^{T}_{K}R_1 $$

Thus

$$ {}^{K}_{B}R = \left({}^{B}_{T}R_2 \, {}^{T}_{K}R_1 \right)^T . $$

We solve this problem in 2 stages

- Stage 1 rotates $\{K\}$ frame to $\{T\}$ frame through the rotation matrix ${}^{T}_{K}R_1$, thus transforms ${}_{K}b$ into ${}_{T}b$, and since the y-axis of $\{T\}$ and $\{B\}$ frames are aligned, we

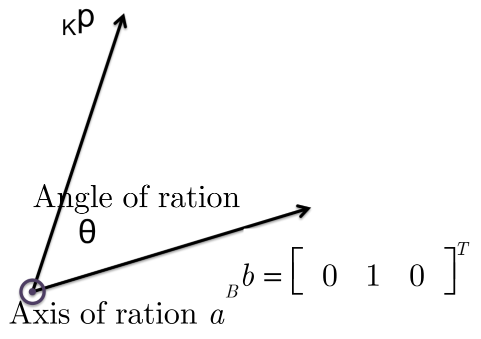


Figure 3.11. Stage 1 of rotating $_B$b to $_K$b

have $_B b = {}_T b$. Therefore we have $_B b = {}_T b = {}^T_K R_1 \, {}_K b$, which means ${}^T_K R_1$ is a matrix that rotates $_B b$ to $_K b$.

- Stage 2 rotates the along the y-axis of $\{T\}$ to $\{B\}$ using rotation ${}^B_T R_2$, so we have

$$ {}^B_T R_2 \, {}^T_K R_1 \, {}_K g = {}_B g \tag{3.30} $$

And since the y-axis of $\{T\}$ is aligned with that of $\{B\}$, we also have

$$ {}^B_T R_2 \, {}^T_K R_1 \, {}_K b = {}_B b = {}_T b = {}^T_K R_1 \, {}_K b \tag{3.31} $$

From (3.30) and (3.31), and knowing that g and p are not parallel to each other, we conclude that the transpose of the multiplication of ${}^B_T R_2$ and ${}^T_K R_1$ is the desired orientation of the limb, which is

$$ {}^K_B R = \left({}^B_T R_2 \, {}^T_K R_1 \right)^T \tag{3.32} $$

Below is the detailed explanation of what is being done.

Stage 1

In stage 1 ${}^{T}_{K}R_1$ rotates ${}_{B}b$ to ${}_{K}b$. This can be shown in the Figure 3.11.

The axis of rotation a is found from cross product of ${}_{K}b$ and ${}_{B}b$, and the angle of rotation is found using dot product of these two vectors as follows

$$a = {}_{B}b \times {}_{K}b$$
$$\theta = \cos^{-1}\left(\frac{{}_{B}b \cdot {}_{K}b}{\left\|{}_{B}b\right\|\left\|{}_{K}b\right\|}\right)$$

We transform this axis-angle representation into rotation matrix ${}^{T}_{K}R_1$ using

$${}^{T}_{K}R_1 = \begin{bmatrix} \cos\theta + a_x^2(1-\cos\theta) & a_x a_y(1-\cos\theta) - a_z\sin\theta & a_x a_z(1-\cos\theta) - a_y\sin\theta \\ a_y a_x(1-\cos\theta) - a_z\sin\theta & \cos\theta + a_y^2(1-\cos\theta) & a_y a_z(1-\cos\theta) - a_x\sin\theta \\ a_z a_x(1-\cos\theta) - a_y\sin\theta & a_z a_y(1-\cos\theta) - a_x\sin\theta & \cos\theta + a_z^2(1-\cos\theta) \end{bmatrix} \quad (3.33)$$

Stage 2

In stage 2 we rotate the y-axis of $\{T\}$ such that the x and z axes of the rotated frame are parallel to those of $\{B\}$ using matrix ${}^{B}_{T}R_2$. Since ${}^{B}_{T}R_2$ is a rotation along the y-axis, we can write ${}^{B}_{T}R_2$ as

$${}^{B}_{T}R_2 = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \quad (3.34)$$

where ϕ is some rotating angle. We find the angle ϕ^* such that the norm the difference between ${}_K g$ and the rotated ${}_B g$ is minimized, which means we have

$$\phi^* = \underset{\phi \in [-\pi \quad \pi)}{\arg\min} \left\| {}^B_T R_2 \, {}^T_K R_1 \, {}_B g - {}_K g \right\|$$

We solve this problem using an exhaustive search. The final static estimate of the orientation is then shown by (3.32).

Given we have noisy accelerometer and Kinect signals, in estimating the static rotation from the body $\{B\}$ to Kinect $\{K\}$ frames, we assume the following

$$\begin{aligned}
{}_K g &= \begin{bmatrix} 0 & -1 & 0 \end{bmatrix}^T \\
{}_B g &= \frac{{}_B \hat{a}}{\left\| {}_B \hat{a} \right\|} \\
{}_K b &= \frac{{}_K \hat{p}_{i+1} - {}_K \hat{p}_i}{\left\| {}_K \hat{p}_{i+1} - {}_K \hat{p}_i \right\|} \\
{}_B b &= \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T
\end{aligned}$$

This assumption indicates that we use the accelerometer reading ${}_B \hat{a}$ as the indication of gravity, and the relative direction of the i^{th} joint to the $i+1^{\text{th}}$ point, which is the difference between positions reported by the Kinect ${}_K \hat{p}_{i+1} - {}_K \hat{p}_i$, as the direction of the limb. By following the method described above, by equation (3.32) we have the final static estimation using the accelerometer and Kinect as

$$ {}^K_E \hat{R}_{\text{KINECT},s} = \left({}^E_T \hat{R}_2 \, {}^T_K \hat{R}_1 \right)^T \tag{3.35}$$

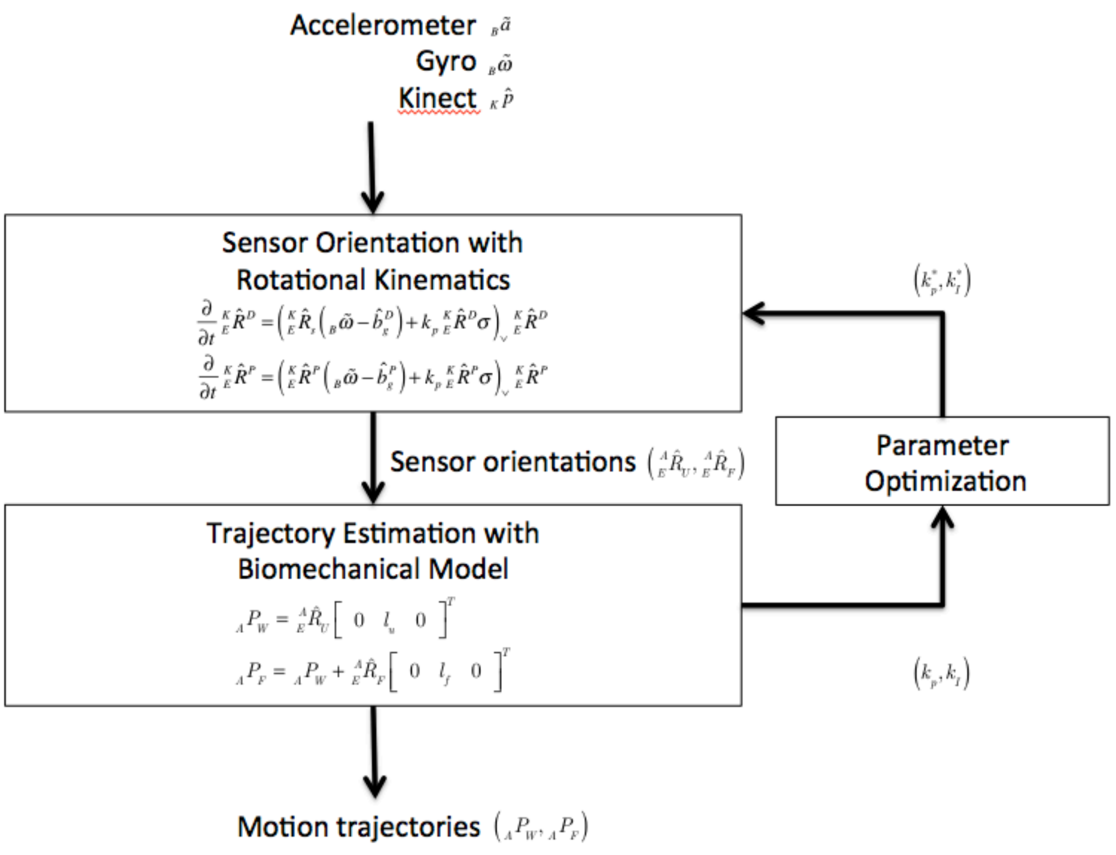

Accelerometer ${}_B\tilde{a}$
Gyro ${}_B\tilde{\omega}$
Kinect ${}_K\hat{p}$
Sensor Orientation with Rotational Kinematics
$\frac{\partial}{\partial t}{}^K_E\hat{R}^D = \left({}^K_E\hat{R}_s\left({}_B\tilde{\omega} - \hat{b}^D_g\right) + k_p\,{}^K_E\hat{R}^D\sigma\right)_\vee {}^K_E\hat{R}^D$
$\frac{\partial}{\partial t}{}^K_E\hat{R}^P = \left({}^K_E\hat{R}^P\left({}_B\tilde{\omega} - \hat{b}^P_g\right) + k_p\,{}^K_E\hat{R}^P\sigma\right)_\vee {}^K_E\hat{R}^P$
(k_p^*, k_I^*)
Sensor orientations $\left({}^A_E\hat{R}_U, {}^A_E\hat{R}_F\right)$
Parameter Optimization
Trajectory Estimation with Biomechanical Model
${}_AP_W = {}^A_E\hat{R}_U\begin{bmatrix} 0 & l_u & 0 \end{bmatrix}^T$
${}_AP_F = {}_AP_W + {}^A_E\hat{R}_F\begin{bmatrix} 0 & l_f & 0 \end{bmatrix}^T$
(k_p, k_I)
Motion trajectories $\left({}_AP_W, {}_AP_F\right)$

Figure 3.12. Motion reconstruction using Kinect and IMU

where ${}^T_K\hat{R}_1$ and ${}^E_T\hat{R}_2$ are acquired from (3.33) and(3.34) respectively.

3.2.2.3.2 Motion Tracking Scheme using Kinect and IMU Model

After obtaining the static estimate of sensor orientations ${}^K_E\hat{R}_{\text{KINECT},s}$, we then use the passive and direct non-linear complementary filters in 3.2.1.2 to find the estimated orientation of the upper arm and forearm from the estimated frame to the Kinect frame for the direct complementary filter $\left({}^A_E\hat{R}^D_{\text{KINECT},U}, {}^A_E\hat{R}^D_{\text{KINECT},F}\right)$ and for the passive complementary filter $\left({}^K_E\hat{R}^P_{\text{KINECT},U}, {}^K_E\hat{R}^P_{\text{KINECT},F}\right)$ respectively. We then use the biomechanical model in 3.2.1.3, and

Table 3.2. Comparison of models

Models	Devices	Frames estimated	Pros	Cons
Integration	Gyro	Virtual frame $\{V\}$	Only one type of sensor needed Portable sensors	Integration drift
IMU	Accelerometer Magnetometer Gyro	Earth frame $\{A\}$	Consistent direction reference Portable sensors	Magnetometer interference when indoor
Kinect and IMU	Acceletometer Gyro Kinect	Kinect frame $\{K\}$	Accurate tracking indoors	Inconsistent direction reference Background constraint Pose constraint Power outlet required

parameter optimization in 3.2.1.4 to find the final elbow and fist position in the Earth frame $\left({}_{A}\hat{P}_{\text{KINECT},W}, {}_{A}\hat{P}_{\text{KINECT},F}\right)$ respectively. The whole model is described in Figure 3.12

3.2.2.4 Comparison of Models

In this section we compare the 3 different methods of 3.2.2.1, 3.2.2.2, and 3.2.2.3. Table 3.2 gives a comparison of these methods. The integration model uses only portable MEMS gyros; therefore from the power saving perspective this model is the most energy-saving one. However, bias and quantization error from measurements cause severe drifts during integration, so the integration model can only serve as a short-time solution, provided other kinds of sensors are not accessible and other models cannot be used.

For the IMU model, since we estimate sensor orientations in the Earth frame $\left\{A\right\}$, the estimated orientations are always expressed in a consistent reference, and also all MEMS accelerometer, magnetometer and gyros are light-weight and portable, such that the model makes the realization of orientation estimation in any place possible. However, by using magnetometers to measure Earth's magnetic direction, we experience severe interference when indoors, which provides inaccurate static estimations.

The third method fuses Kinect and IMU models and uses the Kinect to replace magnetometers. Therefore we avoid this interference issue, and additionally the Kinect provides accurate joint positions information (given its massive amount of training data). But on the other hand, the estimation of orientation is expressed in the Kinect frame of reference $\left\{K\right\}$, which will change if the Kinect is moved, and we do not have a consistent reference. Also, the Kinect is very background and pose sensitive, which means the background has to be clean and the subject has to face the Kinect before it can recognize joint positions. What is worse, we have to have a power outlet for Kinect and thus this solution it not easily portable.

Starting from the next section we will discuss the relationships between frames of reference so we can link different models.

3.2.3 Coordinate Transformations

Now that we have described the methods for finding the trajectories, we show how to link different models. As seen in Table 3.2, each model transforms measurements expressed in the body frame $\{B\}$ to different frames. Therefore, if we want to fuse different models to improve tracking accuracies, we have to first find out how these frames of reference are related. In this thesis, we use the Earth frame $\{A\}$ as our base frame and try to transform other frames to it. The reason to do this is because magnetometers and accelerometers are always available when collecting data, and they are used to find the positions in the Earth frame. On the other hand, the Kinect usually takes a short period of time to recognize human joints, and when there are obstacles blocking the camera, the Kinect signal would not be available. Therefore, by using the Earth frame we can have a universal basis frame of reference. We will explain how we can find the relationship between the Earth frame and other frames of reference.

3.2.3.1 Kinect Frame to Earth Frame

In the Kinect and IMU model described in 3.2.2.3, we end up having the estimated rotation from the estimation frame to the Kinect frame of the upper arm and the forearm using the direct complementary filter $\left({}_{E}^{A}\hat{R}_{\text{KINECT},U}^{D}, {}_{E}^{A}\hat{R}_{\text{KINECT},F}^{D}\right)$ and passive complementary filter $\left({}_{E}^{K}\hat{R}_{\text{KINECT},U}^{P}, {}_{E}^{K}\hat{R}_{\text{KINECT},F}^{P}\right)$ respectively. In order to describe these orientations in the Earth frame, we must find a transformation ${}_{K}^{A}\hat{R}$ that rotates the orientation from Kinect to Earth frames.

This estimate transformation ${}_{K}^{A}\hat{R}$ is calculated when both the Kinect and magnetometer signals are available, and we wait for a short period of time until Kinect signals are stable then start calculating the transformation. When Kinect joint information becomes not available we will recalculate the transformation once again after the Kinect signals are regained since the Kinect may have been moved. After calculating each transformation at the beginning of Kinect

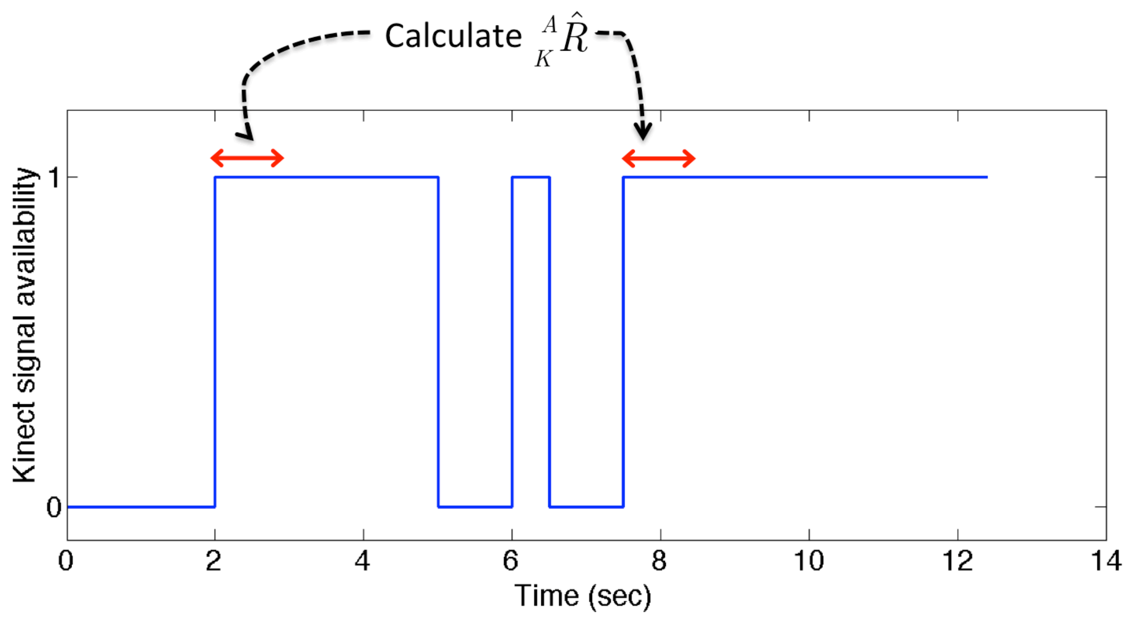


Figure 3.13. Kinect availability

available period, we apply this transformation to the rest of the period, which completes the transformation of orientations estimated in the Kinect frame to Earth frame.

Suppose the availability of the Kinect signal versus time is shown in Figure 3.13. From this figure we can see that after starting the Kinect we have our first Kinect signal available when $t = 2$, then at $t = 5$ Kinect is being blocked and thus not available. At $t = 6$ we regain the Kinect signal for only 0.5 second, then at $t = 7.5$ we have the third period of available Kinect signal. In this study, we set a calculation threshold $cal\_th = p$ second, indicating that if the time period of available Kinect signal is shorter than this time, we would not calculate ${}^{A}_{K}\hat{R}$ since during this short period of time the Kinect just starts to recognize joints and is not stable. If the period of the available Kinect signal is longer than this threshold, we calculate ${}^{A}_{K}\hat{R}$ within this period, and then average the estimate rotation ${}^{A}_{K}\hat{R}$. Thus, for this example, where the calculation threshold is set to $cal\_th = 1$, we calculate ${}_{K}^{A}\hat{R}$ twice. The first instance is for $t = 2$ to $t = 3$, the other is for $t = 7.5$ to $t = 8.5$.

We now explain how this ${}_{K}^{A}\hat{R}$ is actually calculated. Suppose we have the first calculation-start time $t = t_{s1}$, the first calculation-end time $t = t_{ew1}$, and the second calculation-start time $t = t_{s2}$. We want to calculate the average transformation for the first period from the Kinect frame $\{K\}$ to the Earth frame $\{A\}$, which we call ${}_{K}^{A}\hat{R}_{1}$. The first step of this calculation is to find the static orientation estimate of the sensor ${}_{K}^{A}\hat{R}_{\text{KINECT},s}$ from the estimate frame $\{E\}$ to the Kinect frame $\{K\}$ during $t = \left(t_{s1}, t_{e1}\right)$ using equation (3.35),

$$ {}_{E}^{K}\hat{R}_{\text{KINECT},s}\left(t\right)_{t=\left(t_{s1},t_{e1}\right)} \tag{3.36} $$

Also, since magnetometer data is available, we use equation (3.29) to find the static orientation estimate of the sensor ${}_{E}^{A}R_{\text{IMU},s}$ from the estimate frame $\{E\}$ to the Earth frame $\{A\}$ during $t = \left(t_{s1}, t_{e1}\right)$

$$ {}_{E}^{A}\hat{R}_{\text{KINECT},s}\left(t\right)_{t=\left(t_{s1},t_{e1}\right)} \tag{3.37} $$

Using (3.36) and (3.37), we can find the relative transformation ${}_{K}^{A}\hat{R}_{s}$,

$$ \begin{aligned} {}_{K}^{A}\hat{R}_{s}\left(t\right)_{t=\left(t_{s1},t_{e1}\right)} &= {}_{E}^{A}\hat{R}_{\text{KINECT},s}\left(t\right)\left({}_{E}^{K}\hat{R}_{\text{KINECT},s}\left(t\right)\right)^{-1} \\ &= {}_{E}^{A}\hat{R}_{\text{KINECT},s}\left(t\right)\left({}_{E}^{K}\hat{R}_{\text{KINECT},s}\left(t\right)\right)^{T} \end{aligned} \tag{3.38} $$

After finding this relative transformation, we will take the average of it and then apply it to the rest of the signal until the next calculation-start time $t = t_{s2}$. In finding the average of the relative transformation, since the rotation matrix is not continuous we convert the relative transformation into quaternions $Q(t) = \begin{bmatrix} q_w(t) & q_x(t) & q_y(t) & q_z(t) \end{bmatrix}$ using

$$
\begin{aligned}
q_w(t) &= \frac{\sqrt{1 + \left[{}^{A}_{K}\hat{R}_s(t)\right]_{11} + \left[{}^{A}_{K}\hat{R}_s(t)\right]_{22} + \left[{}^{A}_{K}\hat{R}_s(t)\right]_{33}}}{2} \\
q_x(t) &= \frac{\left(\left[{}^{A}_{K}\hat{R}_s(t)\right]_{32} - \left[{}^{A}_{K}\hat{R}_s(t)\right]_{23}\right)}{4q_w} \\
q_y(t) &= \frac{\left(\left[{}^{A}_{K}\hat{R}_s(t)\right]_{13} - \left[{}^{A}_{K}\hat{R}_s(t)\right]_{31}\right)}{4q_w} \\
q_z(t) &= \frac{\left(\left[{}^{A}_{K}\hat{R}_s(t)\right]_{21} - \left[{}^{A}_{K}\hat{R}_s(t)\right]_{12}\right)}{4q_w}
\end{aligned}
$$

Then we find the average of the quaternion

$$
\begin{aligned}
\overline{Q} &= \begin{bmatrix} \overline{q}_w & \overline{q}_x & \overline{q}_y & \overline{q}_z \end{bmatrix} \\
&= \int_{t_{s1}}^{t_{e1}} Q(t)\,dt
\end{aligned}
$$

Finally, we transform the quaternion back to matrix form by

$$
E\left({}^{A}_{K}\hat{R}_s\right) = \begin{bmatrix} 1 - 2\overline{q}_y^2 - 2\overline{q}_z^2 & 2\overline{q}_x\overline{q}_y - 2\overline{q}_z\overline{q}_w & 2\overline{q}_x\overline{q}_z + 2\overline{q}_y\overline{q}_w \\ 2\overline{q}_x\overline{q}_y + 2\overline{q}_z\overline{q}_w & 1 - 2\overline{q}_x^2 - 2\overline{q}_z^2 & 2\overline{q}_y\overline{q}_z - 2\overline{q}_x\overline{q}_w \\ 2\overline{q}_x\overline{q}_z - 2\overline{q}_y\overline{q}_w & 2\overline{q}_y\overline{q}_z + 2\overline{q}_x\overline{q}_w & 1 - 2\overline{q}_x^2 - 2\overline{q}_y^2 \end{bmatrix}
$$

This averaged transformation is then applied to the rest of the Kinect signals to find the static orientation of the sensor from estimate frame $\{B\}$ to Earth frame $\{A\}$ until $t = t_{s2}$, which is

$$ {}_{E}^{A}\hat{R}_{\text{KINECT},s}\left(t\right)_{t=\left(t_{s1},t_{s2}\right)} = E\left({}_{K}^{A}\hat{R}_{s} \right) {}_{E}^{K}\hat{R}_{\text{KINECT},s}\left(t\right)_{t=\left(t_{s1},t_{s2}\right)} \tag{3.39} $$

After we find this static orientation estimate, we replace it with the original static estimate from the estimate frame to the Kinect frame ${}_{E}^{K}\hat{R}_{\text{KINECT},s}$, then by following procedures described in 3.2.2.3.2 we complete the Kinect and IMU motion tracking model.

3.2.3.2 Virtual Frame to Earth Frame

In the integration model described in 3.2.2.1, since there is no static orientation estimate of any kind in this model, this model uses the last frame where the accelerometer and magnetometer/Kinect is accessible. Then the integration model calculates the relative rotations to the last orientation where we have such devices indicating either the Earth frame $\{A\}$ or Kinect frame $\{K\}$.

3.2.4 Coordinate weighting

In this section, we describe how we evaluate the trustworthiness of each model. This is necessary when we fuse different models to acquire more accurate tracking results. By knowing how trustworthy certain methods are we can determine which models we prefer when fusing them.

3.2.4.1 Trustworthiness of Kinect and IMU Model

From [23], we know that every joint has a tracking state, where "tracked" means the joint is clearly visible; "inferred" when it is not clearly visible and being inferred by the Kinect; "non-tracked" when it is not tracked. In this study, we use this tracking state as the indication whether to use the Kinect and IMU model described in 3.2.2.3. In our study, since the sampling rates for heterogeneous sensors need not to be the same, we re-interpolate not only the collected Kinect data, but also the trustworthiness. After we re-interpolate the trustworthiness of the Kinect signal, we then define a threshold. The joint position is inferred when its trustworthiness is below this threshold, and tracked if above the threshold. Figure 3.14 shows an example of the X, Y, and Z data of the collected Kinect signals for the right wrist, while the solid lines represent the tracked positions, and the dotted lines the inferred positions. Figure 3.15 shows the trustworthiness of the Kinect signal for the right wrist, where the decision threshold is set to 0.5. From Figure 3.15 we can see that we have the first tracked Kinect right wrist positions starting from $t = 20.3$ and ending at $t = 26.2$, the second tracked signal from $t = 29.3$ to $t = 30.7$, with a short drop to inferred signals at $t = 30.5$. The third tracked signal starts from $t = 31.3$ to $t = 38.9$.

Notice that this trustworthiness of Kinect position is equivalent to the availability of the Kinect signal mentioned in 3.2.3.1, where we use the period within the calculation threshold after each rising edge to calculate the relative transformation from Kinect frame $\{K\}$ to Earth frame $\{A\}$.

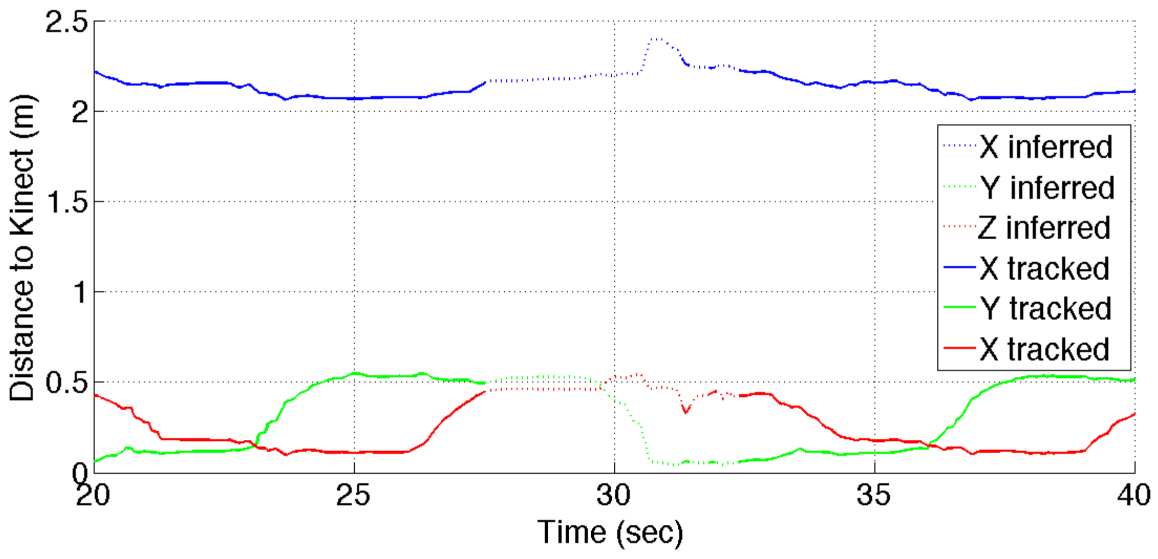


Figure 3.14. Kinect collected signals for the right wrist with tracking status

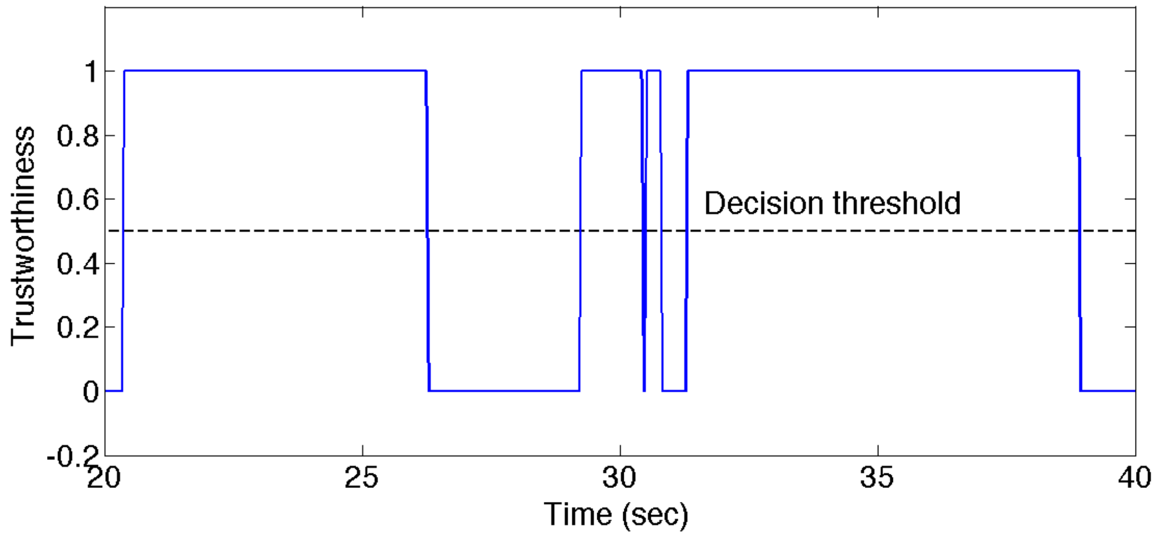


Figure 3.15. Trustworthiness (interpolated) of Kinect signals for the right wrist

3.2.4.2 Trustworthiness of IMU Model

The trustworthiness of the IMU model measures how much we prefer this model and use it to track joint motions.

Ideally, if we have the perfect measurement of accelerometers while the subject is not moving, with no external force applied to the sensor the collected data should reflect the direction of gravity only. Therefore, if you move the accelerometer around this reflection of gravity should be scattered around a sphere centered at the origin and with radius around $9.8\, m/s^2$. However, since the subject applies forces to the sensor when moving, and given the measurement is noisy, the data actually collected will be scattered around a sphere that is not centered at the origin, with various distances different from $9.8\, m/s^2$. Similarly for the magnetometer given we have interference, the measurement will not be scattered around a sphere centered at the origin.

In this study, given accelerometer and magnetometer collected data we first find a best-fit sphere, then with the center and the radius of that sphere we normalize the collected data. After that, by comparing the distance of collected data to the origin we define a curve mapping this distance to trustworthiness, which completes the whole procedure. The detailed explanation is given below.

In the first step we find the best fit sphere given noisy accelerometer and magnetometer measurements. Suppose the we have a total of N measurements, and the i^{th} measurement of the X , Y , Z components of the accelerometer data is expressed as

$$_{B}\tilde{a}_i = \begin{bmatrix} _{B}\tilde{a}_{X_i} & _{B}\tilde{a}_{Y_i} & _{B}\tilde{a}_{Z_i} \end{bmatrix}$$

The center of the best-fit sphere $C^* = \begin{bmatrix} C_X^* & C_Y^* & C_Z^* \end{bmatrix}^T$ is acquired using a closed form for the solution

$$\left\{C_X^*, C_Y^*, C_Z^*\right\} = \underset{\{C_X, C_Y, C_Z\} \in \{\mathbb{R}, \mathbb{R}, \mathbb{R}\}}{\arg\min} \sum_{i=1}^{N} \left({}_B\tilde{a}_{X_i} - C_x\right)^2 + \left({}_B\tilde{a}_{Y_i} - C_Y\right)^2 + \left({}_B\tilde{a}_{Z_i} - C_Z\right)^2 \tag{3.40}$$

According to [84] this can be done by first defining an auxiliary matrix A, a vector b, and a scalar r

$$A = 2 \cdot \begin{bmatrix} E\left[{}_B\tilde{a}_X\left({}_B\tilde{a}_X - E\left[{}_B\tilde{a}_X\right]\right)\right] & E\left[{}_B\tilde{a}_X\left({}_B\tilde{a}_Y - E\left[{}_B\tilde{a}_Y\right]\right)\right] & E\left[{}_B\tilde{a}_X\left({}_B\tilde{a}_Z - E\left[{}_B\tilde{a}_Z\right]\right)\right] \\ E\left[{}_B\tilde{a}_Y\left({}_B\tilde{a}_X - E\left[{}_B\tilde{a}_X\right]\right)\right] & E\left[{}_B\tilde{a}_Y\left({}_B\tilde{a}_Y - E\left[{}_B\tilde{a}_Y\right]\right)\right] & E\left[{}_B\tilde{a}_Y\left({}_B\tilde{a}_Z - E\left[{}_B\tilde{a}_Z\right]\right)\right] \\ E\left[{}_B\tilde{a}_Z\left({}_B\tilde{a}_X - E\left[{}_B\tilde{a}_X\right]\right)\right] & E\left[{}_B\tilde{a}_Z\left({}_B\tilde{a}_Y - E\left[{}_B\tilde{a}_Y\right]\right)\right] & E\left[{}_B\tilde{a}_Z\left({}_B\tilde{a}_Z - E\left[{}_B\tilde{a}_Z\right]\right)\right] \end{bmatrix}$$

$$b = \begin{bmatrix} E\left[R\left({}_B\tilde{a}_X - E\left[{}_B\tilde{a}_X\right]\right)\right] \\ E\left[R\left({}_B\tilde{a}_X - E\left[{}_B\tilde{a}_X\right]\right)\right] \\ E\left[R\left({}_B\tilde{a}_X - E\left[{}_B\tilde{a}_X\right]\right)\right] \end{bmatrix}$$

$$r = {}_B\tilde{a}_X^2 + {}_B\tilde{a}_Y^2 + {}_B\tilde{a}_Z^2$$

where $E[\cdot]$ is the expected value of a random variable. The center of the best-fit sphere is then given by

$$C^* = A^{-1}b \tag{3.41}$$

and the radius of the best-fit sphere R^* is given by

$$R^* = \sqrt{\sum_{i=1}^{N} \left({}_B\hat{a}_{X_i} - C_X^*\right)^2 + \left({}_B\hat{a}_{Y_i} - C_Y^*\right)^2 + \left({}_B\hat{a}_{Z_i} - C_Z^*\right)^2} \tag{3.42}$$

Once we find the center and the radius of the best-fit sphere, we then normalize the noisy measurement by

$$ {}_B\hat{a}' = \begin{bmatrix} \dfrac{{}_B\hat{a}_X - C_X^*}{R^*} & \dfrac{{}_B\hat{a}_Y - C_Y^*}{R^*} & \dfrac{{}_B\hat{a}_Z - C_Z^*}{R^*} \end{bmatrix}^T \tag{3.43} $$

By following the similar procedure we can find the normalized magnetometer data ${}_B m'$. Figure 3.16 and Figure 3.17 show the normalized measurements and the fest-fit sphere for accelerometer and magnetometer data respectively.

After having the normalized measurements, we find the distances of them to the origin. The distances are expressed as the norms of the measurements $d_a = \|{}_B\hat{a}'\|$ and $d_m = \|{}_B\hat{m}'\|$ for accelerometer and magnetometer data respectively. If the distance equals 1, it means that the measurement falls on the best-fit sphere, and we say at this moment we have a good measurement, and we are more in favor of using the IMU model described in 3.2.2.2; if the distance is smaller or larger than 1, then we decrease our trustworthiness following some functions. In our study, we set the trustworthiness of IMU model to 1 to infer that we are very confident of using this model, and 0 to infer that we do not use the model. The function calculating the trustworthiness is defined in the following way

$$ T(d) = \begin{cases} e^{-\lambda_1(d-1)}, & d \geq 1 \\ e^{\lambda_2 t + b} - e^b, & 0 \leq d < 1 \end{cases} \tag{3.44} $$

where λ_1, λ_2, b are some parameters we have previously defined. Figure 3.18 shows this model.

This function transforms the distances of the normalized measurements to the trustworthiness of the IMU model of estimating joint motions. Once both the trustworthiness of the accelerometer and the magnetometer are calculated, we multiply them to form the final trustworthiness.

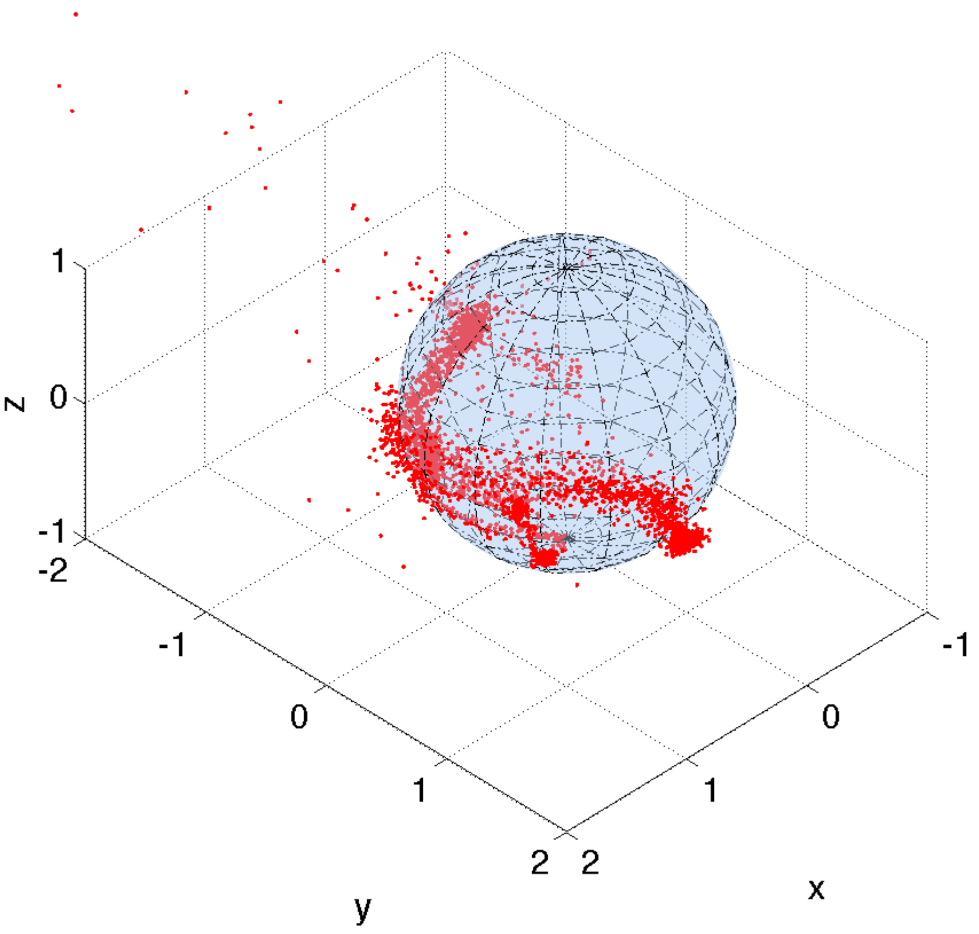


Figure 3.16. Accelerometer sphere

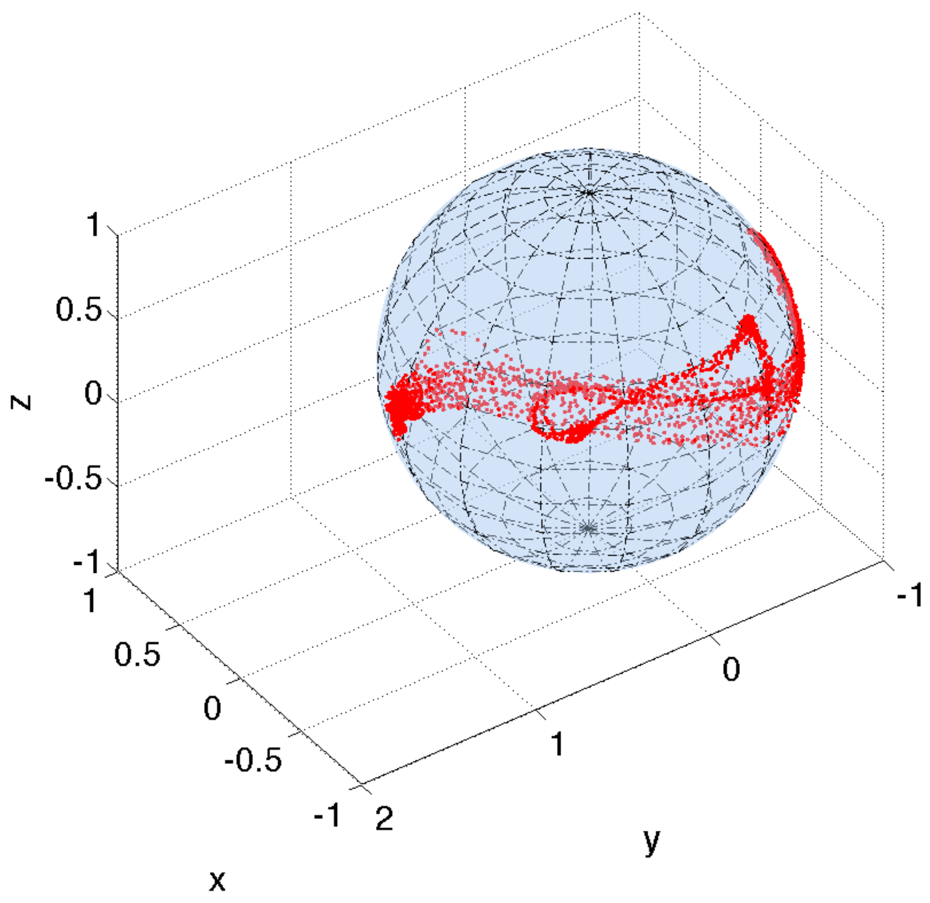


Figure 3.17. Magnetometer sphere

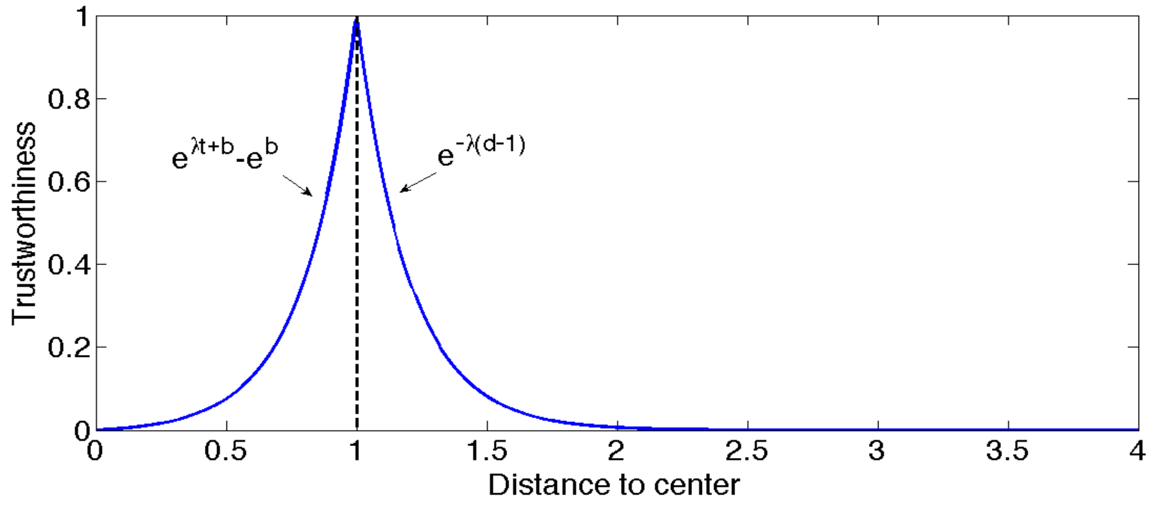
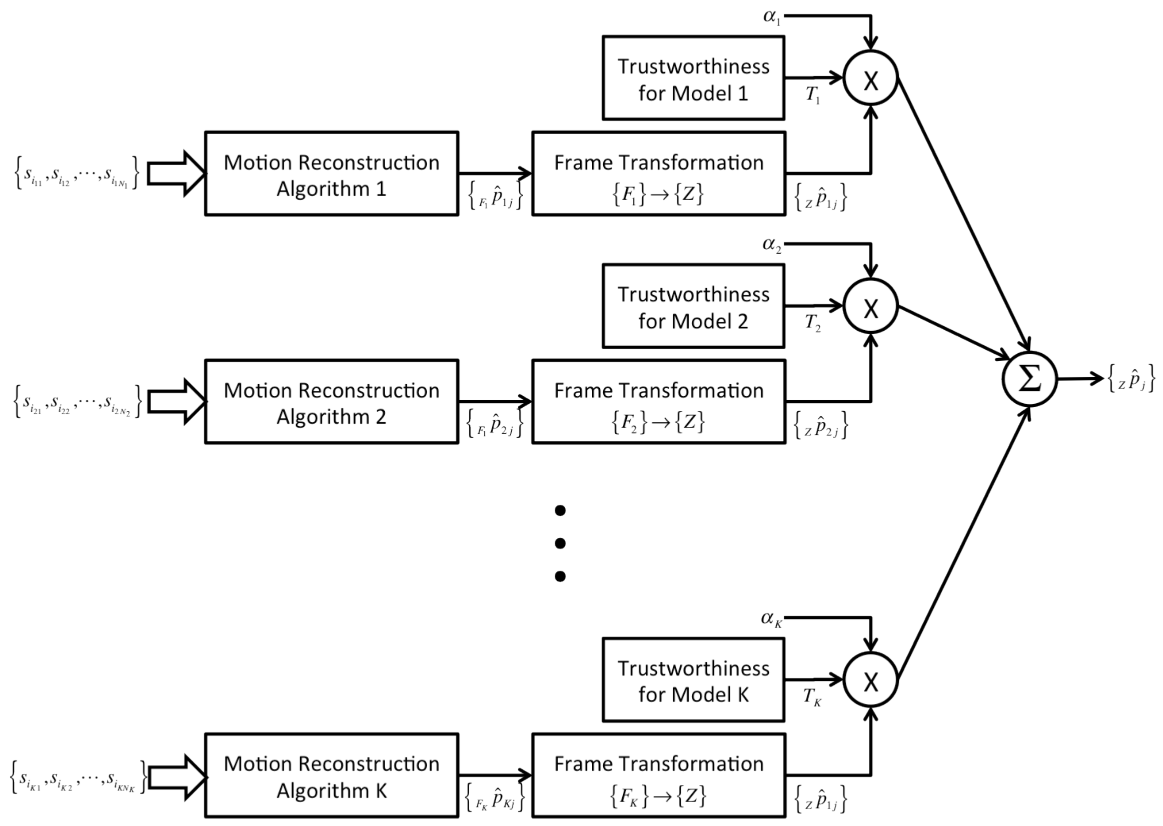


Figure 3.18. Trustworthiness of IMU model



α_1
Trustworthiness for Model 1
T_1
X
$\{s_{i_{11}}, s_{i_{12}}, \cdots, s_{i_{1N_1}}\}$
Motion Reconstruction Algorithm 1
$\{{}_{F_1}\hat{p}_{1j}\}$
Frame Transformation $\{F_1\} \to \{Z\}$
$\{{}_{Z}\hat{p}_{1j}\}$
α_2
Trustworthiness for Model 2
T_2
X
Σ
$\{{}_{Z}\hat{p}_{j}\}$
$\{s_{i_{21}}, s_{i_{22}}, \cdots, s_{i_{2N_2}}\}$
Motion Reconstruction Algorithm 2
$\{{}_{F_1}\hat{p}_{2j}\}$
Frame Transformation $\{F_2\} \to \{Z\}$
$\{{}_{Z}\hat{p}_{2j}\}$
α_K
Trustworthiness for Model K
T_K
X
$\{s_{i_{K1}}, s_{i_{K2}}, \cdots, s_{i_{KN_K}}\}$
Motion Reconstruction Algorithm K
$\{{}_{F_K}\hat{p}_{Kj}\}$
Frame Transformation $\{F_K\} \to \{Z\}$
$\{{}_{Z}\hat{p}_{1j}\}$

Figure 3.19. Coordinate fusion

3.2.5 Coordinate Fusion

After we know how to reconstruct human motions using different methods as described in 3.2.2, how to link each model so that they can all expressed in the Earth frame in 3.2.3, and how to weight each method in 3.2.4, we are ready to fuse all models together and have the final algorithm as shown in Figure 3.19.

At first, we have the measurements from various sensors

$$\left\{ s_{i_{mn}} \middle| m = \{1,2,\cdots,K\}, n = \{1,2,\cdots,N_m\} \right\} .$$

The index m means that we have a total of K models finding motion trajectories, and index n means that for the model m there are N_m sensor measurements fed to the motion reconstruction algorithm m .

For each set of sensor data m, we develop a motion reconstruction algorithm to find out the reconstructed L joint positions

$$\left\{ {}_{F_m}\hat{p}_{mj} \right\}, \qquad j = 1,2,\cdots,L$$

which are expressed in the frame of reference F_m . These estimated positions are then transformed to a basis frame of reference $\{Z\}$ using

$$\left\{ {}_{Z}\hat{p}_{mj} \right\} = {}^{Z}_{F_m}\hat{R} \left\{ {}_{F_m}\hat{p}_{mj} \right\} \qquad j = 1,2,\cdots,L$$

so that every estimated position from all motion reconstruction algorithms is expressed in the same frame of reference.

After we expressed estimated positions in the same frame $\{Z\}$, we find the trustworthiness for each model

$$T_j, \qquad j = 1,2,\cdots,K$$

We also define a set of prior multipliers

$$\alpha_j, \qquad j = 1,2,\cdots,K \ .$$

This set of prior multipliers defines our confidence regarding the environments and conditions under which the different motion reconstruction algorithms are operating. For example, in using

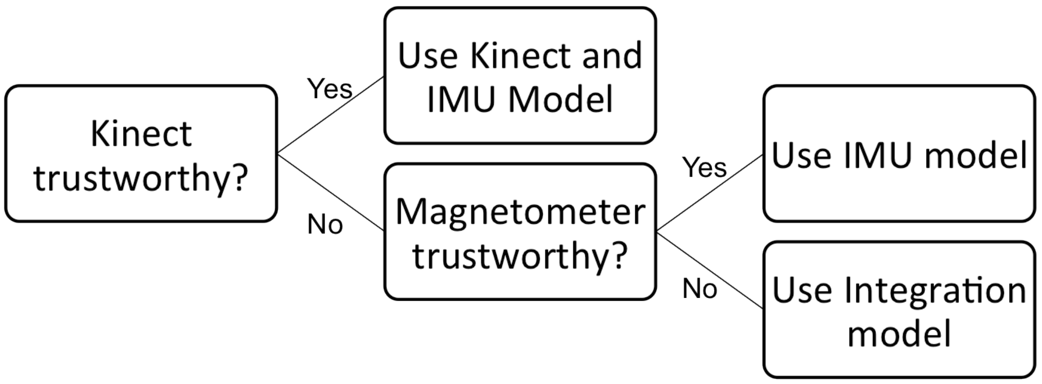


Figure 3.20. Final design flow

the IMU model indoor we would decrease the prior multiplier corresponding to the IMU model, since we know that the magnetometers are generally not accurate when indoors.

The final estimate of joint positions in the base frame $\{Z\}$ is then expressed as

$$\left\{ {}_Z p_j \right\} = \sum_{i=1}^{K} \alpha_i T_i \left\{ {}_Z p_{ij} \right\}, \qquad \sum_{i=1}^{K} \alpha_i T_i = 1,\ j = 1,2,\cdots,L \tag{3.45}$$

The condition $\sum_{i=1}^{K} \alpha_i T_i = 1$ in equation (3.45) ensures that the final estimated positions are normalized.

Equation (3.45) and Figure 3.19 describe the most general case of how we can fuse different models and frames of reference to track human motions in a more accurate way. In this study, we have three ways of motion tracking, which are the integration model in 3.2.2.1, IMU model in 3.2.2.2, and Kinect and IMU model in 3.2.2.3. By combining these three models and specializing the general model in Figure 3.19, we end up with the final design flow in Figure 3.20. In this figure, we first examine if the Kinect trustworthiness is high enough; if yes then we will use the Kinect and IMU model. This is achieved by setting the prior multiplier α_i corresponding to Kinect and IMU model high relative to multipliers corresponding to the IMU and integration models in equation (3.45). If the trustworthiness of the Kinect is not high enough, we use the IMU model to estimate motions, which occurs in equation (3.45) by setting the multiplier of IMU model higher than that of integration model. In this study, we use Earth frame of reference $\{A\}$ as the base frame, and thus every estimate of trajectory is expressed in NED base.

3.3 Experiments and Simulations

In this study, for motion reconstruction we did several types of experiments, which can be categorized into 3 parts. The first part was done outside, where we have good magnetometer measurements. In this part we want to show that the use of complementary filters, which combines high frequency part of gyro and low frequency part of magnetometer and accelerometer, achieves higher accuracy than using them alone. The second part was done indoors, where we have the Kinect to measure joint positions but the magnetometer measurements are assumed to be inaccurate. In this part we want to show that by using the Kinect to replace the magnetometer indoors we can more accurately track joint motions. The third part simulates the time when Kinect data are sometimes unavailable, and we use the coordinate fused model described in 0 to overcome this.

We begin by introducing the devices used for this study, the different types of experiments performed, and the reconstruction error calculations. We then discuss the experiments.

3.3.1 Devices

We used Sparkfun 9 degrees of freedom IMU sensor chips [85] and the Microsoft Kinect (see 3.2.1.1.3.4) for the experiments. These Sparkfun sensors consist of a tri-axial accelerometer (see 3.2.1.1.3.1), gyro (see 3.2.1.1.3.2), and magnetometer (see 3.2.1.1.3.3). The sensors have a sampling rate of 50 Hz and the Kinect has a sampling rate of between 15-30 Hz. Since the sampling rates for the Kinect and IMU are different, and they are not synchronized to each other when doing experiments, we synchronize and re-interpolate the collected data to 50 Hz after collecting them. These two sensors were placed on the subject's right arm using adjustable velcro straps, shown in Figure 3.21: one was placed in the middle of the forearm and the other was placed in the middle of the upper arm. Since there are no handshake communications between 2 sensors mounted in the upper arm and the forearm, when doing experiments we asked the subject to lift his/her heels and hit them hard to the ground for several times to make significant signatures in the collected signals which helped us to align the measured data.

As for Kinect setup, since the best detection range for the Kinect is from 1.8 to 2.4 meters [23], we placed the Kinect 2 meters in front of the subject. When turned on, the Kinect will record 20 joint positions and write them into a log file. We asked the subject to lift his/her arms before and after doing experiments to serve as significant signatures both in Kinect and sensor data. We then align the data sets.

3.3.2 Performed Experiment Types

The general process of each experiment consists of training and testing procedures. During the training procedure, the subject was asked to draw a square for 3 times. This square drawing

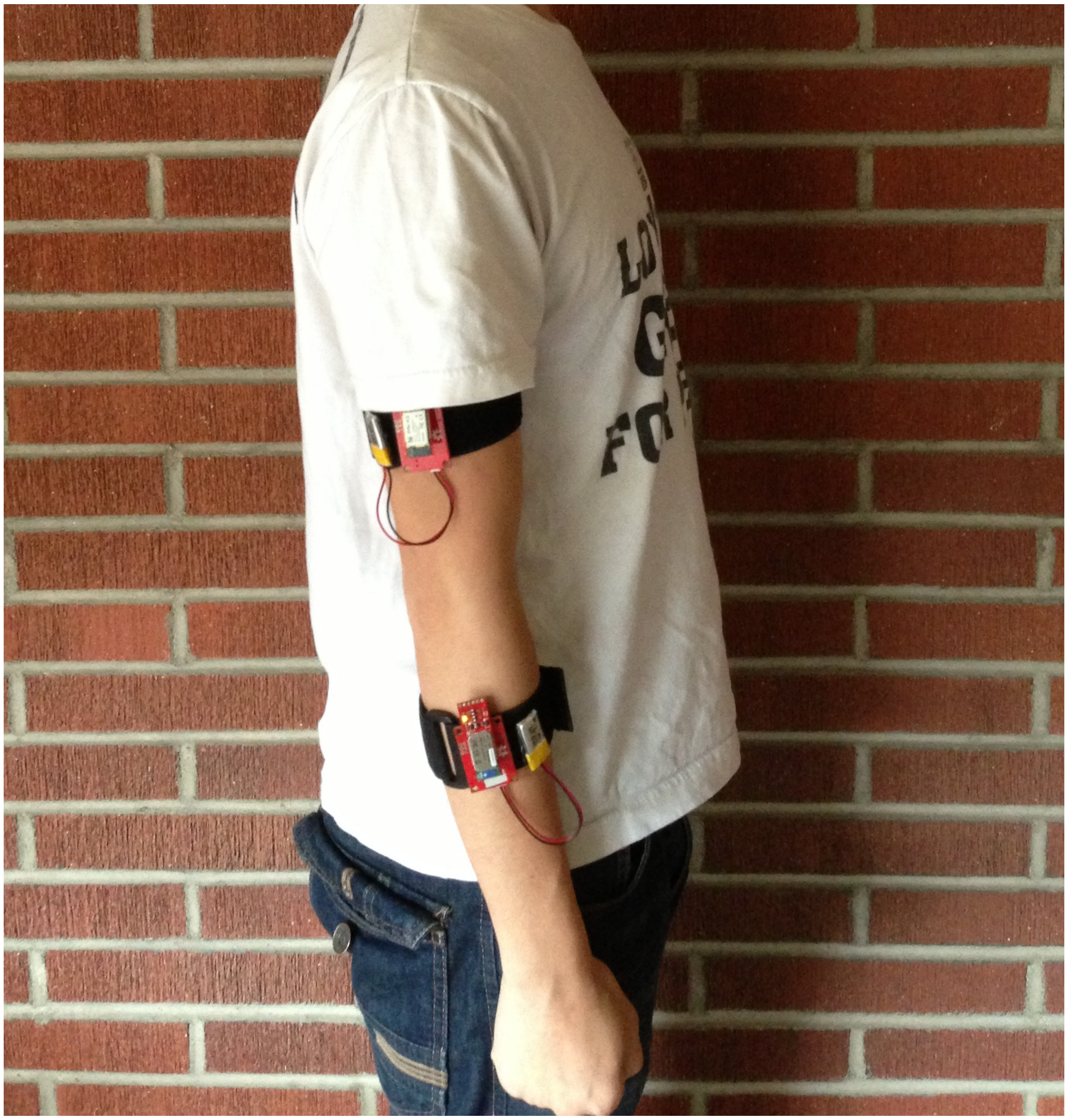


Figure 3.21. Sensor placement

procedure was used to find the best set of parameters, which would be used in the testing procedure as described in 3.2.1.4. In the training procedure there were 5 possible experiments performed in each part, which are square, triangle, bookshelf tapping, bookshelf sliding, and vertical arm-swing experiments. These 5 experiments are described below.

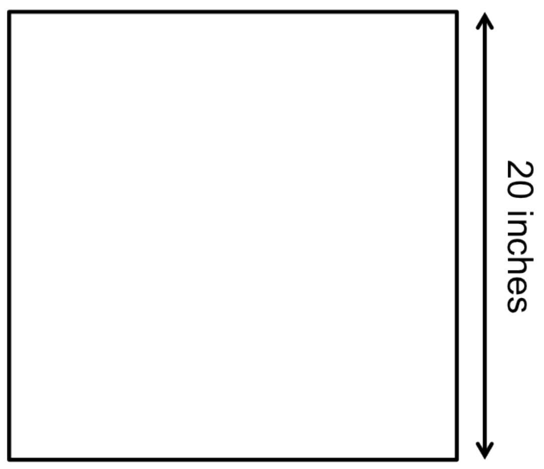


Figure 3.22. Drawn square

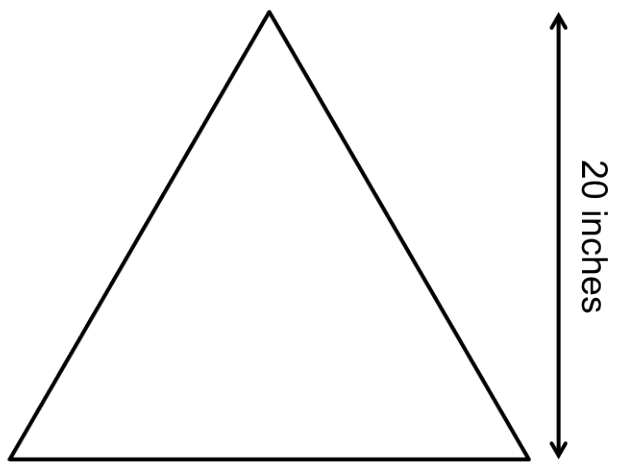


Figure 3.23. Drawn triangle

3.3.2.1.1 Square and Triangle Experiments (Shape Drawing)

In the square experiment, we placed a square of length 20 inches in front of the subject. The subject was asked to follow the sides of a clearly marked square in the clockwise direction for 10 times while we recorded sensor data. The subject stood around 20 inches in front of the square so that when the arm was moved the shoulder was stationary. Each shape took no more than 5 seconds to complete, and during the experiment, we did not instruct subjects how to draw, that is, how the subject bent and twisted the arm. Figure 3.22 shows the drawn squares.

For the triangle experiments, the subject was asked to draw a triangle of height 20 inches, and also repeated for 10 times. All other settings are the same as those of the square experiment. Figure 3.23 shows the triangle.

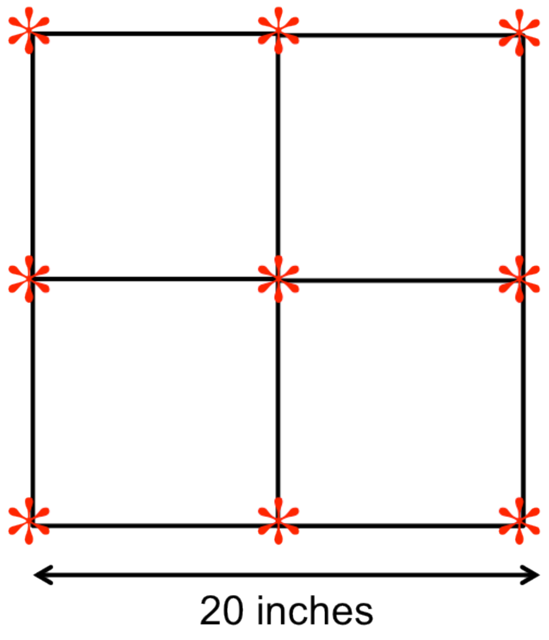


Figure 3.24. Bookshelf reaching shape, red starts indicating vertices to be touched

3.3.2.1.2 Book-reaching experiments

To simulate a more lifelike activity like reaching for a book on a bookshelf, we performed two experiments that involved sliding and tapping of the subject's hand using the square shape as a template. We partitioned the square of side length 20 inches into a 3x3 array and the subject would tap each vertex in the array. These actions were repeated for 10 times, and then we calculate the distances of the estimated vertex positions versus the ground truth. The shape is illustrated in Figure 3.24.

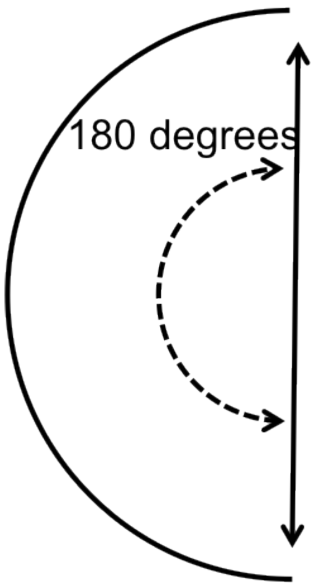


Figure 3.25. Vertical arm swing shape

3.3.2.1.3 Vertical Arm-swing Experiments (Arm Lifting)

Lastly, to simulate how high a subject could reach with our experimental setup, we performed an arm raising experiment where the subject would raise their arm vertically upwards so that they trace out a half circle with their arm as the radius of the circle. During this experiment, the subject's arm was kept straight and each arm swing took no more than 3 seconds to complete. Figure 3.25 shows such patterns.

3.3.2.2 Error Measurements

We define two error measures: the best-fit shape error, and the length-estimation error.

3.3.2.2.1 Best-fit Shape Error

In the best-shape error, for each kind of shape we use the reconstructed vertices to define the corresponding ground truth shape. For example, in the square experiments, for each time after we draw a square there would be 4 reconstructed vertices. We then find the best-fit square given those 4 vertices such that the total distance of the reconstructed vertices to the vertices of the best-fit square is minimized. In the vertical arm-swing experiment, we first calculate the range of angles through which the subject's arm has swung. We then took the mean of that range of angles as the mid-angle, then plus and minus 90 degrees. The arc defined by these 180 degrees is the best-fit arc. Figure 3.26, Figure 3.27, and Figure 3.28 illustrate these best-fit shapes for squares, triangles, and vertical arcs respectively. After we found the best-fit shape, we calculated the closest distances from reconstructed paths (green curves) to the best-fit shape.

We did not define the best-fit shape error for book-reaching experiments.

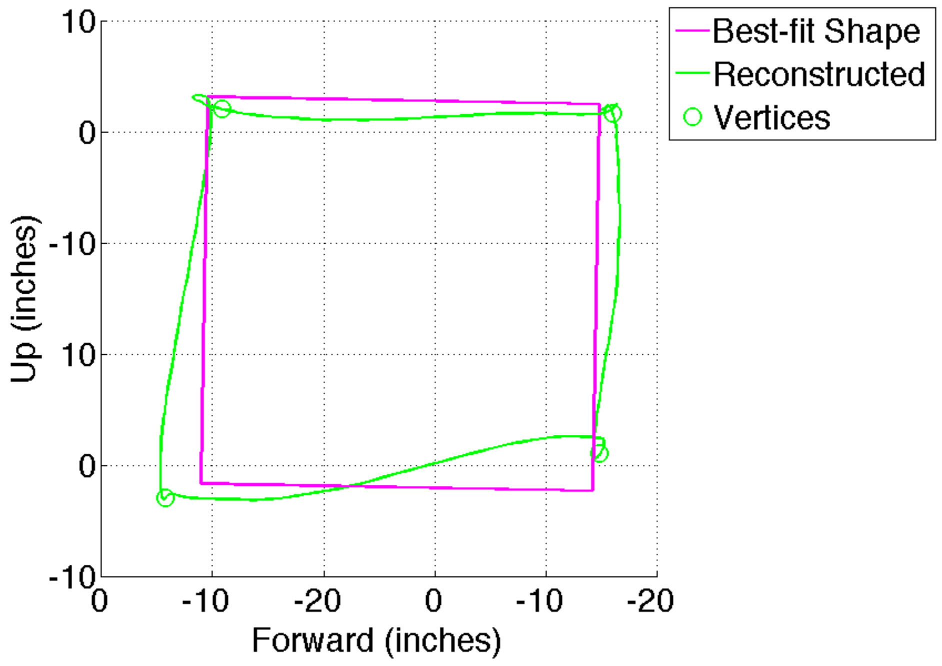


Figure 3.26. Reconstructed and best-fit square

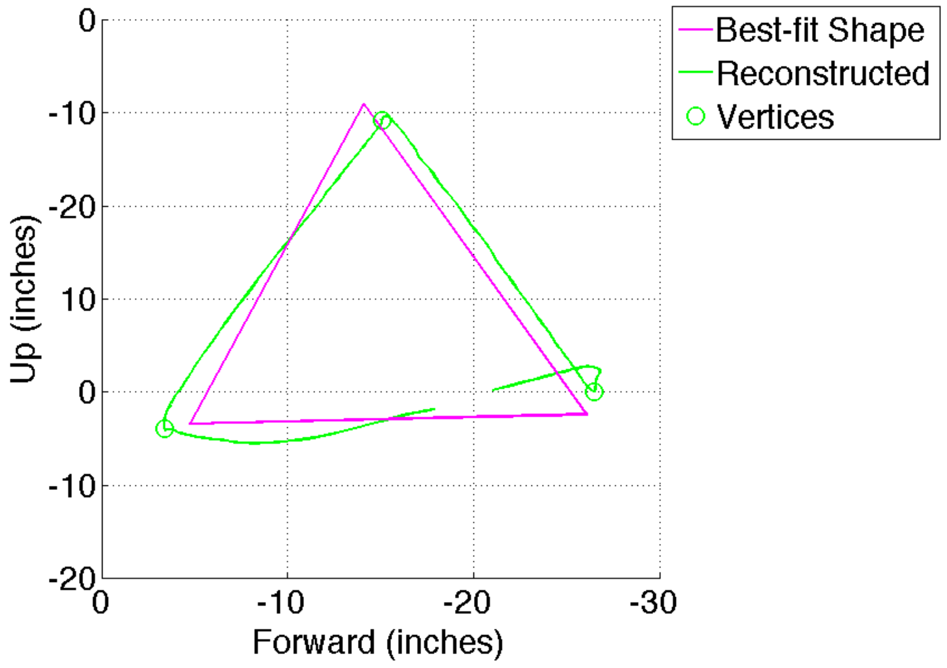


Figure 3.27. Reconstructed and best-fit triangle

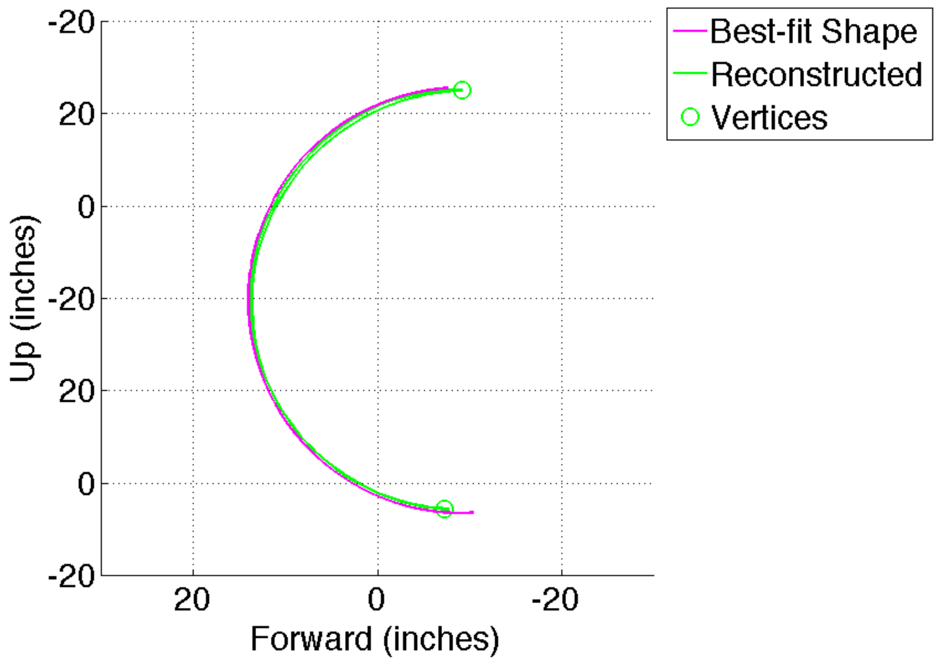


Figure 3.28. Reconstructed and best-fit vertical arc

3.3.2.2.2 Length Estimation Error

In length-estimation error, we calculated the length of each side of reconstructed pattern, and compared them to the ground truth. In the square and triangle experiments, we calculated the edges linked by reconstructed vertices and compared them to the true length of the shape as described in Figure 3.26 and Figure 3.27. For example, in Figure 3.29, the edges connected by reconstructed vertices are 18.0, 19.7, 15.5, and 17.6 inches, where the true length of the square is 20 inches, and therefore we have the sum of absolute error of 9.2 inches, or 11.4% of error. For the triangle, in Figure 3.30, the edges connected by reconstructed vertices are 19.3, 22.1, and 23.4 inches, where the true length of the square is $40/\sqrt{3}$ inches, and therefore we have the sum of absolute error of 5.1 inches, or 7% of error.

For the sliding and tapping book-reaching experiment, we calculated the distance of adjacent reconstructed vertices. Since there are 9 vertices, we calculated 6 horizontal and 6 vertical distances as shown in Figure 3.31. As shown in Figure 3.31, the edges formed by adjacent vertices are of lengths ranging from 7.4 inches to 14.7 inches, where the true length is 10 inches, and therefore we have the sum of absolute error of 18.9 inches, or 15.7% error.

For vertical arc-swing experiments, we calculated the range of angles that the subject's arm had swung. As shown in Figure 3.32, this subject had swung 177.5 degrees, where the true range of the arc is 180 degrees, and therefore we have the absolute error of 2.5 degrees, or 1.4% of error.

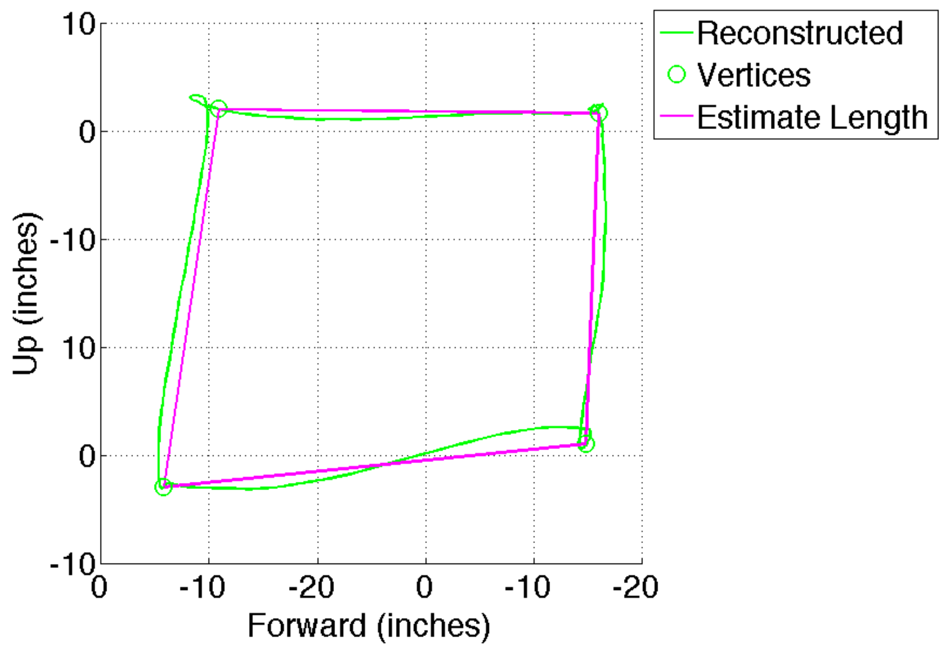


Figure 3.29. Reconstructed square and estimated length

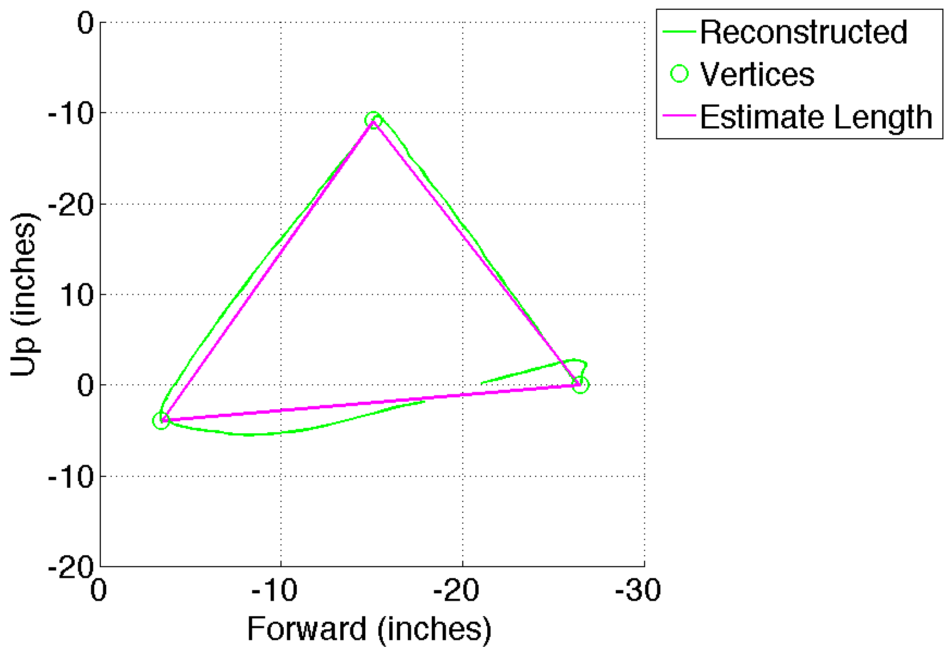


Figure 3.30. Reconstructed triangle and estimated length

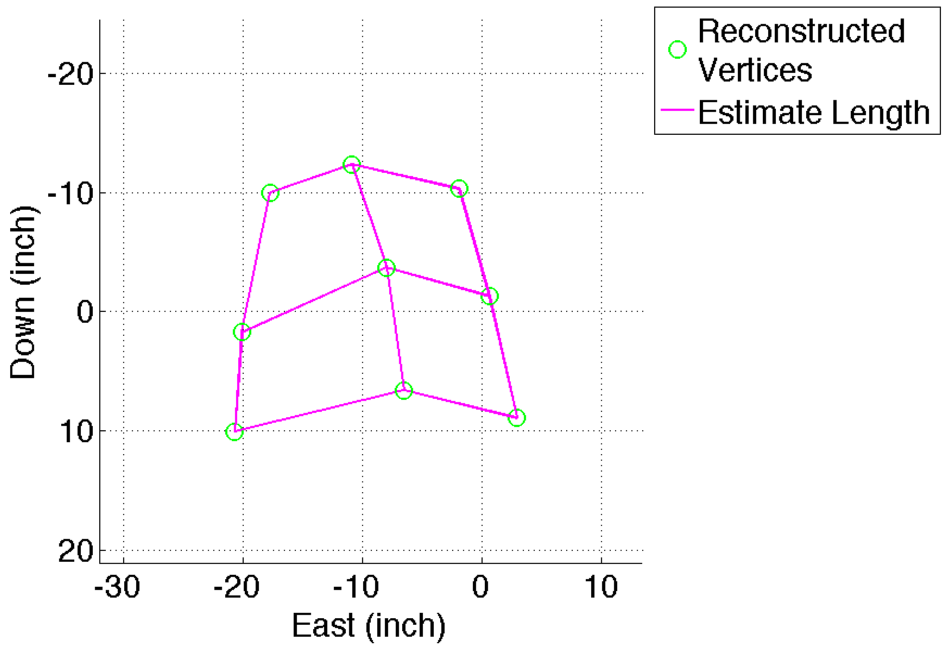


Figure 3.31. Reconstructed book reaching and ground truth length

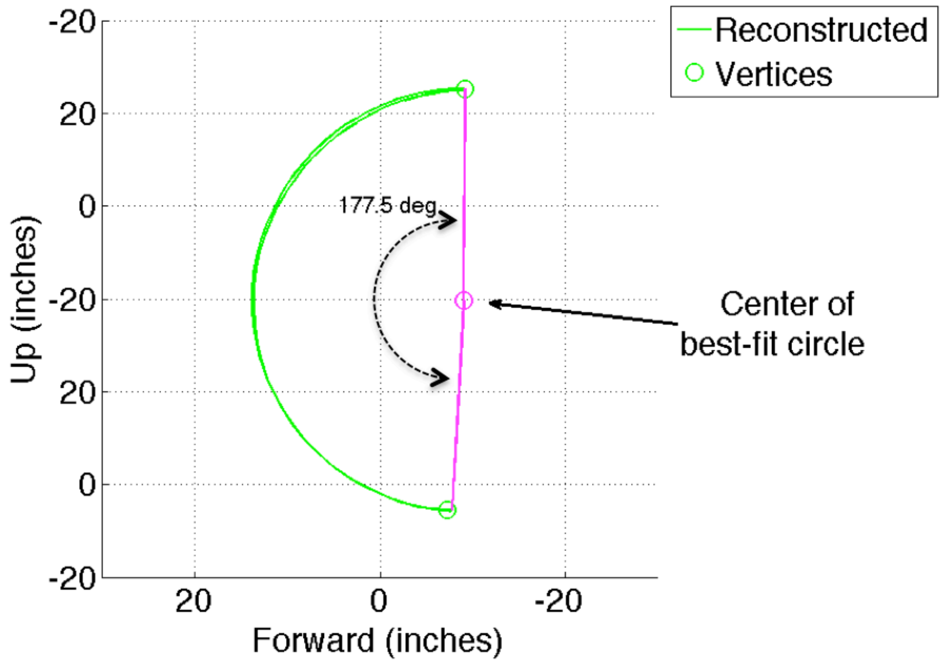


Figure 3.32. Reconstructed vertical arm swing and estimated angle

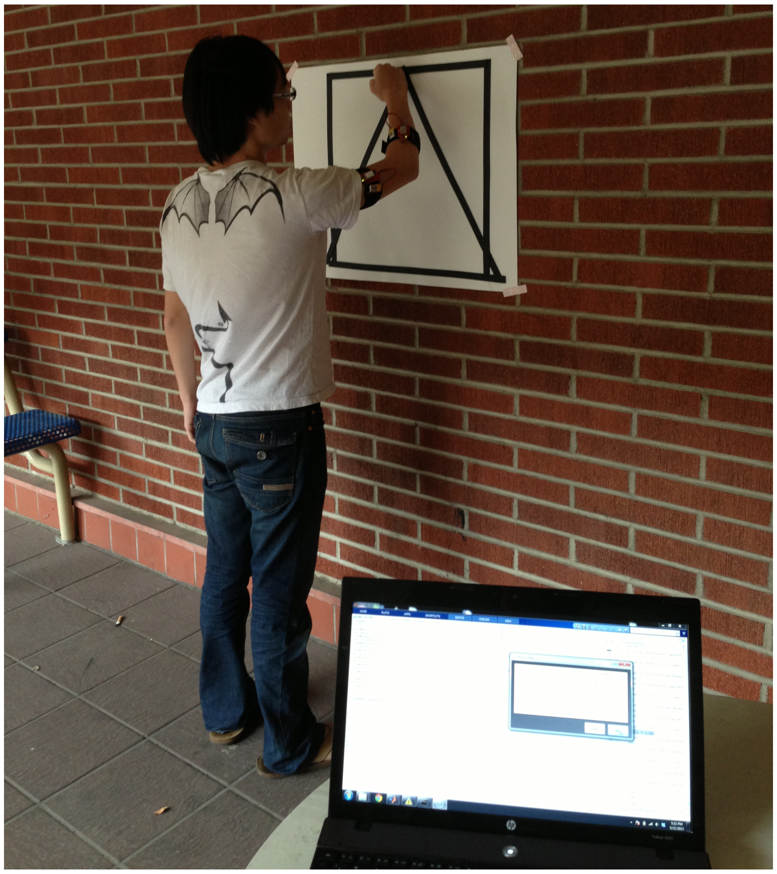


Figure 3.33. Experiment setup when outdoor

3.3.3 Experiment Part 1 – Outdoor Experiments

In this part, the experiments were done outside, where good magnetometer measurements were available. We used the Sparkfun 9DoF IMU chip with sampling rate 50 Hz. Two sensors were placed in the middle of the upper arm and the forearm as depicted in Figure 3.6. Twelve subjects participated in the experiments. We did three types of experiments in this part. The setup of this part of the experiments is shown in Figure 3.33. In this figure, the subject was standing about 20 inches in front of the shape, while the shape was created using tape and posterboard. There was a computer next to the subject to collect sensor data, which are transmitted via Bluetooth. Since the sampling rates of the two sensors were not exactly the same, and the clocks were not synchronized, the subject was asked to raise his/her heels and hit them hard on the ground. This produced significant signals in the accelerometer measurements so later on we can synchronize signals from the sensors. The subject was asked to do the heel-strike actions before and after the whole experiment. In each experiment, the subject drew the 20’ x 20’ square for 3 times first, which served as the training procedure to find the optimal parameters for complementary filters as in 3.2.1.4. Then the subject would do the designated experiments 10 times, which completed the whole experiment process.

In the first experiment subjects drew squares and triangles on a wall. The shape of squares and triangles are described in Figure 3.22 and Figure 3.23. Each shape took no more than 5 seconds to complete and was repeated for 10 times.

The second experiment was conducted to simulate reaching for and grasping books. In this experiment we portioned a 20’ by 20’ square hanging on a wall into a 3×3 array as described in Figure 3.24. Then we tapped each point 10 times to simulate patients taking books out of a shelf.

In the third experiment, the subjects were asked to perform 10 rotations around their shoulders from bottom to top for 10 times. This vertical arc is depicted in Figure 3.25.

After the experiments, we used the combination of non-linear complementary filters (direct and passive type), human biomechanical models, and parameter optimization of complementary filters to process the data, as described in 3.2.2.2.

3.3.4 Experiment Part 2 – Indoor Experiments

For this part of experiment, we continued to use the Sparkfun sensors but added the Microsoft Kinect as a way to increase the accuracy of our results, since there is severe interference of magnetometer measurements in many indoor settings. There were a total of ten subjects who participated in our experiments.

To see whether or not our choice of incorporating the Kinect as a means of making measurements and collecting data was accurate, we performed several experiments. The setup of our experiments is shown in Figure 3.34.

The frame was constructed using PVC pipes for stability and consistency in the experimental setup. This setup enables a thin, hanging shape in front of the subject which gives us an accurate ground truth; in addition, because the shape was thin enough, it would not block the Kinect's ability to recognize joint positions. We fastened threads to our shape in order to suspend it through holes in the PVC pipes and used velcro to hold the wires in place during the experiment. These wires not only helped us assemble and disassemble the experiment quickly, but also allowed for easy adjustment of the height of the shape corresponding to the subject’s height. The subject would extend their arm forward and parallel to the ground with their fist clenched, and we would adjust the height of the shape so that the subject's fist was in the center of the shape. This is necessary because we need to make sure that the subjects do not move their shoulders. We placed the Kinect on a movable cart 2 meters in front of the subject and facing away from the door so that it would not accidentally record other skeletons of passersby. We attached two 9DoF sensors to the subject’s right arm in a similar fashion as shown in Figure 3.34 using adjustable velcro straps.

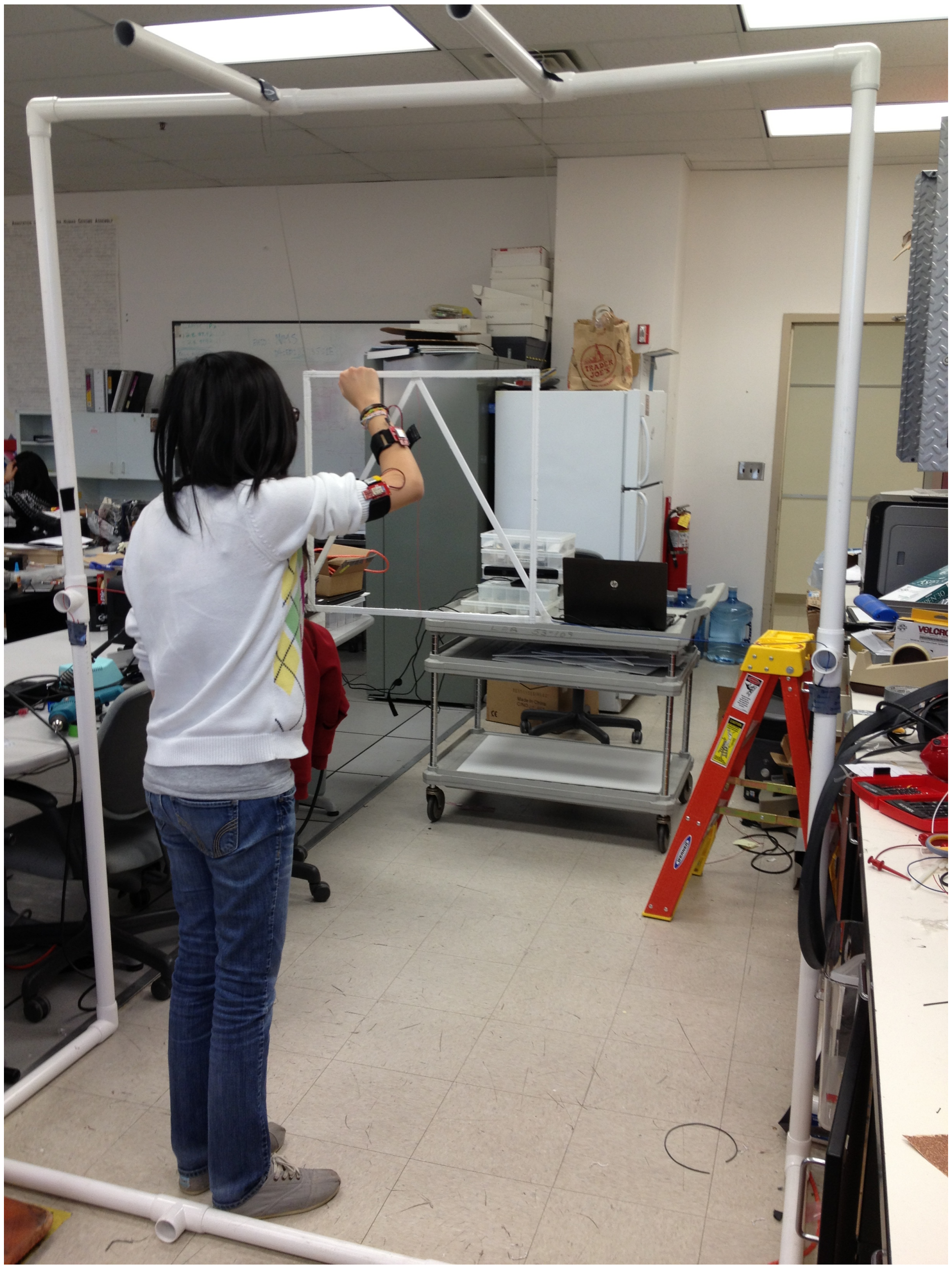


Figure 3.34. Experiment setup when indoor

Using the setup shown in Figure 3.34, we had the subject first perform signature and training motions before performing any of the experiments: three heel strikes, one arm raise, and three square traces. The arm raise was performed by having the subject raise and three square traces. The arm raise was performed by having the subject raise their arm vertically upward from their side which traces a half circle in the air with their arm acting as the radius of the circle (See Figure 3.25 for more details). The three square traces are performed by having the subject trace the square with their hand in a fist in a clockwise fashion starting with the upper left corner, as shown in Figure 3.22.

Because the sensors did not start collecting data at the same time, the three heel strikes are a way to sharply disrupt the sensor data so we know exactly how to align the signals later, as was the case in the previous section. The arm raise was used so that we can align the sensor data with the Kinect's data since a heel strike is not clearly visible from the Kinect's point of view. Lastly, the square traces are used as training in order to find the best filter parameters of the complementary filters as in 3.2.1.4.

Similarly to the previous section, we performed three experiments after the completion of the motion signatures: shape tracing, tapping, and arm-raising. The experiments were performed in the same way as before, but we added the Kinect to collect 3D motion data in addition to the IMU's and we used a new experimental location and setup. This section focuses on the differences and additions that were necessary as we introduced the Kinect into our data collection methods.

First, we performed the shape tracing experiment using a square with a side length of 20 inches and the triangle with a height of 20 inches. The subject would trace the shape for a total of ten times, pausing at each corner for about one second.

To simulate a more lifelike activity like reaching for a book on a bookshelf, we performed an experiment that involved tapping of the subject's hand using the square shape as a template. The positions of the taps are shown in Figure 3.24. We partitioned the square into a

3x3 array and the subject would tap each vertex in the array. This is repeated for 10 rounds for a total of 90 taps.

Lastly, to simulate how high a subject could reach with our experimental setup, we performed an arm raising experiment where the subject would raise their arm vertically upwards so that they trace out a half circle with their arm as the radius of the circle. This shape is shown in Figure 3.25. Likewise, the subjects kept their arm straight during this experiment.

There were several differences between these experiments and the experiments from the previous section. For instance, while using the Kinect, if the subject performed the experiment too quickly, the Kinect had difficulty in keeping up with the subject. In other words, if the movements were too fast, the Kinect could not detect the subject's arm as easily and there would sometimes be a delay between the Kinect and the person's actual arm movement. This was a problem that was especially apparent in the arm raise experiment. To remedy this, we made sure that the subject did not perform the experiment too quickly. Another difference is that we needed to perform more arm raising motion signatures to make sure we aligned the IMU data with the Kinect properly. Lastly, the experimental design and environment were different.

3.3.5 Experiment Part 3 – Coordinate Fusion

The third part of the experiment was conducted to verify our coordinate fusion algorithm described in 3.2.5. We used our current pool of experimental data collected in 3.4.1 to simulate situations where our algorithm would switch to a different model to analyze the data. There are two motivations for improving our previous Kinect model and for creating and testing the fused model. First, the magnetometer was very noisy especially indoors where there can be large amounts of interference contained in a room. Second, sometimes the Kinect cannot always determine the joints of the subject, which results in inferred joint positions. Finally, the Kinect did not perform optimally in the outdoor environment. Therefore, our goal for this simulated experiment was to determine whether or not our algorithm was working properly when the environment suddenly changed, i.e., how well it would switch models according to the trustworthiness described in 3.2.4.

To test the trustworthiness of the Kinect, rather than designing a physical experiment, we simulated the results of the square, triangle, and arm raising experiments by randomly destroying a certain percent of the Kinect's data using our previous data, and set the trustworthiness to 0. These three experiments are described in previous sections. We corrupted the Kinect data in steps of 10% starting from 0% to 100% and examined the error between our reconstruction and the ground truth. Figure 3.35 and Figure 3.36 illustrate the corrupted Kinect signals of right elbow for different percentages and the trustworthiness respectively.

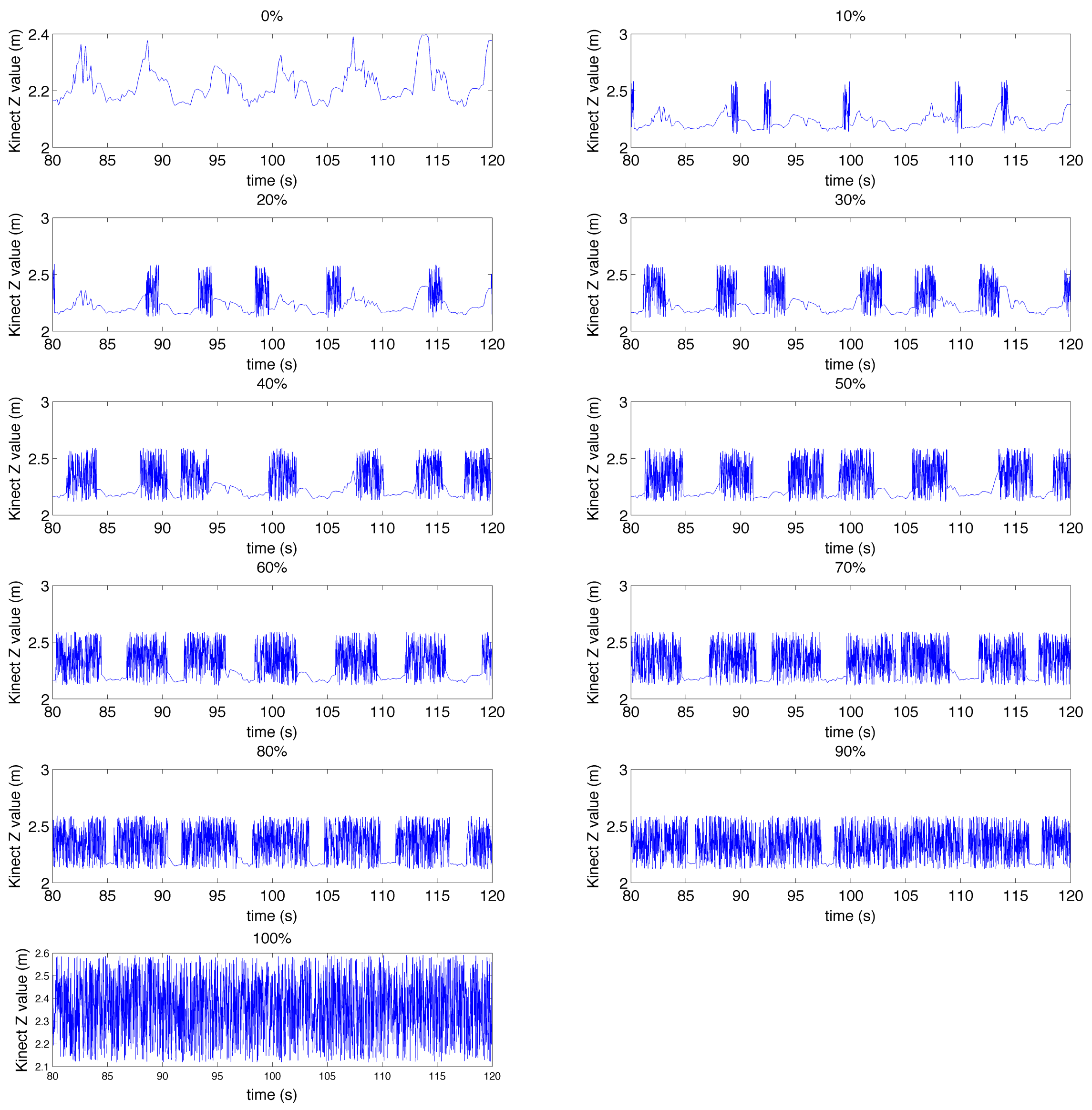


Figure 3.35. Corrupted Kinect signals for different percentages

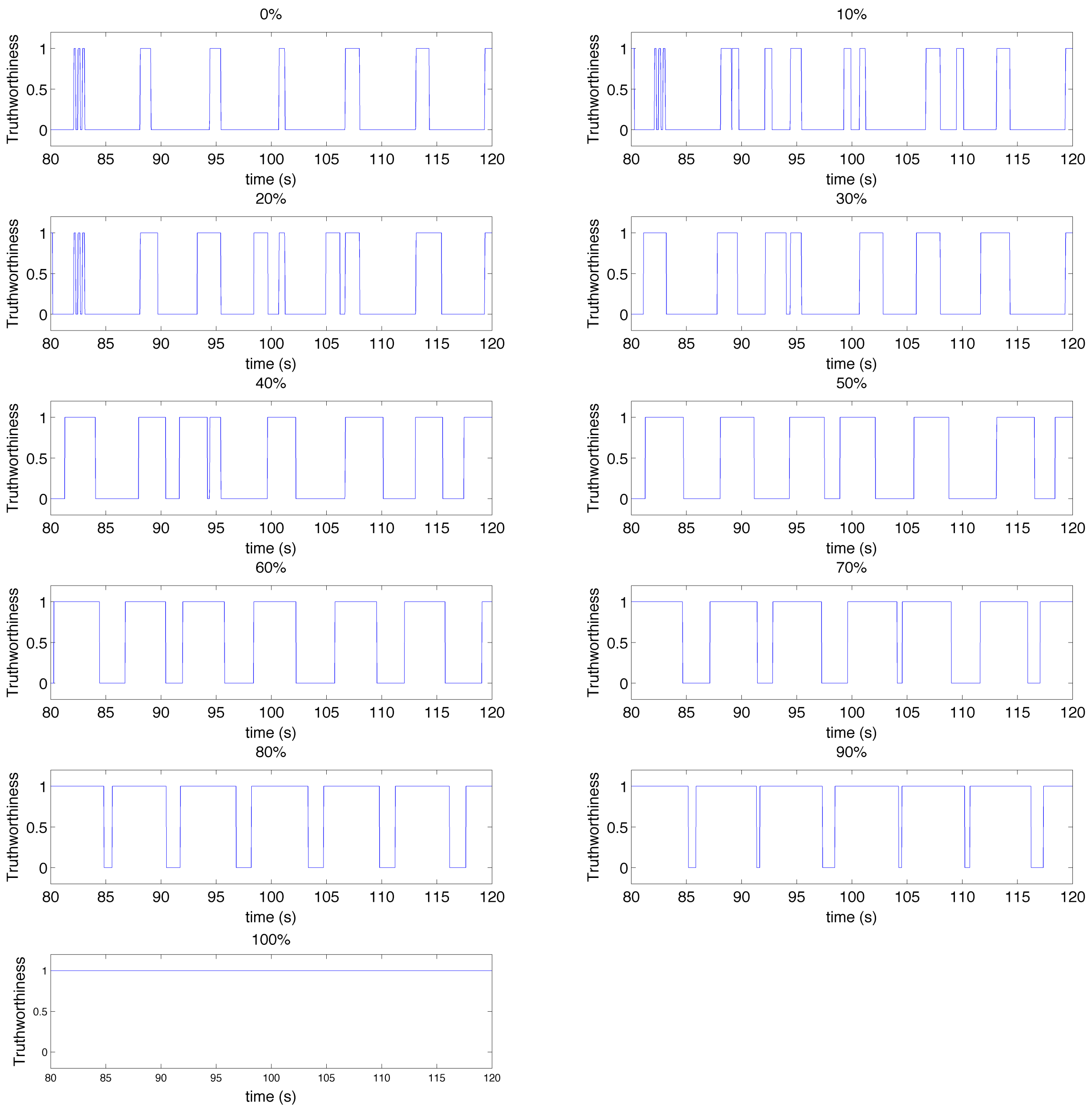


Figure 3.36. Corrupted Kinect trustworthiness for different percentages

3.4 Results

3.4.1 Experiment Part 1 – Outdoor Experiments

This section shows the results of indoor reconstruction using the IMU model described in 3.2.2.2. Table 3.3 and Table 3.4 show the mean and variance of distances to the best-fit shape for square, triangle, and vertical arm-swing experiments respectively. Table 3.5 and Table 3.6 show the mean and variance of percentages of length-estimation error of square, triangle, book reaching, and vertical arm-swing experiments respectively.

The results of 3D motion reconstruction of the fist are shown from Figure 3.37 to Figure 3.39. We show total of 6 reconstructed trajectories using different methods, which are the Kinect raw data, static model using the accelerometer and the magnetometer, direct integration using the gyro signal (dynamic models), the direct filter, and the passive filter. For the first experiment of square and triangle plotting, Figure 3.37 show the 3-dimensional motion reconstruction of the fist movements. Figure 3.38 shows the reconstructed trajectories of triangle-drawing experiments. Figure 3.39 shows the reconstructed result of book-reaching experiments, and finally Figure 3.40 shows the reconstructed result of vertical arm-swing experiments.

Table 3.3. Mean of distances to best-fit shape of outdoor experiments (inches)

Experiments	Static	Dynamic	Passive	Direct
Square arc	4.49	10.78	4.31	4.25
Triangle	7.60	12.48	7.12	7.23
Vertical arc	7.39	9.24	5.94	5.97

Table 3.4. Variance of distances to best-fit shape of outdoor experiments (inches)

Experiments	Static	Dynamic	Passive	Direct
Square arc	1.60	4.19	1.70	1.73
Triangle	4.75	4.91	4.10	4.39
Vertical arc	1.74	2.54	1.36	1.41

Table 3.5. Mean of percentage of length-estimation error of outdoor experiments (%)

Experiments	Static	Dynamic	Passive	Direct
Square arc	10.66	30.47	6.28	5.27
Triangle	10.37	27.17	6.41	5.86
Reaching	17.79	38.86	12.05	13.01
Vertical arc	16.11	21.69	10.42	5.56

Table 3.6. Variance of percentage of length-estimation error of outdoor experiments (%)

Experiments	Static	Dynamic	Passive	Direct
Square arc	0.88	4.84	0.43	0.35
Triangle	0.05	1.90	0.01	0.01
Reaching	4.36	5.18	3.27	4.28
Vertical arc	2.13	4.01	0.16	0.13

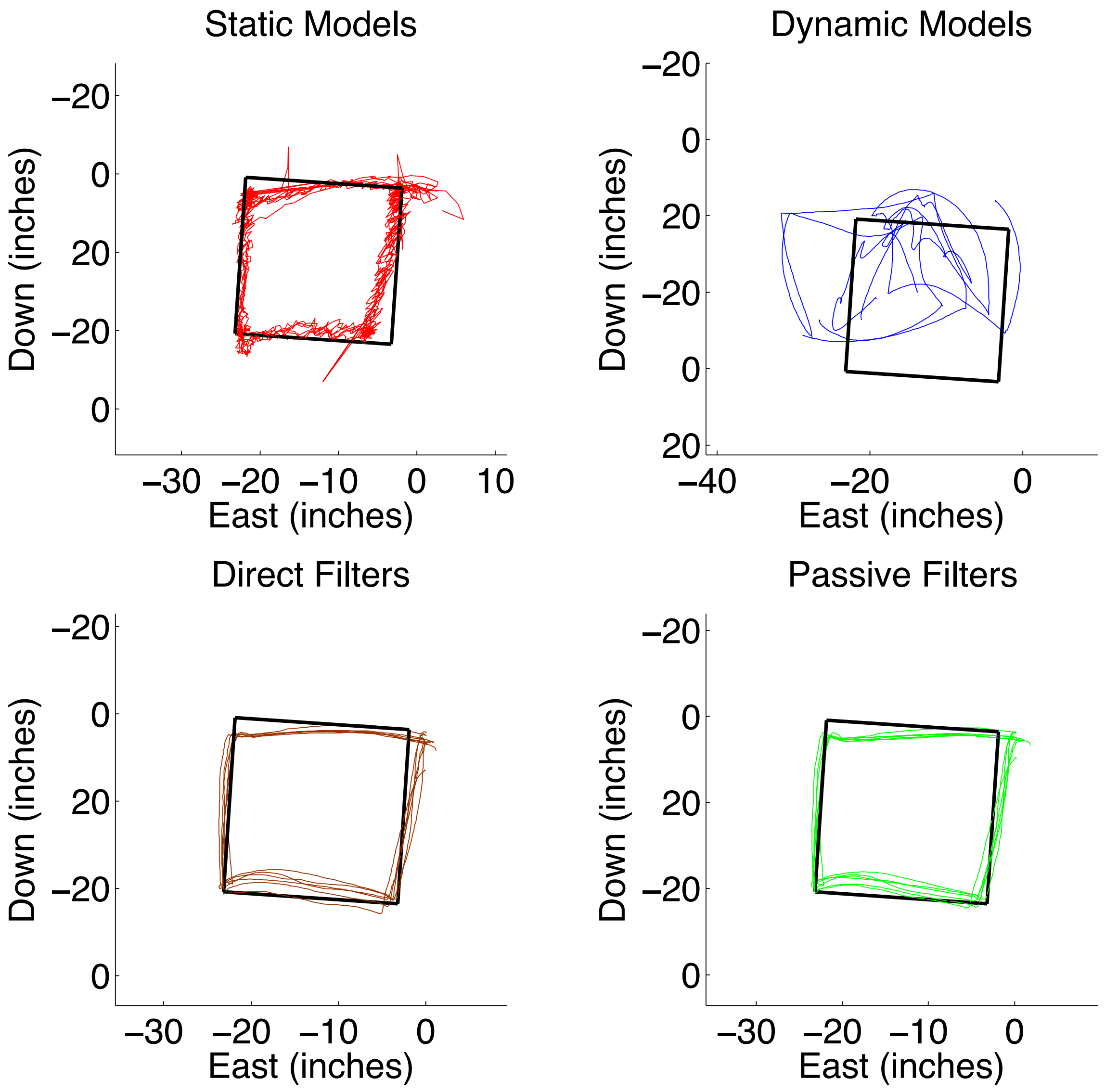


Figure 3.37. Reconstruction of square drawing of outdoor experiments, ground truths are shown in black curves

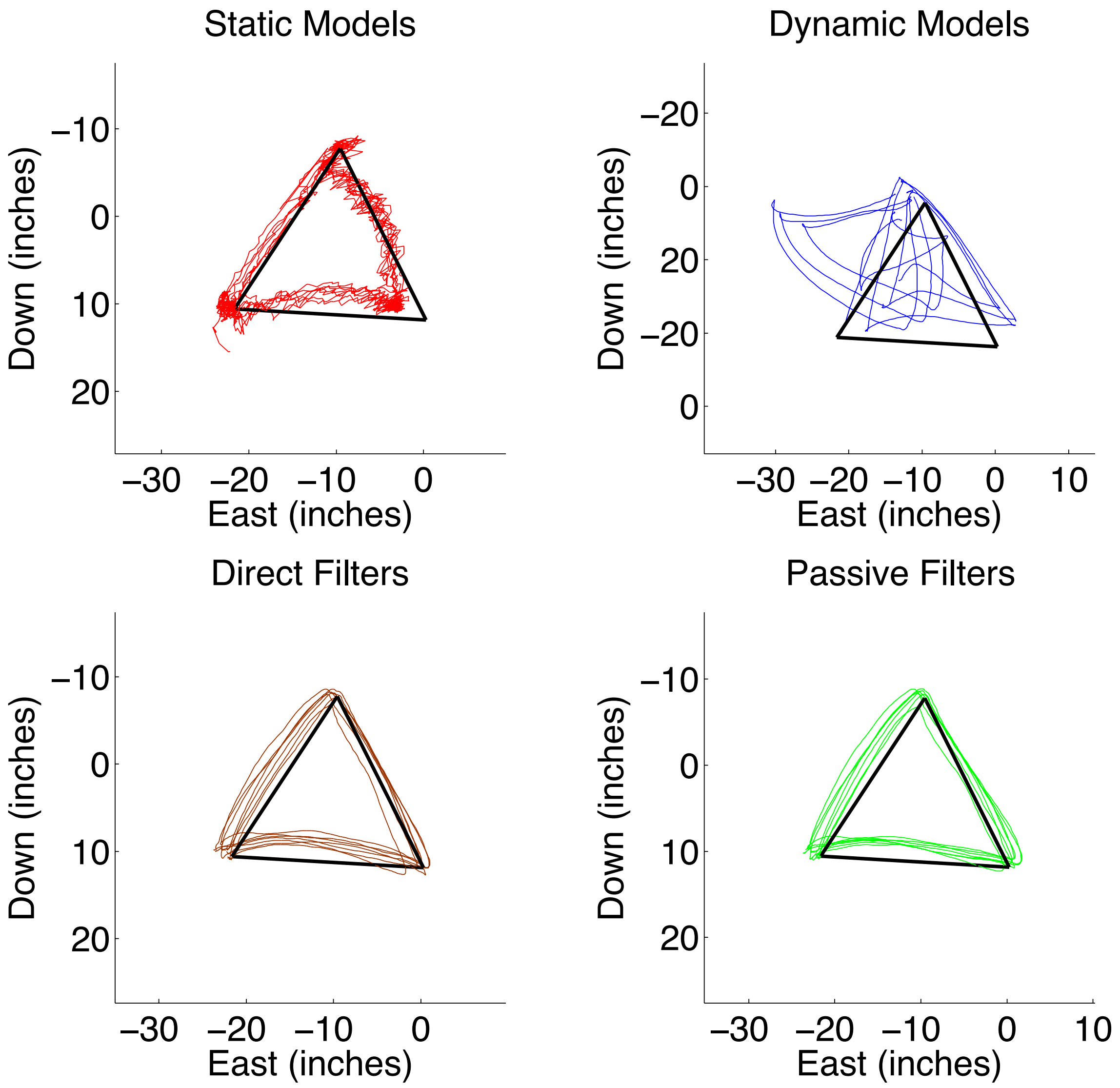


Figure 3.38. Reconstruction of triangle drawing of outdoor experiments, ground truths are shown in black curves

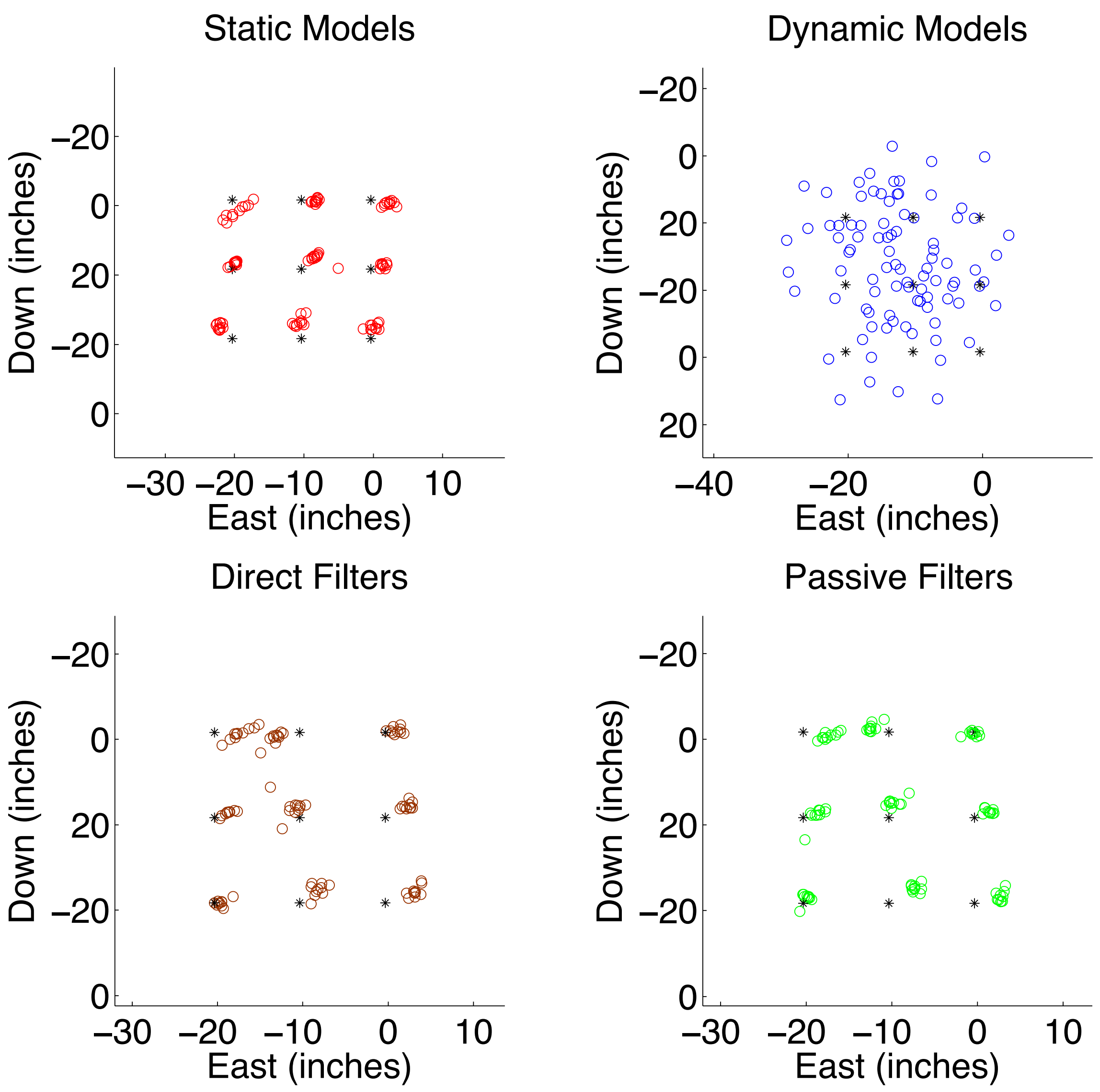


Figure 3.39. Reconstruction of book reaching of outdoor experiments, ground truths are shown in black stars

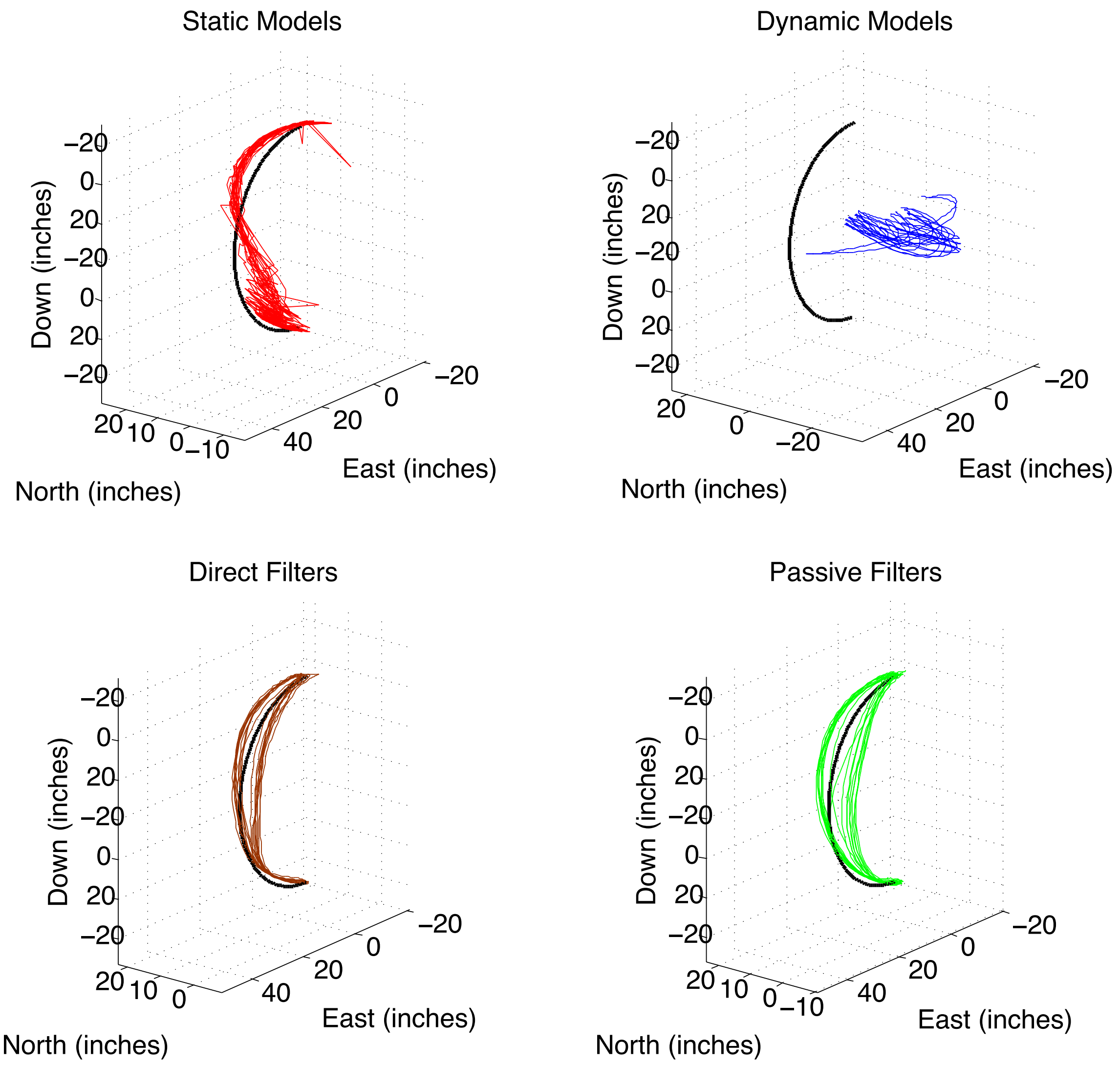


Figure 3.40. Reconstruction of vertical arm swing of outdoor experiments, ground truths are shown in black curves

3.4.2 Experiment Part 2 – Indoor Experiments

This section shows the results of indoor reconstruction results using the Kinect and IMU model described in 3.2.2.3. Table 3.7 and Table 3.8 show the mean and variance of distances to the best-fit shape for square, triangle, and vertical arm-swing experiments respectively. Table 3.9 and Table 3.10 show the mean and variance of percentages of length-estimation error of square, triangle, and vertical arm-swing experiments respectively. These figures show the static result using the magnetometer and accelerometer, Kinect only, Kinect and accelerometer, dynamic model using gyro, and complementary filtered result (passive and direct).

The results of 3D motion reconstruction of the fist are shown from Figure 3.41 to Figure 3.43. We show a total of 6 reconstructed trajectories using different methods, which are the Kinect raw data, static model using the accelerometer and the magnetometer (ACC+MAG), static model using the accelerometer and the Kinect (ACC+KIN), direct integration using the gyro signal (Dynamic models), the direct filter, and the passive filter. For the first experiment of square and triangle plotting, Figure 3.41 shows the 3-dimensional motion reconstruction of the fist movements. Figure 3.42 shows the reconstructed trajectories of triangle-drawing experiments, and Figure 3.43 shows the reconstructed result of vertical arm-swing experiments.

Table 3.7. Mean of distances to best-fit shape of indoor experiments (inches)

Experiments	Static (M+A)	Static (K)	Static (K+A)	Dynamic (G)	Passive (K+A+G)	Direct (K+A+G)
Square	1.33	0.76	0.41	1.04	0.34	0.33
Triangle	1.34	0.78	0.37	1.06	0.36	0.35
Vertical arc	1.77	0.95	1.19	0.94	1.02	1.19

Table 3.8. Variance of distances to best-fit shape of indoor experiments (inches)

Experiments	Static (M+A)	Static (K)	Static (K+A)	Dynamic (G)	Passive (K+A+G)	Direct (K+A+G)
Square	0.291	0.048	0.024	0.347	0.022	0.019
Triangle	0.35	0.07	0.03	0.32	0.03	0.03
Vertical arc	0.61	0.07	0.07	0.06	0.19	0.15

Table 3.9. Mean of error percentages of length-estimation error of indoor experiments (%)

Experiments	Static (M+A)	Static (K)	Static (K+A)	Dynamic (G)	Passive (K+A+G)	Direct (K+A+G)
Square	24.06	24.79	20.44	28.11	13.12	13.54
Triangle	22.54	25.84	16.10	24.86	10.27	9.95
Vertical arc	55.65	17.21	19.54	12.48	6.57	11.09

Table 3.10. Variance of error percentages of length-estimation error of indoor experiments (%)

Experiments	Static (M+A)	Static (K)	Static (K+A)	Dynamic (G)	Passive (K+A+G)	Direct (K+A+G)
Square	0.18	0.11	0.092	1.50	0.029	0.058
Triangle	0.28	0.11	0.28	1.34	0.04	0.07
Vertical arc	8.06	0.32	1.78	1.46	0.61	0.80

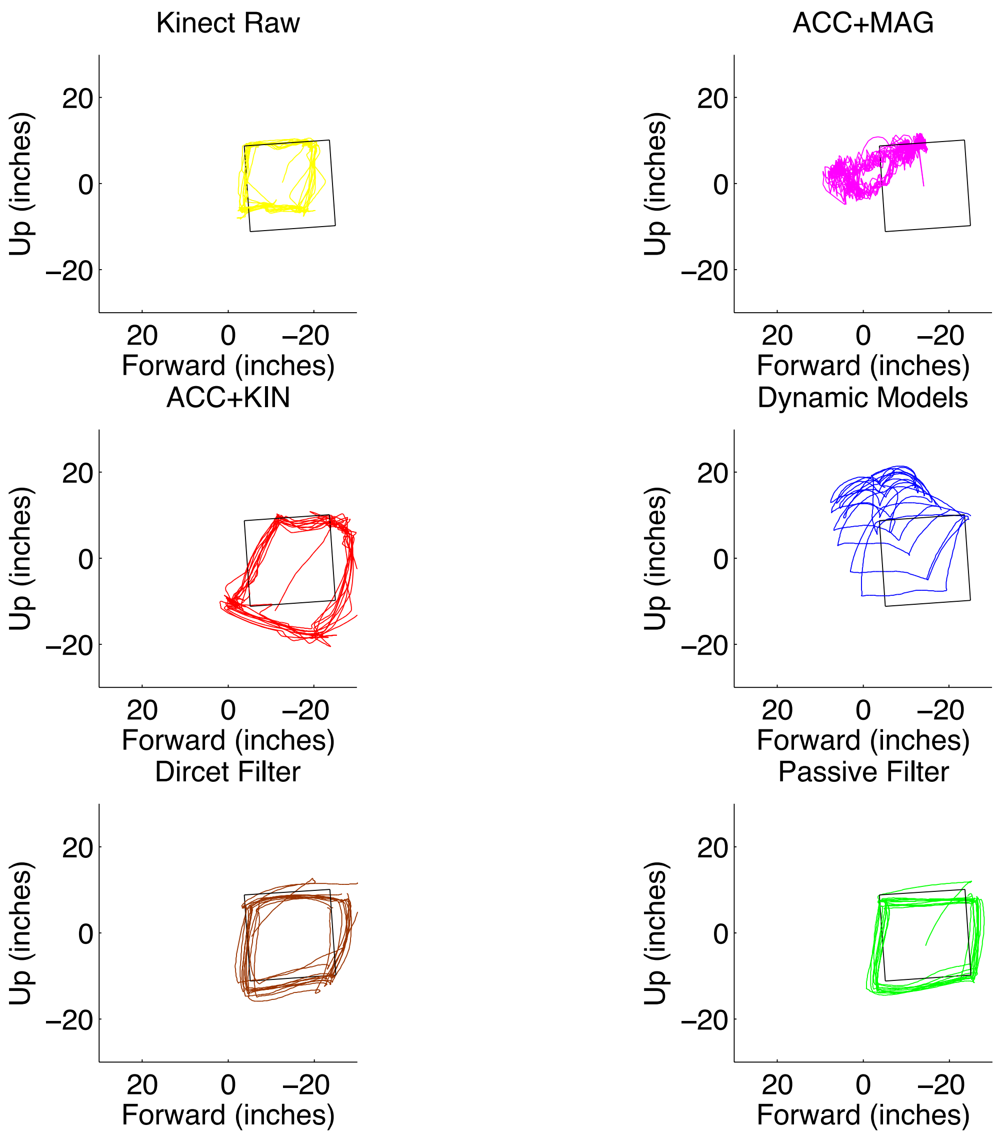


Figure 3.41. Reconstruction of square drawing, of indoor experiments ground truths are shown in black curves

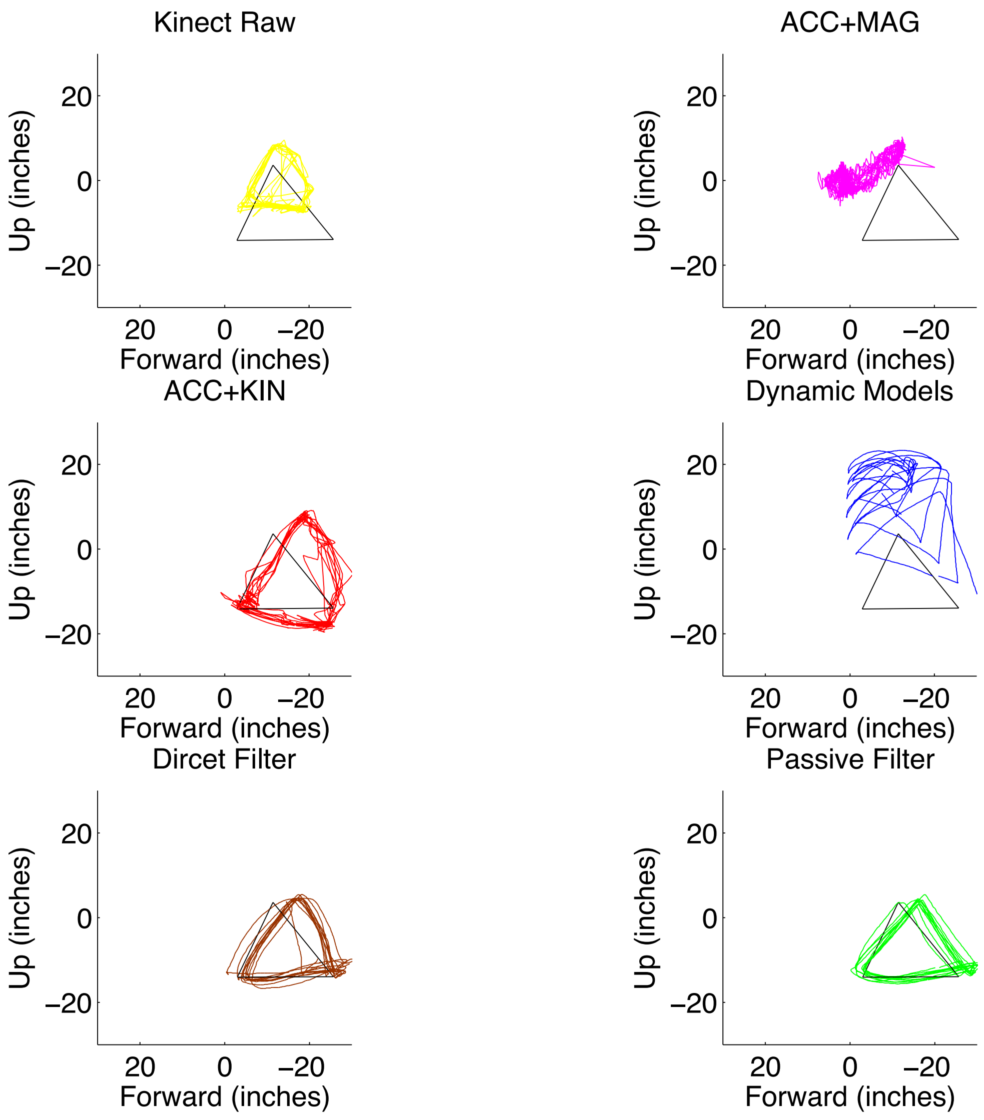


Figure 3.42. Reconstruction of triangle drawing of indoor experiments, ground truths are shown in black curves

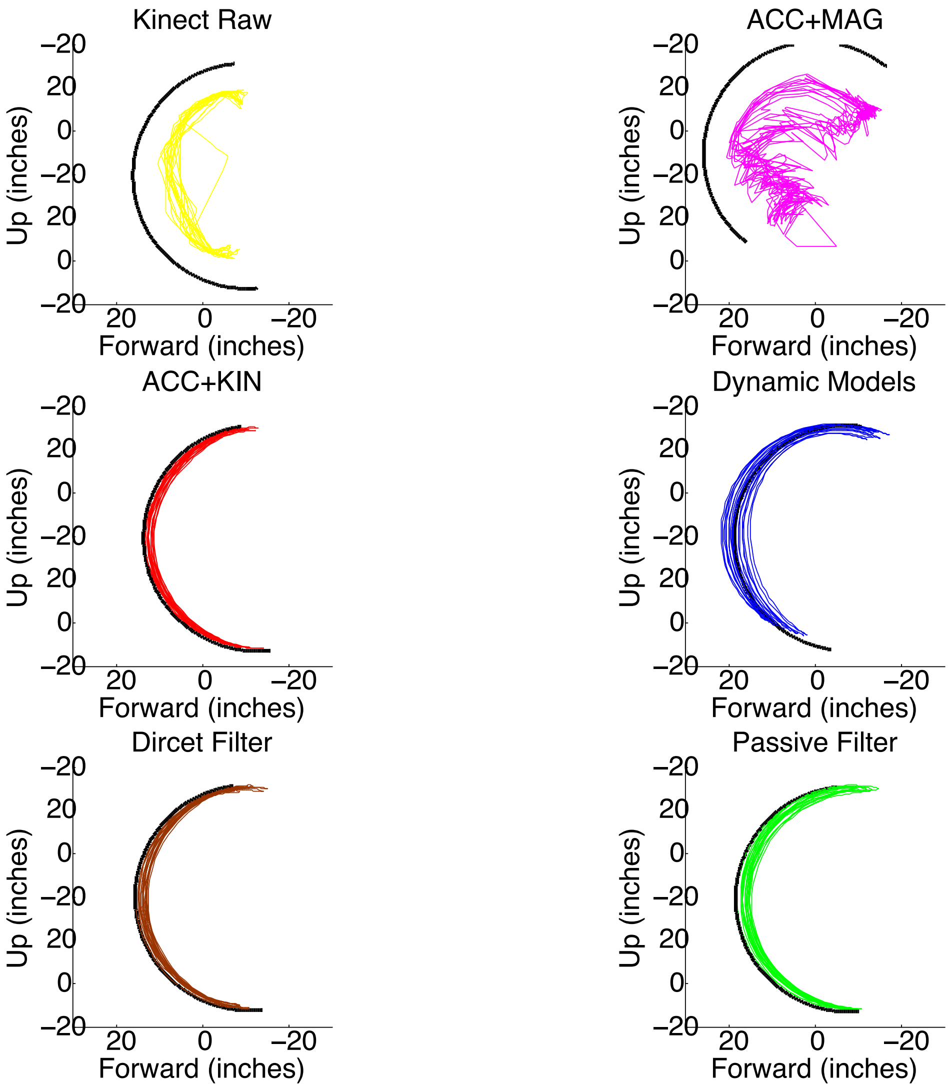


Figure 3.43. Reconstruction of vertical arm swing, of indoor experiments ground truths are shown in black curves

3.4.3 Experiment Part 3 – Coordinate Fusion

This section shows the experimental results of the coordinate fusion model described in 3.2.5 on the simulated date described in 3.3.5. In this simulation, we destroyed a portion of the Kinect data and saw how the coordinate fused model behaved in comparison to other methods.

In the following graphs in this section, we show the simulation results using best-fit shape and length estimation error evaluations described in 3.3.2.2.1 and 3.3.2.2.2 respectively. For best-fit shape error, we plot the average and variances of distances to the best-fit shape (in inches); for length-estimation error, we plot the average and variances of error percentages of length estimation. We show the simulation results using static model with magnetometer and accelerometer, using Kinect, using Kinect and accelerometer, dynamic model using gyro, and the fused result averaging passive and direct filters using IMU model in 3.2.2.2, using Kinect and IMU model in 3.2.2.3, and the proposed coordinate fusing model in 3.2.5.

For square-plot experiments, the average and variances of distances to the best-fit square are described in Figure 3.44 and Figure 3.45 respectively. The average and variances of error percentages of square side length estimation are shown in Figure 3.46 and Figure 3.47 respectively.

For triangle-plot experiments, the average and variances of distances to the best-fit triangle are described in Figure 3.48 and Figure 3.49 respectively. The average and variances of error percentages of triangle side length estimation are shown in Figure 3.50 and Figure 3.51 respectively.

For vertical arm-swing experiments, the average and variances of distances to the best-fit arc are described in Figure 3.52 and Figure 3.53 respectively. The average and variances of error percentages of swung angle estimation are shown in Figure 3.54 and Figure 3.55 respectively.

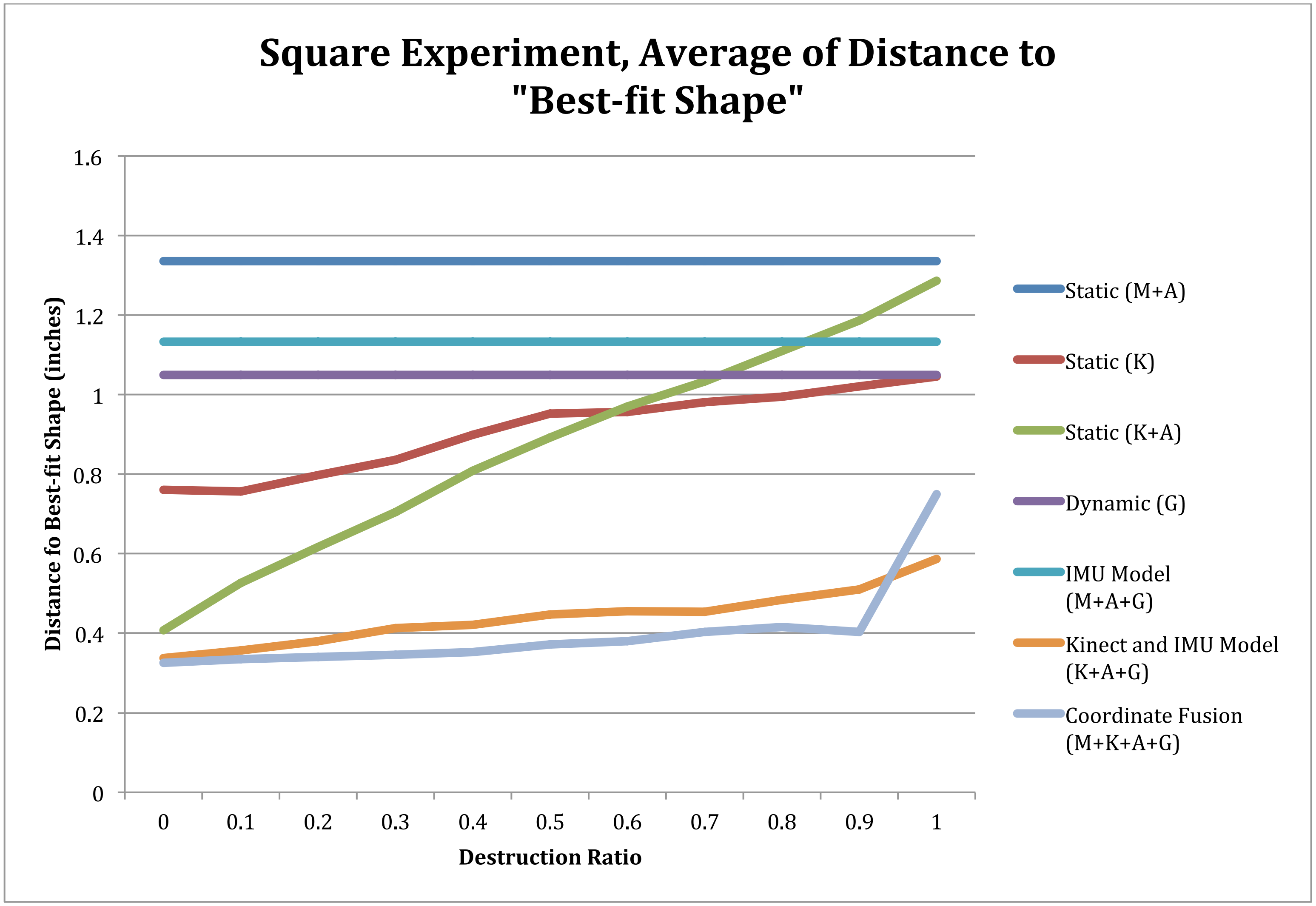


Figure 3.44. Average distance to the best-fit shape, square experiment

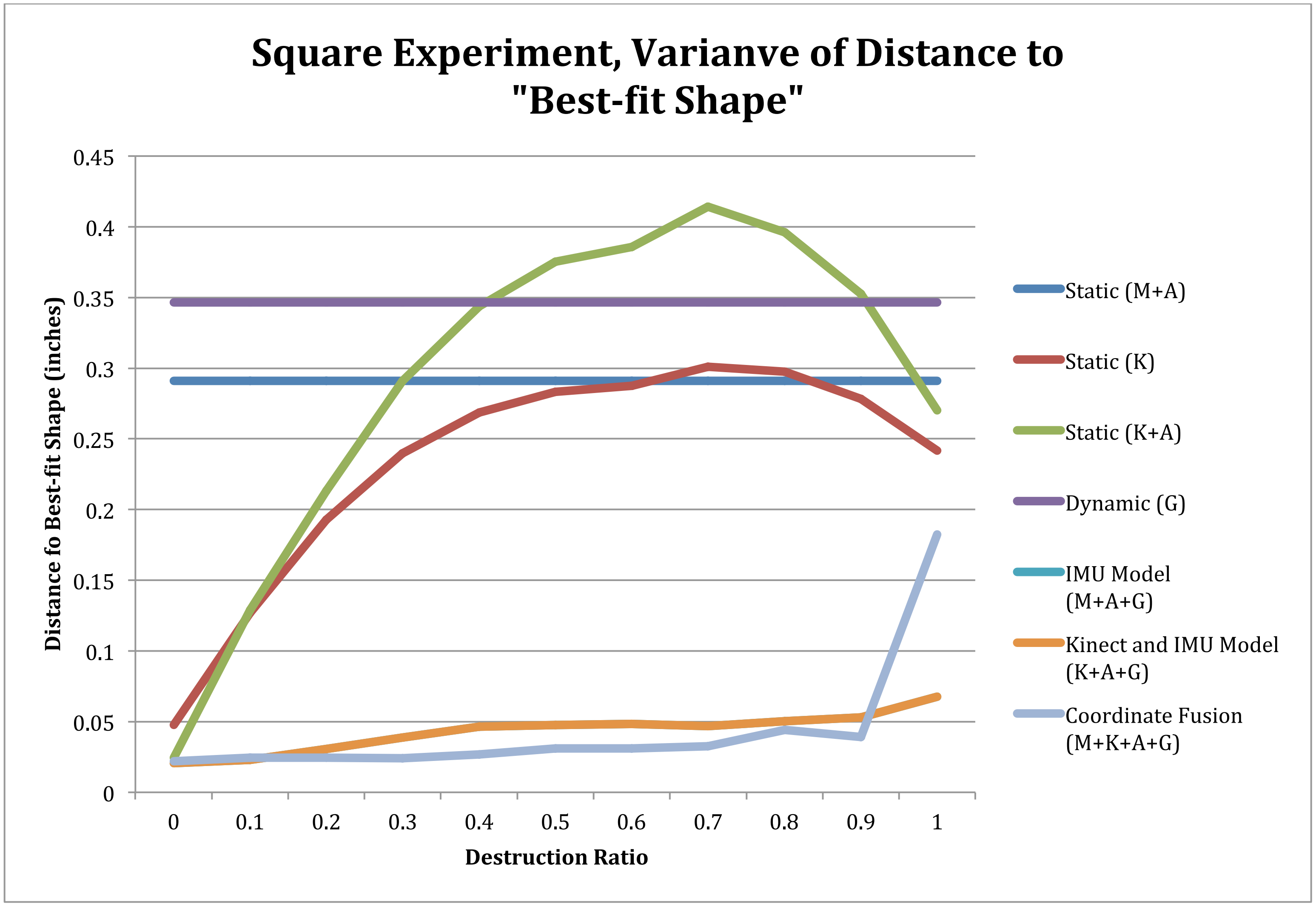


Figure 3.45. Variance of distance to the best-fit shape, square experiment

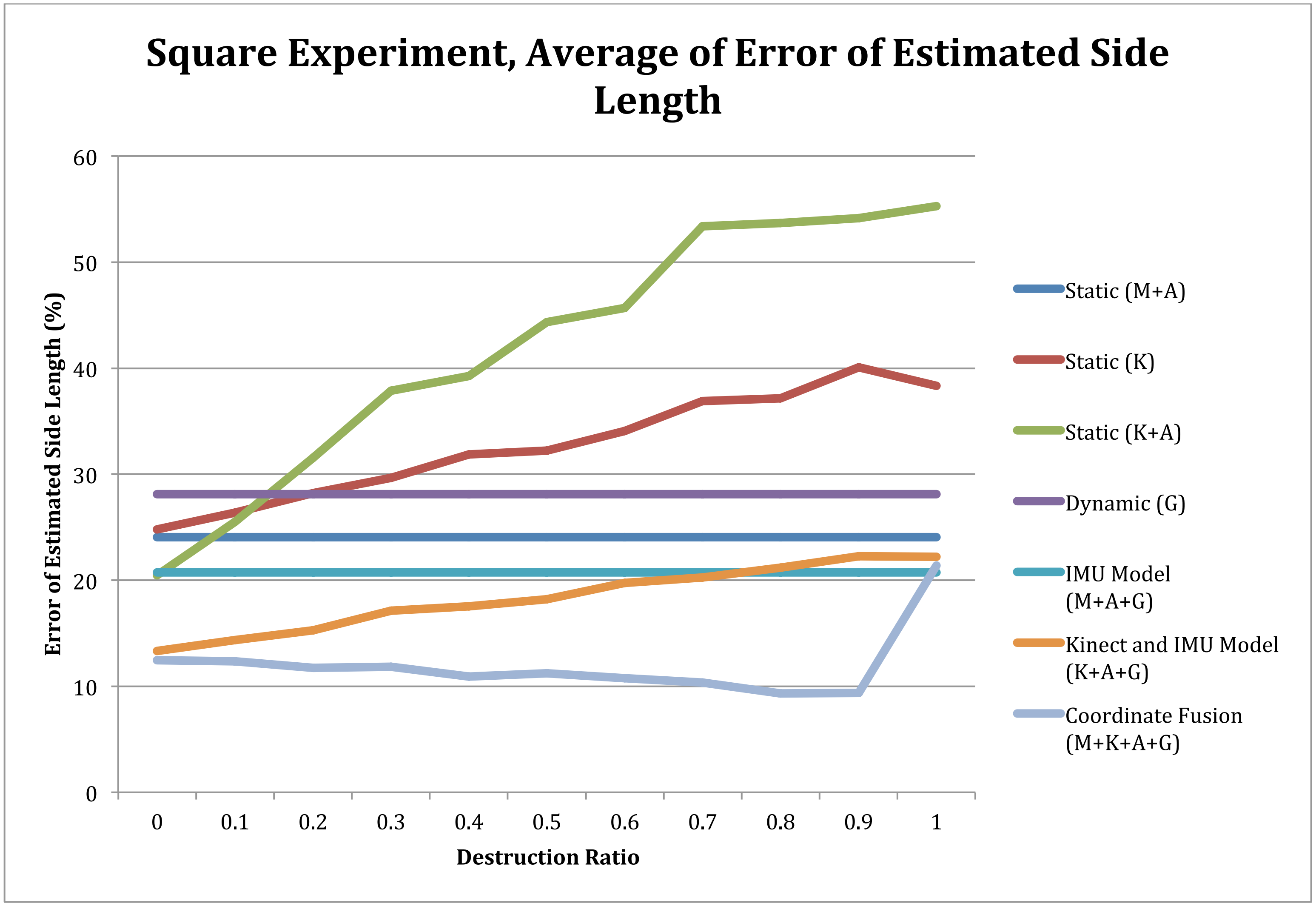


Figure 3.46. Average error of length estimation, square experiment

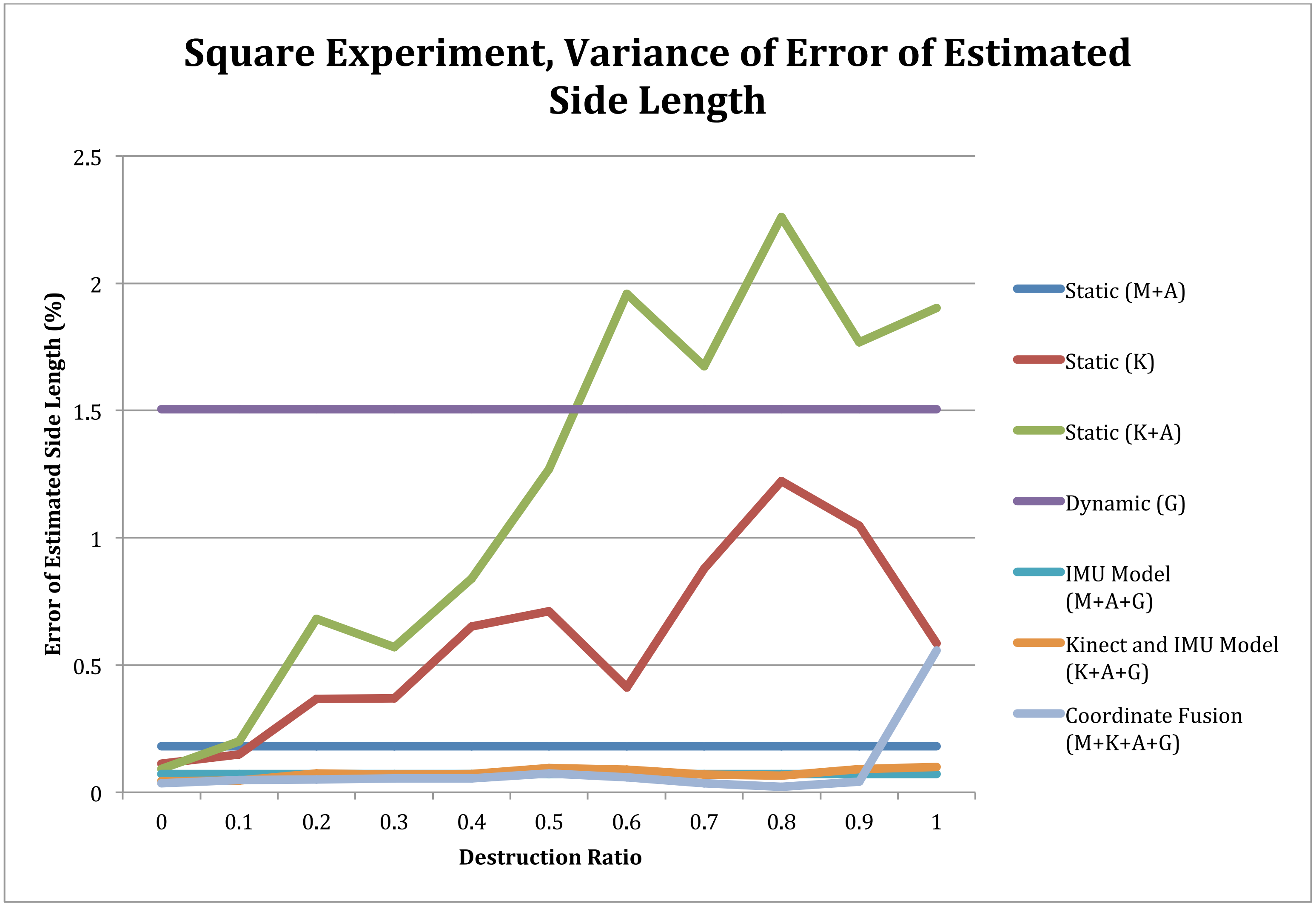


Figure 3.47. Variance of error of length estimation, square experiment

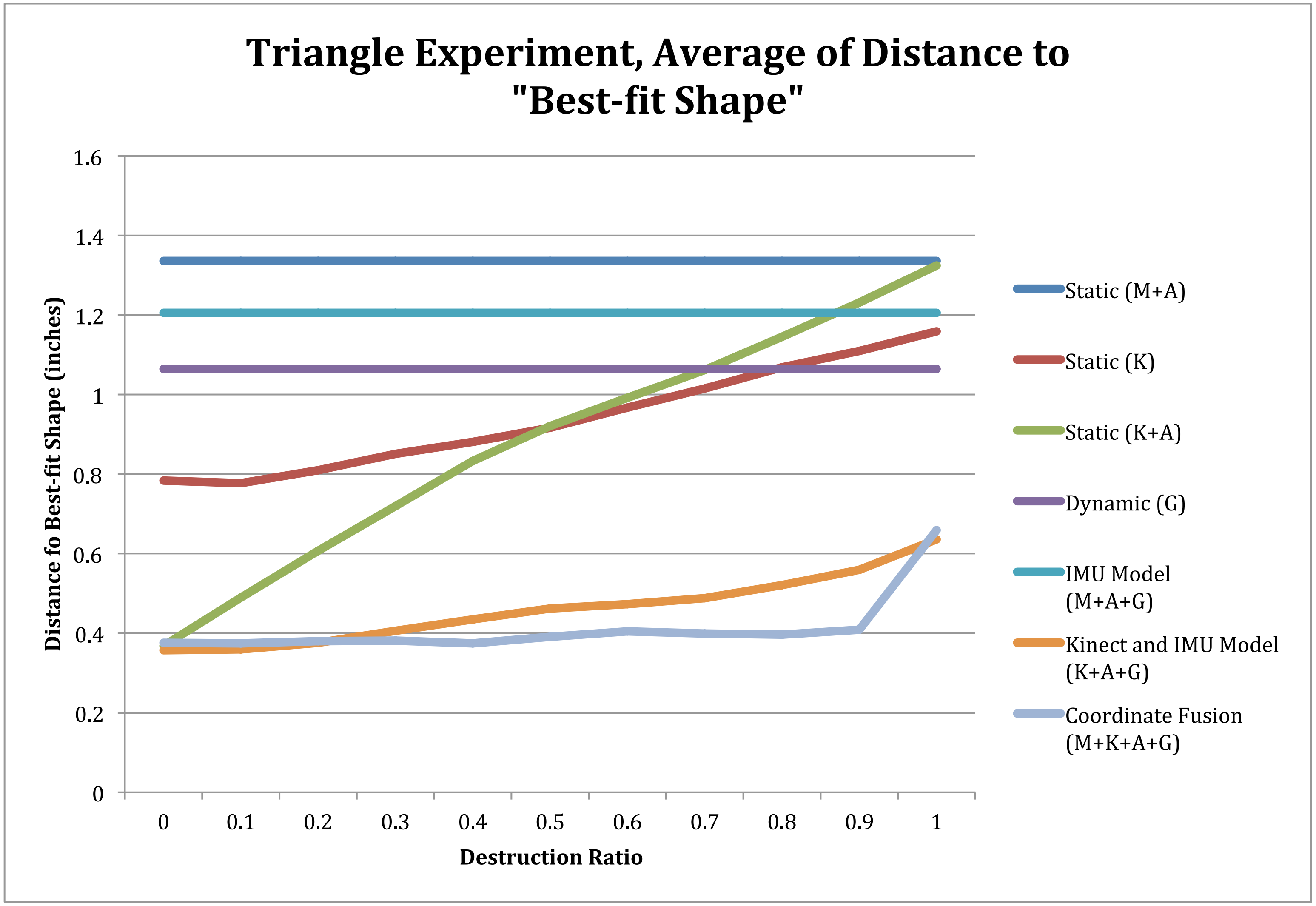


Figure 3.48. Average distance to the best-fit shape, triangle experiment

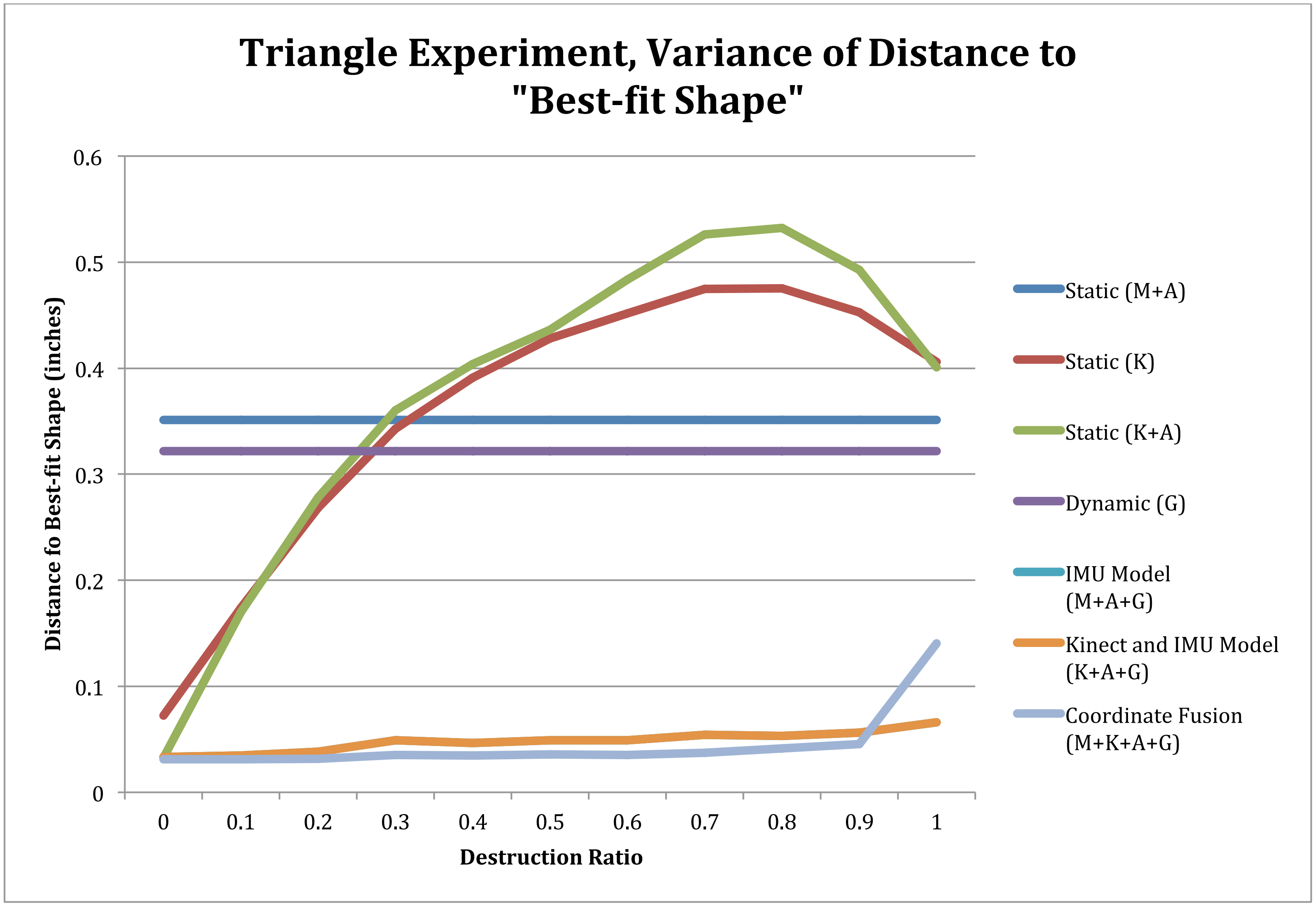


Figure 3.49. Variance of distance to the best-fit shape, square experiment

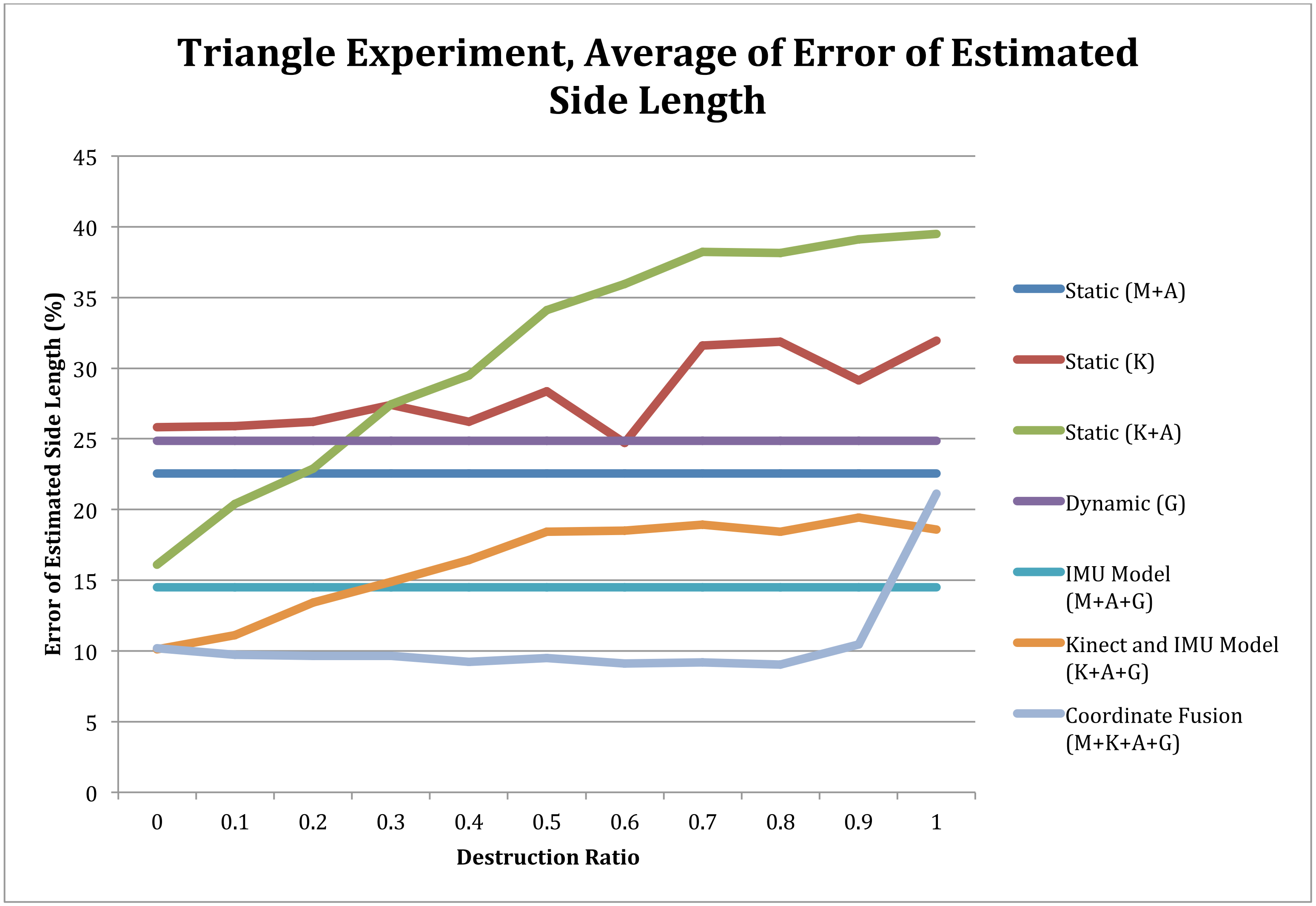


Figure 3.50. Average error of length estimation, triangle experiment

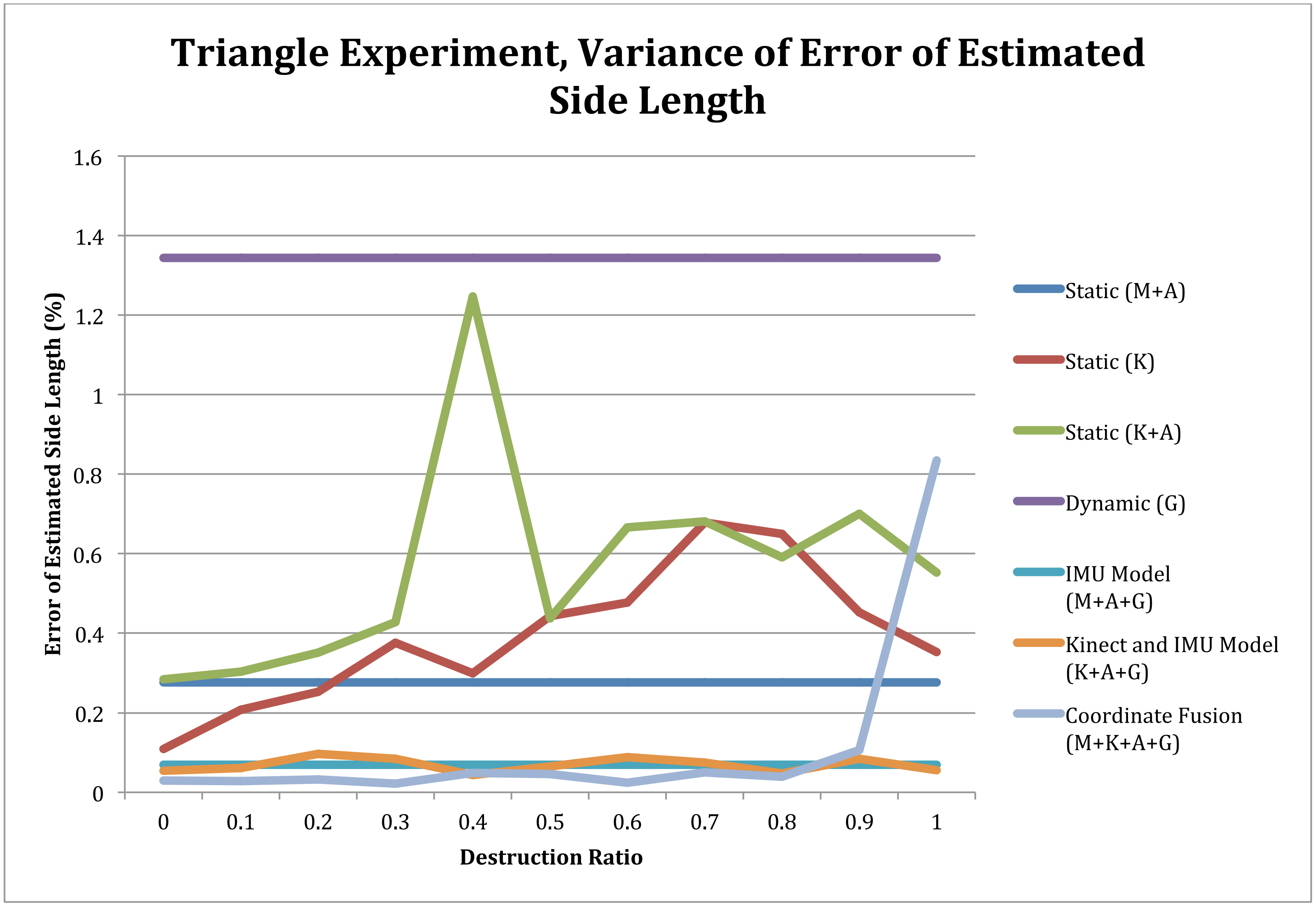


Figure 3.51. Variance of error of length estimation, triangle experiment

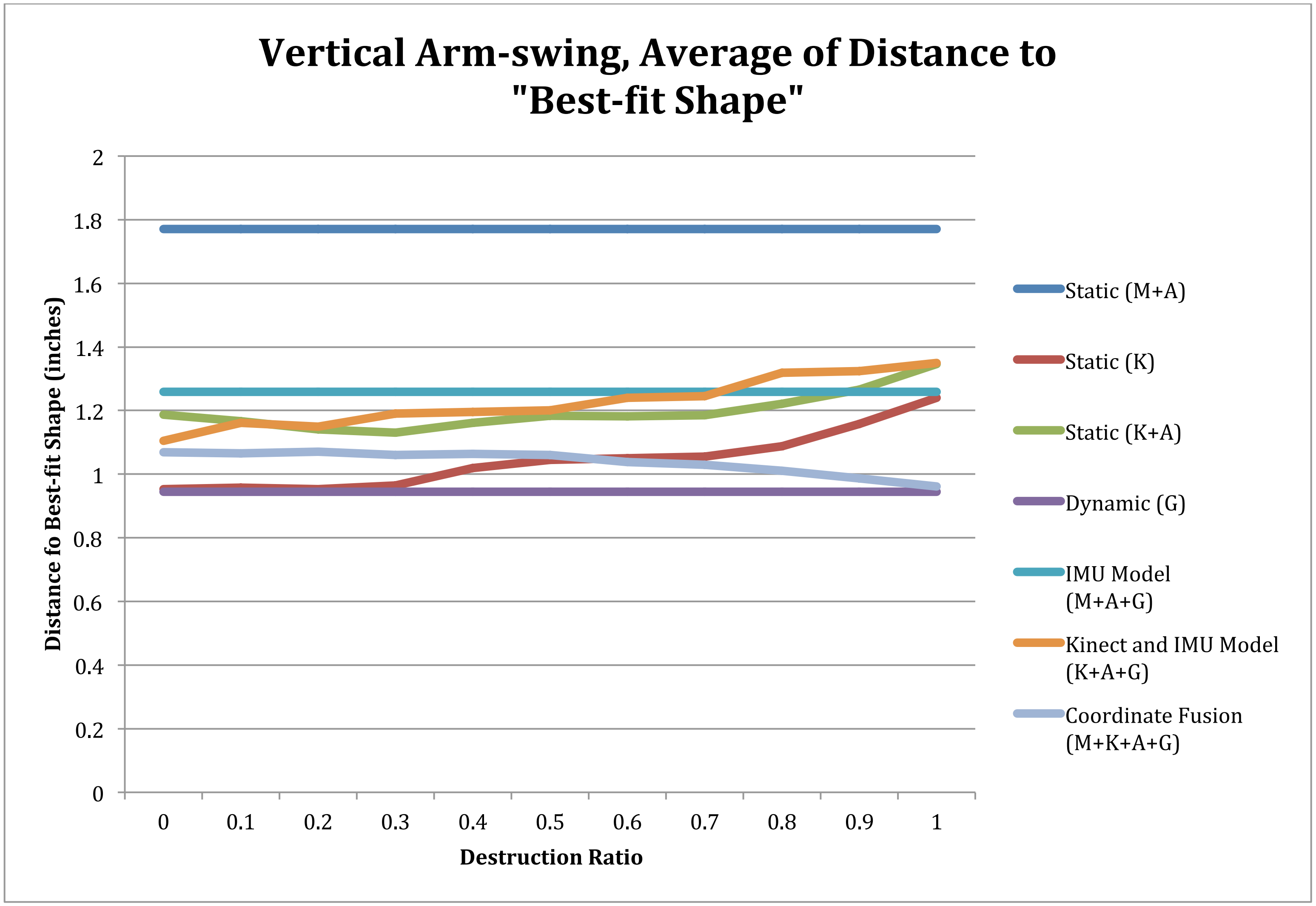


Figure 3.52. Average distance to the best-fit shape, arm-swing experiment

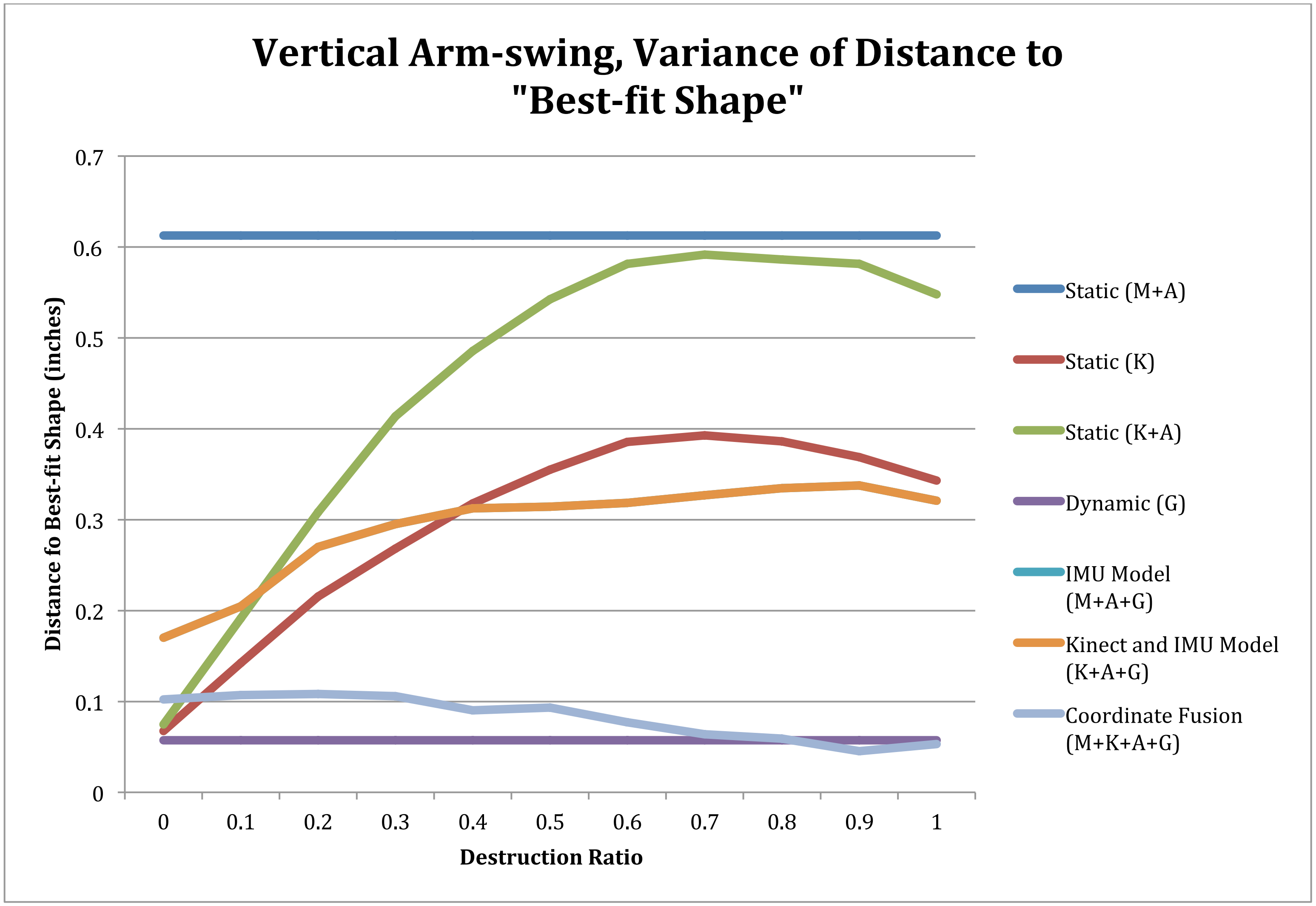


Figure 3.53. Variance of distance to the best-fit shape, arm-swing experiment

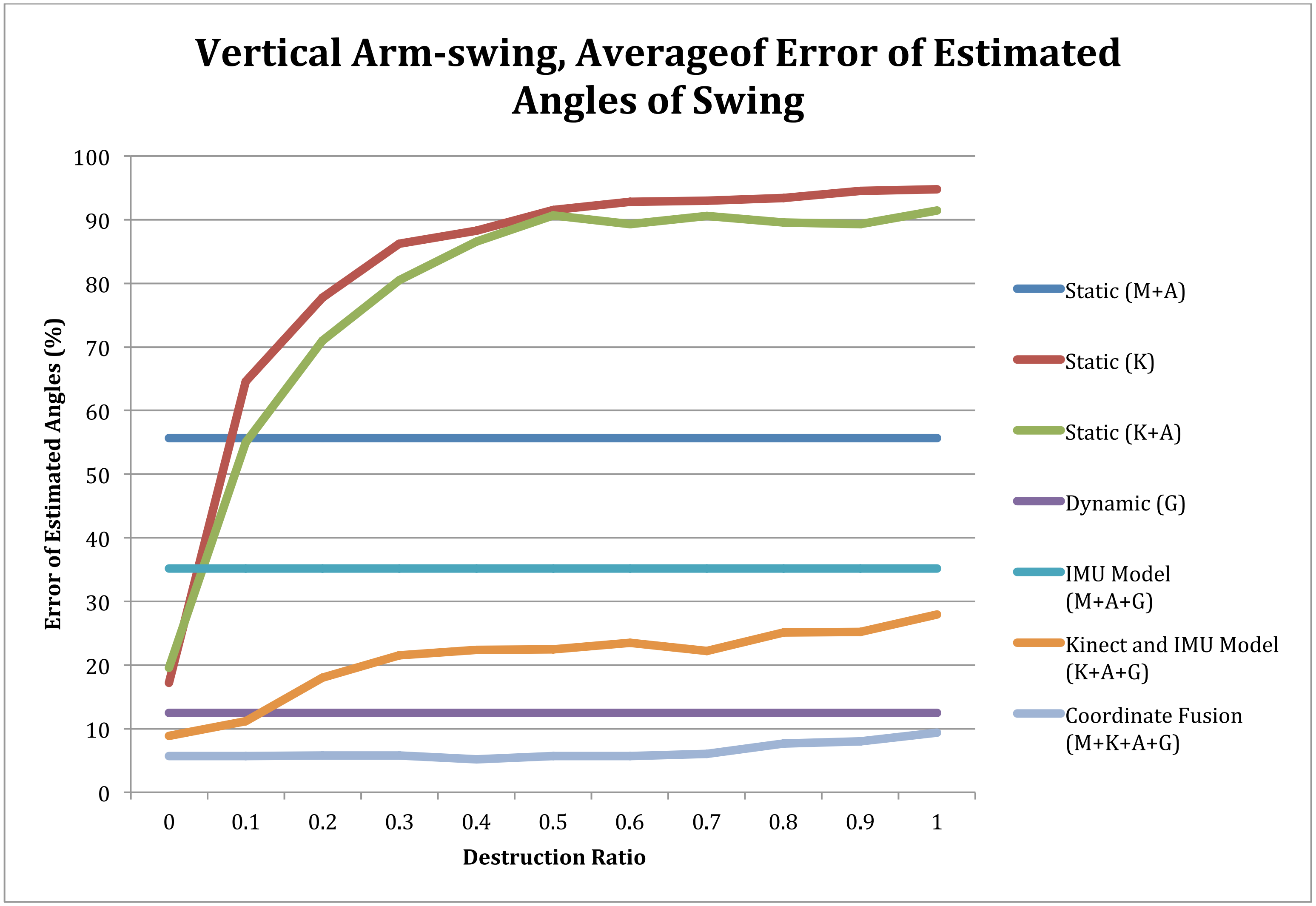


Figure 3.54. Average error of angle estimation, arm-swing experiment

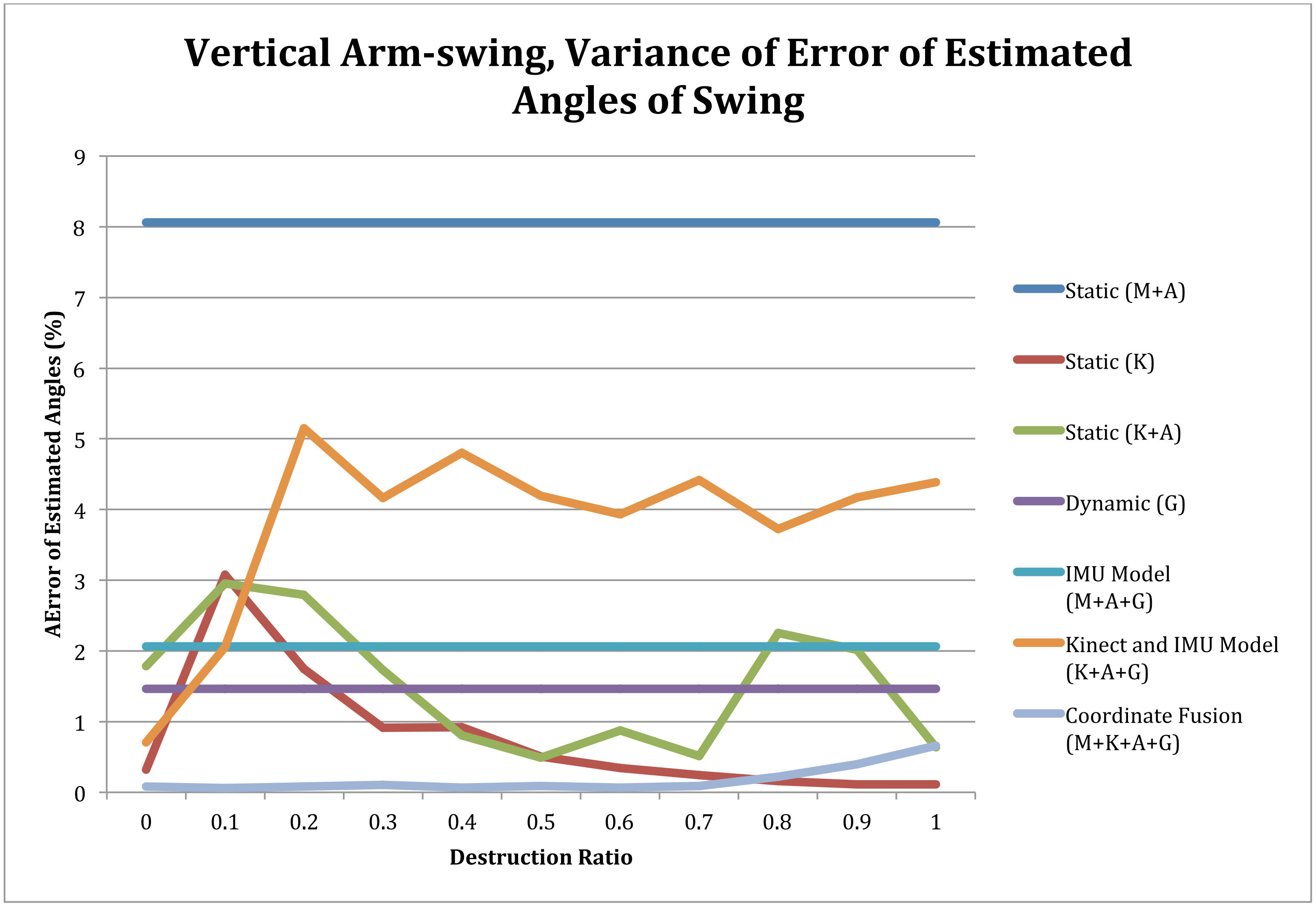


Figure 3.55. Variance of error of angle estimation, arm-swing experiment

3.5 Discussion

3.5.1 Complimentary Filters with Human Biomechanical Model using Accelerometer, Gyro and Magnetometer – Outdoor Case

From the outdoor experiment results presented in 3.4.1, we can see from Table 3.3 and Table 3.4 that only integrating the gyro signals will give us a gradually shifted shape due to error accumulation during integration. Therefore, the average distance to the best-fit shape is almost twice as large as the passive and direct complementary filtered results, and so is the variance of distances to the best-fit shape. The drifts from integration can also be seen from Figure 3.37 to Figure 3.40, where initially the first few shapes were close to their best-fit shapes, but eventually they will deviate and thus it suggests how inaccurate the dynamic model will be as time goes by. Similar results can be seen from Table 3.5 and Table 3.6, which tell us the average and variance of error percentages of length estimation. In these tables we can see that the error percentages of dynamic models ranged from twice to even four times larger than using complementary filters, and their variances were more than 20 times larger than the filtered results, indicating that even for a short period of time, using integration to estimate the lengths or distances of human motions would give us inaccurate and highly variable results.

For the static estimate using accelerometers and magnetometers, from Figure 3.37 to Figure 3.40 we can see that generally the patterns were recognizable; in the book-reaching experiments, the static estimates were very close to the ground truth. This can also be explained in Table 3.3 where the average distances to their best-fit shape were comparable to the complementary filtered results. However, from the reconstructed patterns shown in Figure 3.37, Figure 3.38 and Figure 3.40 we can see that the reconstructed results using static models with accelerometers and magnetometers were very noisy, and the reason behind this noisy reconstruction results came from first the sensor measurements themselves were very noisy, and thus made the static reconstruction without any kind of filters produce noisy results. Second, when moving we applied forces to the accelerometers, so that it made the estimation of the direction of gravity inaccurate. Third, in using magnetometers they experienced more or less interference, and with the lack of more advanced calibration devices we could not correct them.

Finally, from the passive and direct complementary filters results of Figure 3.37 to Figure 3.40 we can see that by using the non-linear complementary filters combined with the human biomechanical model, we successfully filtered the noise from static models, while avoiding integration drift from dynamic models. The drawn patterns were clearly recognizable and very close to the ground truth, with small errors, which were mainly caused from twists of subjects' wrists and deformations of forearms. From Table 3.3 and Table 3.4 of tabular errors, we can see that the proposed IMU model not only has lower error to the ground truth, **BUT** also with lower variance it means that our model was less noisy. Compared to prior works [68][77], the proposed method can track more complex and rapid arm movements.

3.5.2 Complimentary Filters with Human Biomechanical Model using Accelerometer, Gyro and Kinect – Indoor Case

After the outdoor experiments using IMU model utilizing accelerometers, gyros and magnetometers, we then moved inside and did the same experiments indoors while the Kinect was available, but severe magnetometer interference would exist.

First, from the reconstruction results with static models using accelerometers and magnetometers, which are the pink-colored lines from Figure 3.41 to Figure 3.43, we can see that because of severe interference to magnetometer measurements when indoors, the estimated traces were very inaccurate. Also, as said in 3.5.1 because of measurement noise the reconstructed results were very noisy. From Table 3.7 and Table 3.8 we can see that the average and variances of distances to the best-fit shape were the largest compared to using other models. Similar results can also be seen in Table 3.9 and Table 3.10, where we had very large error estimating lengths of drawn patterns, and angle of swung vertical arcs.

We now discuss the motion reconstruction using static models utilizing Kinect solely, and the fused model of Kinect and accelerometers. The results of only using the Kinect are shown in yellow lines, and those of using the Kinect and accelerometers are shown in red lines from Figure 3.41 to Figure 3.43. Compared to the reconstruction results using accelerometers and magnetometers depicted in the previous paragraph, we saw a great improvement while using the Kinect. However, in the reconstruction result only using Kinect's raw data, we first saw that the Kinect measurements were still noisy and its measurements were not accurate enough such that the reconstructed patterns looked smaller than the ground truth since in this model the human biomechanical constraints were not incorporated and we might have inaccurate information regarding to the lengths of limbs. On the other hand, the reconstruction results with the static model using the Kinect and accelerometers were skewed. This was mainly due to fact that the extra force that we applied to the accelerometers made the judgment of the direction of gravity incorrect. From Table 3.7 and Table 3.8 of the average and variances of distances to the best-fit shape we can see that though these distances were less accurate compared to the static model with accelerometer and magnetometer, they were still more accurate compared to the complementary filtered results. Table 3.9 and Table 3.10 of estimated side lengths of the shapes and angles of swung also suggested similar results, where the error percentages were less than static model using magnetometers and accelerometers, but larger than fused model using Kinect and IMU.

Third, we discuss the reconstruction result using purely gyro signals. This was shown in blue lines from Figure 3.41 to Figure 3.43. Similar to the integration results of the previous section, during short time intervals we avoid the measurement noise, which suggests this model is accurate in high frequency. However, as time goes by this dynamic model experienced gradually increased drift since error accumulates from numerical integration. Table 3.7 and Table 3.8 show that the average and variances of distances to the best-fit shape were much larger than the fused model results. Table 3.9 and Table 3.10 also suggested high average percentage of errors, and high variances in estimating the lengths of the drawn shape, and angles of vertical arm swing.

Finally, the filtered results using direct and passive non-linear complementary filters and human biomechanical models utilizing the Kinect, accelerometers and gyros gave us the most accurate estimation of scatterplot and arm movements. Not only did it filter out the noise from sensor measurements and avoid drifts from integration, but also the use of the Kinect indoors provided more accurate joint position information than that of magnetometer did. Therefore, from the discussion in this section and that in 3.5.1, we conclude that under different environments and resources, different strategies are required for human motion tracking and reconstruction.

3.5.3 Coordinate Fusion Model using Accelerometers, Gyros, Magnetometers and Kinect

In this section we discuss the effect of using the coordinate fusion model depicted in 3.2.5, which fuses the integration model in 3.2.2.1, the IMU model in 3.2.2.2, the Kinect and IMU model in 3.2.2.3 to deliver even better estimation results.

First we notice that from Figure 3.44 to Figure 3.55 the distance or error were unchanged for models that did not include Kinect data (static model using magnetometers and accelerometer, dynamic model using gyros, and IMU model using magnetometer, accelerometer, and gyros), since the simulated data being destroyed was the Kinect data. And for these models, the distance to the best-fit shape and the error percentage of length estimation are higher than filtered results, indicating that without the use of the Kinect we would get a very high error.

The other static models including that using the Kinect (red lines) and that using the Kinect and accelerometers (cyan-blue lines) would give us high average and variances of distances to the best-fit shapes, and estimated lengths and angles of plotted patterns. And as the destruction ratio of Kinect data increased, so did these error and distances, suggesting that if we tried to use Kinect data that were not trustworthy, the error caused by doing so would be even higher than using integration with gyro measurement.

For the integration model (purple lines), there is always high distance and error percentage of distance to the best-fit shapes and estimation lengths and angles respectively. Except for vertical arm-swing experiments, we have low average distance to the best-fit arc, and low error percentages of estimation swung angles. This is because arm-swing movements involved only rotations among one axis (in our coordinate system it was the z-axis of the sensor), and thus the errors were fewer compared to those of reconstructed plotted squares and triangles.

We also notice that the distances and error percentages of the integration model were the top limit of the proposed coordinate fusion model. This was because when we had 100% of Kinect data destroyed, unless the magnetometer was trustworthy (which usually was not true when indoors), the fused model would select the integration model as its reconstruction strategy, thus making the errors to increase to be close to those using the integration model.

When we compared the distance to the best-fit shape and error percentages of estimation lengths and angles using the IMU model in 3.2.2.2 with the Kinect and IMU model in 3.2.2.3, we found that the errors of the IMU model were higher than the later ones, even if data provided by the Kinect started getting untrustworthy. This suggested that we should use Kinect whenever we had it, to collect data and fuse Kinect data with the IMU data.

Finally, if we compared the Kinect and IMU model in 3.2.2.3 and coordinate fusion model in3.2.5, we found that unless the Kinect data were totally destroyed and totally not trustworthy, the coordinate fusion model always performed better than the Kinect and IMU model. Compared to other methods, the proposed coordinate fusion model always gave the lowest average and variances distance to the best-fit shape and those of length and angle estimation. As described in the previous paragraph, the gyro based integration models served as the lower limit of the reconstruction accuracy.

3.6 Conclusion

To conclude, in this chapter we present a coordinate fusion system that tracks and reconstructs human motions using commercial MEMS and other sensors off the shelf. The system combines several existing motion reconstruction methods, thus making it more adaptable to various environments with different sensors available. Further, the proposed system is expandable when new type of sensors and motion reconstruction methods become available. This system will not only benefit medical professional and therapists who want to analyze more detail in human motion trajectories, but this system can also be included in the medical remote monitoring system for many medical purposes, e.g., a medical surveillance system which keeps track of patients requiring long-term care and reports to emergency if necessary, or a system to see if the patients in rehabilitation have followed doctors' directions to exercise for a prescribed amount of time daily.

Chapter 4

CONCLUSIONS AND FUTURE RESEARCH

4.1 Research Contribution

This research mainly concerns machine learning on activity classification and motion tracking / reconstruction using various off-the-shelf sensors and digital filtering techniques. That is, this research tries to construct a system, which collects data from sensors attached to human bodies, and then makes inferences from collected data. The system either tries to tell which activities the subject is doing at any given moment (activity classification in Chapter 2), or to show detailed movements of human actions (motion tracking / reconstruction in Chapter 3).

The research is a building block for end-to-end wireless health systems that target either patients requiring long-term care or the general public. For patients requiring long-term care, such as people in the rehabilitation process or people with chronic diseases, traditional treatments require them to stay at hospitals each day. The system will serve as a remote monitoring system that can record and store patients' activities in their daily lives while patients stay at home, then doctors can access the data and make diagnostics remotely, which will save tremendous medical costs and resources.

For the general public, this system will monitor people's health status day by day and make suggestions based on their health level. As mentioned in Chapter 1, according to WHO more than 60 percent of populations fails to exercise 30 minutes per day, which is the least requirement for fitness. Therefore, if there is a system that can accurately classify human activities, and then compute health inferences, statistics, and suggestions from the data, all at low cost, then it can greatly improve our health levels.

4.2 Future Research

This section points out some potential research directions in the future. On one hand, we wish to develop a larger system of activity classification / motion reconstruction with various levels, where the current research contributes to two such levels. On the other hand, in order to acquire the more accurate ground truth, another project named Virtual Sensor Platform in cooperation with UCLA Rehabilitation Unit of the Ronald Reagan Hospital to collect data using a multi-camera motion capture system will also be launched.

4.2.1 Multilevel System Optimization

As mentioned in 4.1, we expect an end-to-end wireless health system that will have better performance for targeted applications. To achieve this, in the future it is desirable to expand the current research to create a broader system operating at multiple levels of abstraction. In this dissertation we have already proposed 2 systems. For activity classification, the use of hybrid tree classifiers in 2.4 combines decision tree classifiers with naïve Bayes classifiers and support vector machines to adapt to many more kinds of complicated activities; the idea of coordinate fusion in 3.2.5 weights different motion reconstruction methods to perform better motion tracking under various environments and circumstances. In the future, these two systems will become levels of an even larger system; the levels of this system will communicate with each other to optimize the performance as a whole.

One instance of such levels can be the concept of determining and using contexts. Contexts identify the surrounding environments, and hence suggest possible constraints and available resources. For example, if from the Wi-Fi connection the system knows that the subject stays at the office, then chances are this subject would not do any intense exercises, thus we can lower the prior of corresponding classes in the naïve Bayes classifier. Another context example is that by knowing the phone is plugged into a power outlet, the system can increase the sampling rate to achieve better classification accuracy. Therefore, by knowing the contexts we can have a better activity classification / motion reconstruction result.

What is more, the research of context detection is not a one-way thing as the context will help with increasing activity classification / motion reconstruction accuracy. The current systems of activity classification / motion reconstruction can also improve context detection. For instance, if by using the universal hybrid tree classifier we know that the subject was sitting for a long time in the afternoon, and then we can suspect that he/she was in the office during that time. Therefore, not only do we need to find the new levels in the larger system, but also how the levels interact with each other.

4.2.2 Virtual Sensor Platform and Vicon Motion Capture System

The idea of the Virtual Sensor is to construct a platform that simulates outputs of MEMS sensors including accelerometers, gyros and magnetometers as if they were put on the subject's body parts while he/she is doing various activities. This platform will provide a simulated ground truth, that is, the sensor output should be given as if the subject were doing prescribed motions. This is exactly the reverse of what this dissertation tries to achieve – to reconstruct the subject's motions given the sensor output. Therefore, with this platform, not only can it provide an approximate signal template for a given motion, but through the bootstrap process, of which this platform and the proposed motion reconstruction system help each other, we can improve the accuracies of both systems.

Another accurate ground truth will also be acquired using an eight-camera Motion Capture System here at the University of California, Los Angeles (UCLA) [86]. This is a high-speed motion capture system that can capture locations of special markers attached on the human body using cameras. With the aid of this system, we can accurately record human motions and thus provide better and more detailed motion trajectories. In the future, with the cooperation of the UCLA Rehabilitation Unit of the Ronald Reagan Hospital and by accessing this system not only can we accurately record human motions and thus provide better and more detailed motion trajectories, but also we will discover more potential ways of realizing the system of activity classification and motion reconstruction for real-life medical applications.

REFERENCES

[1] L. Atallah, B. Lo, R. King, and G.-Z. Yang, “Sensor placement for activity detection using wearable accelerometers,” in *Proc. Wearable Implantable Body Sens. Netw.*, Los Alamitos, CA, 2010, pp. Int.Workshop, pp. 24–29.

[2] Long, X., Yin, B., and Aarts, R.M. 2009. Single accelerometer-based daily physical activity classification. In *31st Annual International Conference of the IEEE EMBS*, 6107-6110.

[3] J. Pärkkä, M. Ermes, P. Korpipää, J. M¨antyjärvi, J. Peltola, and I. Korhonen, “Activity classification using realistic data from wearable sensors,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 10, no. 1, pp. 119–128, Jan. 2006.

[4] L. Bao and S. S. Intille, “Activity recognition from user-annotated acceleration data,” in *Proceedings 2nd Int. Conference on. Pervasive Computing*. Springer, 2004, pp. 1-17.

[5] Renk, Erin L., et al. "Calibrating a triaxial accelerometer-magnetometer-using robotic actuation for sensor reorientation during data collection." *Control Systems*, IEEE 25.6 (2005): 86-95.

[6] Bellman, Richard. "Dynamic Programming and Lagrange Multipliers." *The Bellman Continuum: A Collection of the Works of Richard E. Bellman (1986): 49*.

[7] Bellman, Richard. *Adaptive control processes: a guided tour.* Vol. 4. Princeton: Princeton university press, 1961.

[8] Oommen, Thomas, et al. "An objective analysis of support vector machine based classification for remote sensing." *Mathematical Geosciences* 40.4 (2008): 409-424.

[9] F. Foerster and J. Fahrenberg, "Motion pattern and posture: Correctly assessed by calibrated accelerometers," *Behav. Res. Methods Instrum. Comput.*, vol. 32, no. 3, pp. 450–457, Aug. 2000.

[10] AS.R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology." *IEEE Trans. Systems Man Cybernet*. 21 (1991), pp. 660–674.

[11] M. W. Kurzynski, "The optimal strategy of a tree classifier," *Pattern Recognition* vol. 16,. 81-87 (1983).

[12] Breiman, L., J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Boca Raton, FL: CRC Press, (1984).

[13] De'Ath, G. & Fabricius, K.E. (2000) Classification and regression trees: a powerful yet simple technique for ecological data analysis. *Ecology*, 81, 3178–3192.

[14] D. Lowd, P. Domingos, "Naïve Bayes models for probability estimation," in Proceedings of 22nd *International Conference on Machine Learning*, Bonn, Germany, 2005.

[15] K. P. Murphy, "Naïve Bayes classifiers," http://www.cs.ubc.ca/~murphyk/Teaching/CS340-Fall06/reading/NB.pdf

[16] Harry Zhang "The Optimality of Naïve Bayes". *FLAIRS2004 conference*.

[17] Bottou, L., and Chih-Jen Lin. *Support Vector Machine Solvers*. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.4209&rep=rep1&type=pdf

[18] Christianini, N., and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, Cambridge, UK, 2000.

[19] Hastie, T., R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*, second edition. Springer, New York, 2008.

[20] Higgins, W. T. "A comparison of complementary and Kalman filtering." *Aerospace and Electronic Systems*, IEEE Transactions on 3 (1975): 321-325.

[21] S. S.Osder, W. e. Rouse, and L. S. Young, “Navigation, guidance and control systems for V/STOL aircraft,” *Sperry Tech*. col. 1, no. 3, 1973

[22] Marins, João Luís, et al. "An extended Kalman filter for quaternion-based orientation estimation using MARG sensors." *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*. Vol. 4. IEEE, 2001.

[23] Webb, Jarrett, and James Ashley. *Beginning Kinect Programming with the Microsoft Kinect SDK*. Apress, 2012.

[24] Waldner, Jean-Baptsite. *Nanocomputers and swarm intelligence*. Vol. 12. Wiley. com, 2010.

[25] Bernstein, Jonathan. "An Overview of MEMS Inertial Sensing Technology", Sensors Weekly, February 1, 2003.

[26] Elmenreich, W. (2002). *Sensor Fusion in Time-Triggered Systems*, PhD Thesis. Vienna, Austria: Vienna University of Technology. p. 173.

[27] Haghighat, M. B. A., Aghagolzadeh, A., & Seyedarabi, H. (2011). *Multi-focus image fusion for visual sensor networks in DCT domain*. Computers & Electrical Engineering, 37(5), 789-797.

[28] Shala, Ubejd, and Angel Rodriguez. *Indoor positioning using sensor-fusion in android devices*. Diss. Kristianstad University, 2011.

[29] Grewal, M. S.; L. R. Weill, and A. P. Andrew (2007). *Global Positioning, Inertial Navigation & Integration*. New York: John Wiley & Sons.

[30] Lai, Kam, Janusz Konrad, and Prakash Ishwar. "A gesture-driven computer interface using Kinect." *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on. IEEE*, 2012.

[31] Oikonomidis, Iason, Nikolaos Kyriazis, and Antonis A. Argyros. "Efficient model-based 3D tracking of hand articulations using Kinect." *BMVC*. 2011.

[32] Khoshelham, Kourosh, and Sander Oude Elberink. "Accuracy and resolution of kinect depth data for indoor mapping applications." *Sensors* 12.2 (2012): 1437-1454.

[33] Xia, Lu, Chia-Chih Chen, and J. K. Aggarwal. "Human detection using depth information by Kinect." *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011.

[34] Chang, Yao-Jen, Shu-Fang Chen, and Jun-Da Huang. "A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities." *Research in developmental disabilities* 32.6 (2011): 2566-2570.

[35] Gallo, Luigi, Alessio Pierluigi Placitelli, and Mario Ciampi. "Controller-free exploration of medical image data: Experiencing the Kinect." *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*. IEEE, 2011.

[36] G. Landerweerd, T. Timmers, E. Gelsema, M. Bins and M. Halic, "Binary tree versus single level tree classification of while blood cells," *Pattern Recognition* vol. 16, 571-577 (1983).

[37] X. Li and R. C. Dubes, "Tree classifier design with a Permutation statistic," *Pattern Recognition* vol. 19, 229-235 (1986).

[38] http://www.gcdataconcepts.com/

[39] Cortes and V. Vapnik. "Support vector networks." *Machine Learning*, 20:273 – 297, 1995.

[40] Jonathan Lester, Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, James L, Kate Everitt, and Ian Smith, “Sensing and modeling activities to support physical fitness,”In *Proceedings of UbiComp*, 2005.

[41] H.Z. Tan, L.A. Slivovsky, and A. Pentland, “A sensing chair using pressure distribution sensors,” *IEEE/ASME Transactions On Mechatronics*, vol. 6, no. 3, pp. 261–268, 2001.

[42] Mota, Selene, and Rosalind W. Picard. "Automated posture analysis for detecting learner's interest level." *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*. Vol. 5. IEEE, 2003.

[43] R. Cucchiara, C. Grana, A. Prati, and R. Vezzani, “Probabilistic posture classification for human-behavior analysis,” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 35, no. 1, pp. 42–54, 2005.

[44] G.S. Bruss, A.M. Gruenberg, R.D. Goldstein, and J.P. Barber, “Hamilton Anxiety Rating Scale Interview guide: joint interview and test-retest methods for interrater reliability,” *Psychiatry research*, vol. 53, no. 2, pp. 191–202, 1994.

[45] P.A. Reilly, “Fibromyalgia in the workplace: a ’management’ problem*,” Annals of the rheumatic diseases*, vol. 52, no. 4, pp. 249, 1993.

[46] D. Blazer, D. Hughes, L.K. George, M. Swartz, and R. Boyer, “8 Generalized Anxiety Disorder,” *Psychiatric disorders in America: The epidemiologic catchment area study*, p. 180, 1991.

[47] David Lewis, “Naïve (bayes) at forty: The independence assumption in information retrieval,” in *Machine Learning: ECML-98, Claire Ndellec and Cline Rouveirol, Eds*., vol. 1398 of Lecture Notes in Computer Science, pp. 4–15. Springer Berlin / Heidelberg, 1998.

[48] Occupational Safety & Health Administration U.S. Department of Labor, “Good working positions,” http://www.osha.gov/SLTC/etools/computerworkstations/positions.html

[49] C. Glaros, D.I. Fotiadis, A. Likas, A. Stafylopatis, ”A Wearable Intelligent System for Monitoring Health Condition and Rehabilitation of Running Athletes,” In *the 4th International EMBS Special Topic Conference on Information Technology Applications in Biomedicine*, April 2003

[50] Youngbum Lee and Myoungho Lee, ”Implementation of Accelerometer Sensor Module and Fall Detection Monitoring System based on Wireless Sensor Network” In *Proceedings of the 29th Annual International Conference of the IEEE EMBS*, August 2007.

[51] Q. Yuan, I-M. Chen, S. P. Lee, ”SLAC: 3D Localization of Human Based on Kinetic Human Movement Capture,” *IEEE International Conference on Robotics and Automation*, Shanghai International Conference Center, May 2011.

[52] J. M. McCarthy, ”An Introduction to Theoretical Kinematics.” *MIT Press*, 1990.

[53] Gulf Coast Data Concepts: http://www.gcdataconcepts.com

[54] DK White, RC Wagenaar, ME Del Olmo, and TD Ellis, “Test-retest reliability of 24 hours of activity monitoring in individuals with parkinsons disease in home and community,” *Neurorehabil Neural Repair*, 2007.

[55] F. M. Impellizzeri, S. M. Marcora, C. Castagna, T. Reilly, A. Sassi, F. M. Iaia, and E. Rampinini, “Physiological and performance effects of generic versus specific aerobic training in soccer players,” *International Journal of Sports Medicine*, 2006.

[56] N. Ruchansky, C. Lochner, E. Do, T. Rawls, N. Hajj Chehade, J. Chien, G. Pottie, and W Kaiser, “Monitoring workspace activities using accelerometers,” *ICASSP*, 2011.

[57] Zhongtang Zhao, Yiqiang Chen, Junfa Liu, Zhiqi Shen, and Mingjie Liu, “Cross-people mobile-phone based activity recognition,” 2011.

[58] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello, “A practical approach to recognizing physical activities,” pp. 1–16, 2006.

[59] Jhun-Ying Yang, Jeen-ShingWang, and Yen-Ping Chen, “Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers,” *Pattern Recognition Letters*, vol. 29, no. 16, pp. 2213 – 2220, 2008.

[60] Kern, Nicky, Bernt Schiele, and Albrecht Schmidt. "Multi-sensor activity context detection for wearable computing." *Ambient Intelligence*. Springer Berlin Heidelberg, 2003. 220-232.

[61] Aminian, K., et al. "Physical activity monitoring based on accelerometry: validation and comparison with video observation." *Medical & biological engineering & computing* 37.3 (1999): 304-308.

[62] Kenji Kira and Larry A. Rendell, “A practical approach to feature selection,” pp. 249–256, 1992.

[63] R. Battiti, “Using mutual information for selecting features in supervised neural net learning,” *Neural Networks*, IEEE Transactions on, vol. 5, no. 4, pp. 537–550, jul 1994.

[64] S. Roberts and R. Everson, Independent component analysis: *principles and practice*, Cambridge University Press, 2001.

[65] T. M. Cover and J. A. Thomas, Elements of information theory, Wiley Series in *Telecommunications and Signal Processing*. Wiley-Interscience, 2006.

[66] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of maxdependency, max-relevance, and min-redundancy,” *Pattern Analysis and Machine* Intelligence, IEEE Transactions on, vol. 27, no. 8, pp. 1226 –1238, aug. 2005.

[67] M. F. Huber, T. Bailey, H. Durrant-Whyte, and U. D. Hanebeck, "On entropy approximation for gaussian mixture random vectors," in *Multisensor Fusion and Integration for Intelligent Systems, 2008. MFI 2008. IEEE International Conference* on, aug. 2008, pp. 181 –188.

[68] S. Bertrand, T. Hamel, H. Piet-Lahanier, and R. Mahony, "Attitude tracking of rigid bodies on the special orthogonal group with bounded partial state feedback," in *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, pp. 2972–2977, December 2009.

[69] S. P. Tseng, W.-L. Li, C.-Y. Sheng, J.-W. Hsu, and C.- S. Chen, "Motion and attitude estimation using inertial measurements with complementary filter," in *Proceedings of 2011 8th Asian Control Conference (ASCC),* pp. 863–868, May 2011.

[70] R. Mahony, T. Hamel, and J.-M. Pimlin, "Non-linear complementary filters on the special orthogonal group," *IEEE Transactions on Automatic Control, vol. 53,* pp. 1203–1217, May 2008.

[71] D. Roetenberg, *Inertial and Magnetic Sensing of Human Motion*. PhD thesis, Universiteit Twente, 2006.

[72] C. Chien and G. J. Pottie, "A universal hybrid decision tree classifier design for human activity classification," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, August 2012.

[73] N. H. Chehade, A. Friedman, , C. Chien, and G. J. Pottie, "Estimation of accelerometer orientation for activity recognition," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, August 2012.

[74] B. Fish, A. Khan, N. H. Chehade, C. Chien, and G. J. Pottie, “Feature selection based on mutual information for human activity recognition,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 2012.

[75] Y. Wang, X. Xu, M. Batalin, and W. Kaiser, “Detection of upper limb activities using multimode sensor fusion,” in *Biomedical Circuits and Systems Conference (BioCAS), 2011 IEEE*, November 2011.

[76] M. El-Gohary, L. Holmstrom, J. Huisinga, E. King, J. McNames, and F. Horak, “Upper limb joint angle tracking with inertial sensors,” in *33rd Annual International Conference of the IEEE EMBS*, 2011.

[77] S. Suvorova,T. Vaithianathan, and T. Caelli, “Action trajectory reconstruction from inertial sensor,” in The *11th International Conference on Information Science, Signal Processing and their applications*, July 2012.

[78] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, “A high integrity imu/gps navigation loop for autonomous land vehicle applications,” *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 572–578, June 1999.

[79] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, “Circumventing dynamic modeling: Evaluation of the error-state Kalman filter applied to mobile robot localization,” in *International Conference on Robotics and Automation*, pp. 1656–1663, May 1999.

[80] N. El-Sheimy, H. Hou, and X. Niu, “Analysis and modeling of inertial sensors using Allan variance,” *IEEE Transactions on Instrumentation and Measurement*, vol. 57, pp. 140–149, January 2008.

[81] C. Chien, J. Xia, O. Santana, Y. Wang, Greg J. Pottie, “Non-linear complementary filter based upper limb motion tracking using wearable sensors” in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2013.

[82] R. Mahony and T. Hamel, “Attitude estimation on SO(3) based on direct inertial measurements,” in *International Conference on Robotics and Automation,* ICRA2006, 2006.

[83] R. Mahony, T. Hamel, and J.-M. Pimlin, “Complementary filter design on the special orthogonal group SO(3),” in *44th IEEE Conference on Decision and Control, and the European Control Conference 2005*, pp. 1477–1484, December 2005.

[84] Pratt, Vaughan. "Direct least-squares fitting of algebraic surfaces." *ACM SIGGRAPH Computer Graphics*. Vol. 21. No. 4. ACM, 1987.

[85] Sparkfun Electronics, “9 degrees of freedom - razor imu.”

[86] Vicon 8 motion capture system, http://www.vicon.com/System/Cara