# A Novel Accelerometer-based Gesture Recognition System

by

Ahmad Akl

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

# Abstract

A Novel Accelerometer-based Gesture Recognition System

Ahmad Akl

Master of Applied Science

Graduate Department of Electrical and Computer Engineering

University of Toronto

2010

Gesture Recognition provides an efficient human-computer interaction for interactive and intelligent computing. In this work, we address the problem of gesture recognition using the theory of random projection and by formulating the recognition problem as an $\ell_1$-minimization problem. The gesture recognition uses a single 3-axis accelerometer for data acquisition and comprises two main stages: a training stage and a testing stage. For training, the system employs dynamic time warping as well as affinity propagation to create exemplars for each gesture while for testing, the system projects all candidate traces and also the unknown trace onto the same lower dimensional subspace for recognition. A dictionary of 18 gestures is defined and a database of over 3,700 traces is created from 7 subjects on which the system is tested and evaluated. Simulation results reveal a superior performance, in terms of accuracy and computational complexity, compared to other systems in the literature.

# Dedication

To my sister Zeina

# Acknowledgements

I would like to thank all the people who have helped and inspired me during the pursuit of my masters degree.

First, I would like to thank my supervisor, Prof. Shahrokh Valaee, for his guidance during my research and study at the University of Toronto. Prof. Valaee was always accessible and willing to help with my research making my research experience a smooth and a rewarding one.

I also would like to thank all my lab mates at Wirlab (Wireless and Internet Research Laboratory) for making the lab a friendly place to work. I, especially, would like to thank Veria Nassab and Le Zhang for their incredible help and assistance. If it were not for them, I don't think that I would have been able to make it through.

My deepest gratitude and appreciation to my family for their unyielding love and support throughout my life; this thesis would have simply been impossible without their help. I am completely indebted to my father, for his overwhelming care and encouragement. Being the person he is, he strived all along to support the family and spared no effort to provide the best life for me. As for my mother, I could not have asked for more from her. No words can express my appreciation to her for her everlasting love and her constant support whenever I encountered difficulties. Finally, I cannot leave out my sister and brothers who have always been there for me with their prayers and advice.

My sincere appreciation to my wife Rebecca whose dedication, love, and persistent confidence in me, have constantly taken the load off my shoulders. The past two years would not have been the same without her.

Above all, thanks be to God for all the blessings in my life and for giving me this opportunity to achieve a higher level of education. My life has become more bountiful.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

A variety of spontaneous gestures, such as finger, hand, body, or head movements are used to convey information in interactions among people. Gestures can hence be considered a natural communication channel with numerous aspects to be utilized in human-computer interaction. Up to date, most of our interactions with computers are performed with traditional keyboards, mouses, and remote controls designed mainly for stationary interaction. With the help of the great technological advancement, gesture-based interfaces can serve as an alternate modality for controlling computers, e.g. to navigate in office applications or to play some console games like Nintendo Wii [3]. Gesture-based interfaces can enrich and diversify interaction options and provide easy means to interact with the surrounding environment especially for handicapped people who are unable to live their lives in a traditional way.

On the other hand, mobile devices, such as PDAs, mobile phones, and other portable personal electronic devices provide new possibilities for interacting with various applications, if equipped with the necessary devices especially with the proliferation of low-cost MEMS (Micro-Electro-Mechanical Systems) technology. The majority of the new gen-

eration of smart phones, PDAs, and personal electronic devices are embedded with an accelerometer for various applications. Small wireless devices containing accelerometers could be integrated into clothing, wristwatches, or other personal electronic devices to provide a means for interacting with different environments. By defining some simple gestures, these devices could be used to control home appliances for example, or the simple up and down hand movement could be used to operate a garage door, or adjust the light intensity in a room or an office.

## 1.2 Gestures and Gesture Recognition

Expressive and meaningful body motions involving physical movements of the hands, arms, or face can be extremely useful for 1) conveying meaningful information, or 2) interacting with the environment. This involves: 1) a posture: a static configuration without the movement of the body part and 2) a gesture: a dynamic movement of the body part. Generally, there exist many-to-one mappings from concepts to gestures and vice versa. Hence gestures are ambiguous and incompletely specified. For example, the concept "stop" can be indicated as a raised hand with the palm facing forward, or an exaggerated waving of both hands over the head. Similar to speech and handwriting, gestures vary between individuals, and even for the same individual between different instances. Sometimes a gesture is also affected by the context of preceding as well as following gestures. Moreover, gestures are often language- and culture-specific. They can broadly be of the following types [4]:

- *hand and arm gestures*: recognition of hand poses, sign languages, and entertainment applications (allowing children to play and interact in virtual environments).
- *head and face gestures*: Some examples are a) nodding or head shaking, b) direction of eye gaze, c) raising the eyebrows, d) opening and closing the mouth, e) winking, f) flaring the nostrils, e) looks of surprise, happiness, disgust, fear, sadness, and

many others represent head and face gestures.

- *body gestures*: involvement of full body motion, as in a) tracking movements of two people having a conversation, b) analyzing movements of a dancer against the music being played and the rhythm, c) recognizing human gaits for medical rehabilitation and athletic training.

Gesture recognition refers to the process of understanding and classifying meaningful movements of the hands, arms, face, or sometimes head, however hand gestures are the most expressive, natural, intuitive and thus, most frequently used. Gesture recognition has become one of the hottest fields of research for its great significance in designing artificially intelligent human-computer interfaces for various applications which range from sign language through medical rehabilitation to virtual reality. More specifically, gesture recognition can be extremely useful for:

- Sign language recognition in order to develop aids for the hearing impaired. For example, just as speech recognition can transcribe speech to text, some gestures representing symbols through sign language can be transcribed into text.

- Socially assistive robotics. By using proper sensors and devices, like accelerometers and gyros, worn on the body of a patient and by reading the values from those sensors, robots can assist in patient rehabilitation.

- Developing alternative computer interfaces. Foregoing the traditional keyboard and mouse setup to interact with a computer, gesture recognition can allow users to accomplish frequent or common tasks using hand gestures to a camera.

- Interactive game technology. Gestures can be used to control interactions within video games providing players with an incredible sense of immersion in the totally engrossing environment of the game.

- Remote Controlling. Through the use of gesture recognition, various hand gestures can be used to control different devices, like secondary devices in a car, TV set, operating a garage door, and many others.

Gesture recognition consists of gesture spotting that implies determining the start and the end points of a meaningful gesture trace from a continuous stream of input signals, and, subsequently, segmenting the relevant gesture. This task is very difficult due to two main reasons. First of all, the segmentation ambiguity in the sense that as the hand motion switches from one gesture to another, there occur intermediate movements as well. These transition motions are also likely to be segmented and matched with template traces, and need to be eliminated by the model. The spatio-temporal variability is the second reason since the same gesture may vary dynamically in duration and, very possibly in shape even for the same gesturer.

## 1.3 Means of Acquisition

The first step in recognizing gestures is sensing the human body position, configuration (angles and rotations), and movement (velocities or accelerations). This can be done either by using sensing devices attached to the user which can take the form of magnetic field-trackers, instrumental (colored) gloves, and body suits or by using cameras and computer vision techniques [4]. Each sensing technology varies along several dimensions, including accuracy, resolution, latency, range of motion, user comfort, and cost. Glove-based gestural interfaces typically require the user to wear a cumbersome device and carry a load of cables connecting the device to the computer. This hinders the ease and naturalness of the user's interaction with the computer. Vision-based techniques, while overcoming this problem, need to contend with other problems related to occlusion of parts of the user's body. Most of the work on gesture recognition available in the literature is based on computer vision techniques [5] [6] [7] [8] [9]. Vision-based techniques

vary among themselves in 1) the number of cameras used, 2) their speed and latency, 3) the structure of environment such as lighting and speed of movement, 4) any user requirements such as any restrictions on clothing, 5) the low-level features used such as edges, regions, silhouettes, moments, histograms, and others, and 6) whether 2D or 3D is used [4]. Therefore, these limitations restrict the applications of vision-based systems in smart environments. More specifically, suppose you are enjoying watching movies in your home theatre with all the lights off. If you decide to change the volume of the TV with a gesture, it turns out to be rather difficult to recognize your gesture under poor lighting conditions using a vision-based system. Furthermore, it would be extremely uncomfortable and unnatural if you have to be directly facing the camera to complete a gesture.

A very promising alternative is to resort to other sensing techniques such as acceleration-based techniques or electromyogram-based (EMG-based) techniques. Acceleration-based gesture control is well-suited to distinguish noticeable, larger scale gestures with different hand trajectories. However, it is not very effective when it comes to detecting more subtle finger movements which is completely overcome by electromyogram-based techniques since they are very sensitive to muscle activation and thus provide rich information about finger movements. Yet, due to some inherent problems with EMG measurements including separability and reproducibility of measurements, the size of discriminable hand gesture set is limited to 4-8 gestures [10]. As a result, in this thesis and after examining all the sensing devices and techniques in literature, a 3-axis accelerometer is the sensing device utilized to acquire the data pertaining to gestures. Gesture recognition based on data from an accelerometer is an emerging technique for gesture-based interaction after the rapid development of the MEMS technology. Accelerometers are embedded in most of the new generation personal electronic devices such as Apple iPhone [11], Nintendo wii mote [12] which provide new possibilities for interaction in a wide range of applications, such as home appliances, in offices, and in video games. In the sequel, we represent


Figure 1.1: A sideview, a topview, and a bottomview of the wiimote

vectors by bold lower case letters, e.g. $\boldsymbol{r}$, matrices by bold upper case letters, e.g. $\mathbf{R}$, and sets by calligraphic upper case letters, $\mathcal{R}$.

### 1.3.1 Wiimote(Wii Remote)

The wiimote, short for Wii Remote, is the primary controller for Nintendo Wii console [3]. Fig. 1.1 shows a sideview, a topview, and a bottomview of the wiimote. The wiimote provides an inexpensive and robust packaging of several useful sensors, with the ability to rapidly relay the information to the computer [12]. The Wiimote connects to a computer wirelessly through Bluetooth technology. When both the 1 and 2 buttons are pressed simultaneously, the Wiimote's LEDs blink, indicating it is in discovery mode. A main feature of the wiimote is its motion sensing capability, which allows the user to interact with and manipulate items on screen via gesture recognition and the use of a 3-axis linear accelerometer and optical sensor technology. The accelerometer is physically rated to measure accelerations over a range of at least $\pm 3g$ with 10% sensitivity [13] and reports acceleration in the device's x-, y-, and z-directions, expressed in the unit of gravity $g$.

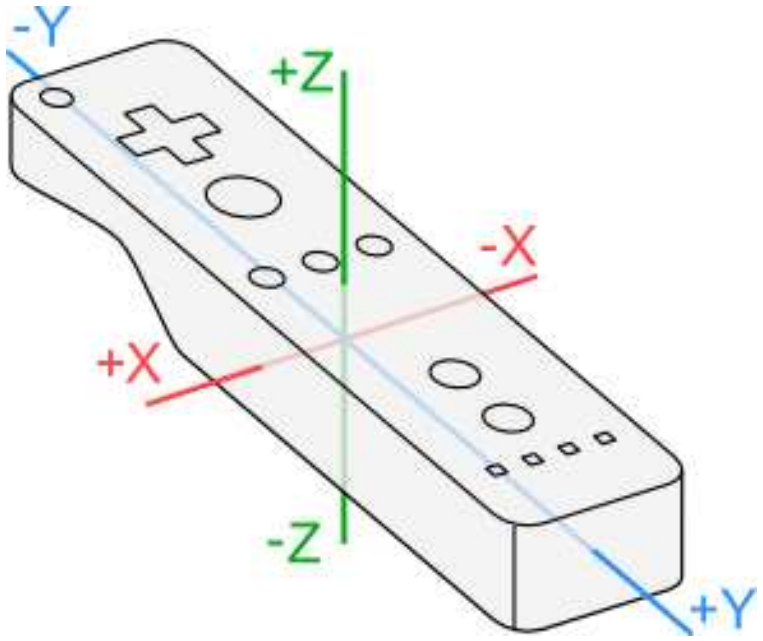


Figure 1.2: Wiimote with coordinate system labels

This reported acceleration vector is composed of the wiimote's actual acceleration $\boldsymbol{a}'$ and the Earth's gravity $\boldsymbol{a}_g$, so obtaining the actual acceleration of the movement requires subtracting the gravity vector from the wiimote's reported acceleration, i.e. $\boldsymbol{a} = \boldsymbol{a}' - \boldsymbol{a}_g$.

Due to its ease of use and familiarity in the world of human-computer interaction, the wiimote is chosen in this thesis as the device to acquire the different gesture traces and to relay the information to the computer. Accordingly, it is worth briefly describing how the remote works and how exactly it senses the change in acceleration. Fig. 1.2 shows a picture of the wiimote with labels to indicate the wiimote's coordinate system and to show how the acceleration changes in the shown directions [14]. For example, if you are holding the wiimote naturally, i.e. +z pointing up, and you move the wiimote towards the top right corner of the room you are in, then you would expect to observe a decrease in the x- and the y-accelerations and an increase in the z-acceleration.

The embedded 3-axis accelerometer detects the changes in acceleration by measuring the force exerted by a set of small proof masses inside of it with respect to its enclosure,

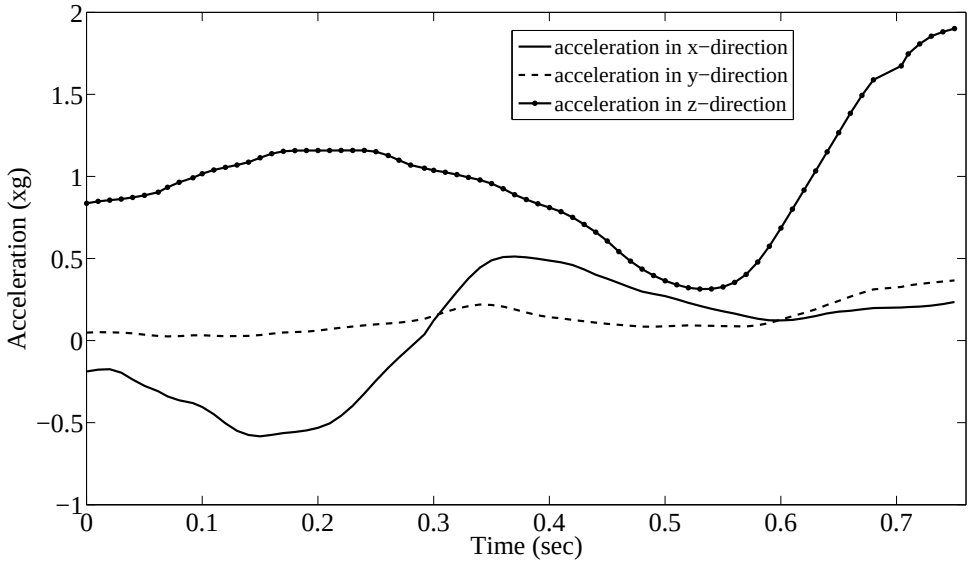


Figure 1.3: Acceleration waveforms defining the gesture of moving the hand in a clockwise circle

and therefore measures linear acceleration in a free fall frame of reference. In other words, if the Wiimote is in free fall, it will report zero acceleration. At rest and the normal orientation, it will report an upward acceleration in the positive z-direction equal to the acceleration due to gravity, $g$ (approximately 9.8 m/s) but in the opposite direction. Fig. 1.3 shows the acceleration waveforms for moving the hand in a clockwise circle. The solid line represents the acceleration in the x-direction, the dashed line represents the acceleration in the y-direction, and the solid-dot line represents the acceleration in the z-direction. According to the way the accelerometer is manufactured, tilting the accelerometer can result in different measurements for the same hand movement. Fig. 1.4 shows the acceleration waveforms representing a simple hand movement to the right. The wiimote is held in a horizontal orientation as shown in Fig. 1.2. However, Fig. 1.5 shows the acceleration waveforms representing the same hand movement to the right but with the wiimote held up-side down. Notice how the constant acceleration of magnitude $g$ is now in the negative z-direction. In this scenario, the waveforms in the x- and the y-directions are similar to a movement of the hand to the left. This can be a major source

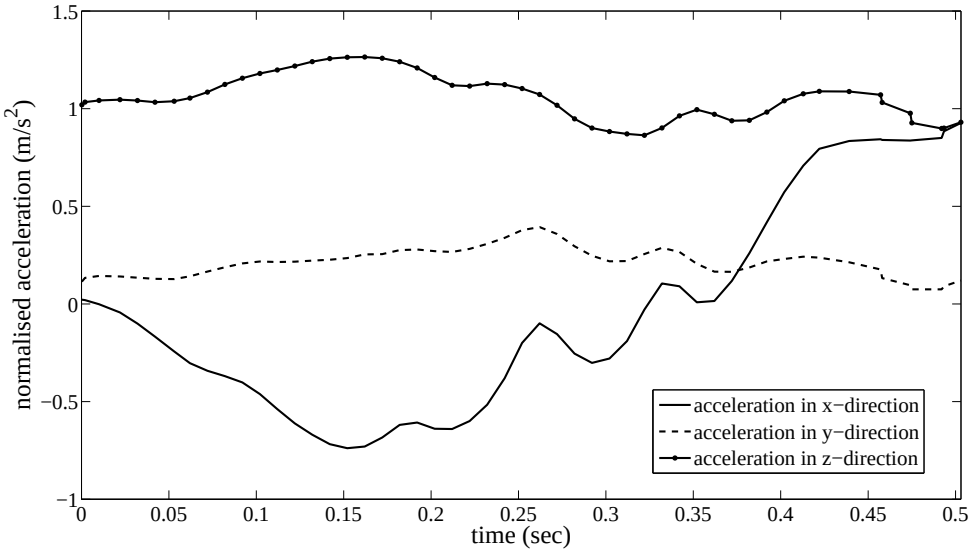


Figure 1.4: Acceleration waveforms defining the simple gesture of moving the hand to the right with the normal orientation of the wiimote

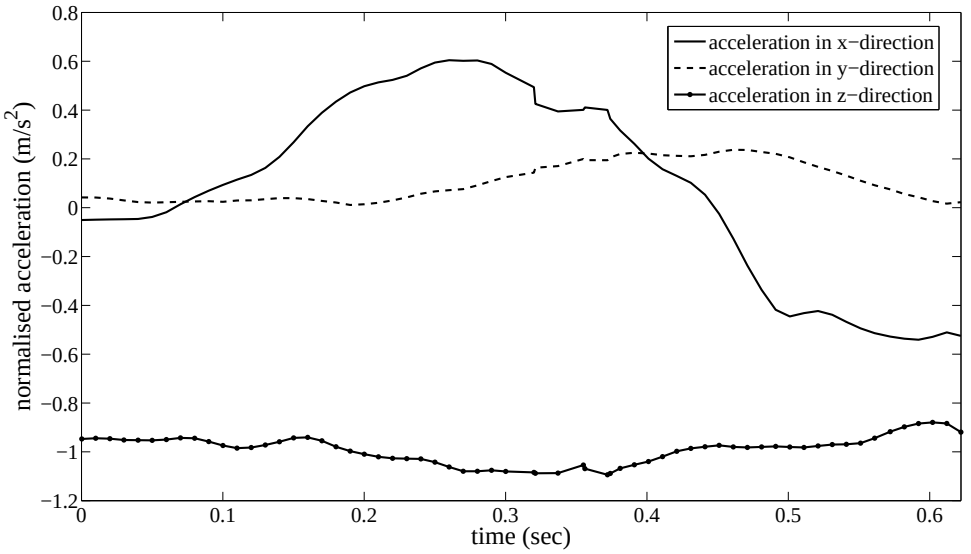


Figure 1.5: Acceleration waveforms defining the simple gesture of moving the hand to the right with a 90o clockwise rotation of the wiimote

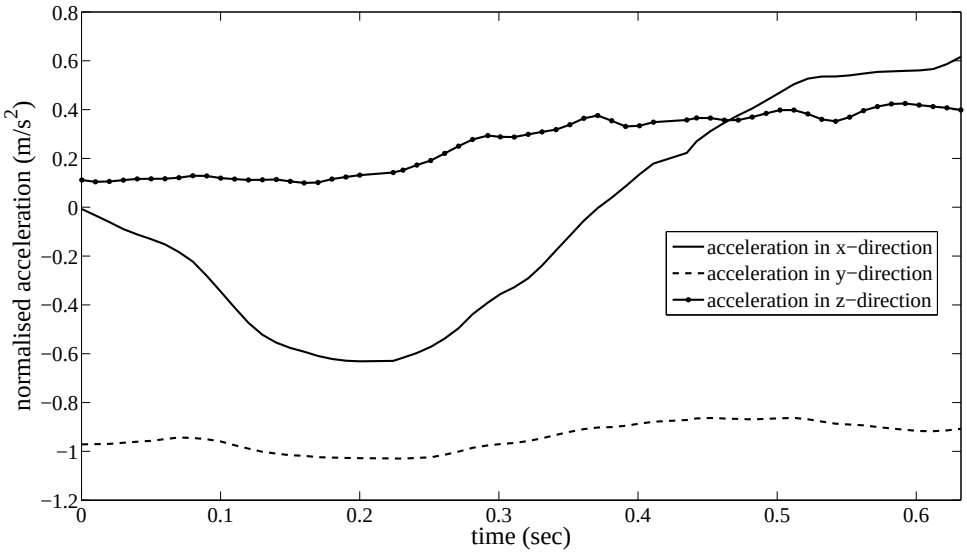


Figure 1.6: Acceleration waveforms defining the simple gesture of moving the hand to the right with the wiimote pointing up

of confusion. Yet, the acceleration in the z-direction can indicate that the wiimote is held up-side down.

Fig. 1.6 shows the acceleration waveforms, again, representing the same hand movement to the right but with the wiimote pointing up in a vertical orientation. In this case, the constant acceleration of magnitude $g$ is now in the negative y-direction. This highlights the fact that tilting the wiimote while performing a gesture can lead to different measurements.

All the above waveforms address how serious the issue of tilting can be and how it should definitely be taken into account while designing a gesture recognition system with data acquired using a wiimote.

### 1.3.2 Example of Importance of a 3-axis Accelerometer

Again, the accelerometer is a device for measuring the acceleration of moving objects. Fig. 1.7 shows the raw acceleration waveforms acquired by a wiimote for a person walking a distance of 22 ft. Theoretically, the speed can be obtained by integrating the acceleration

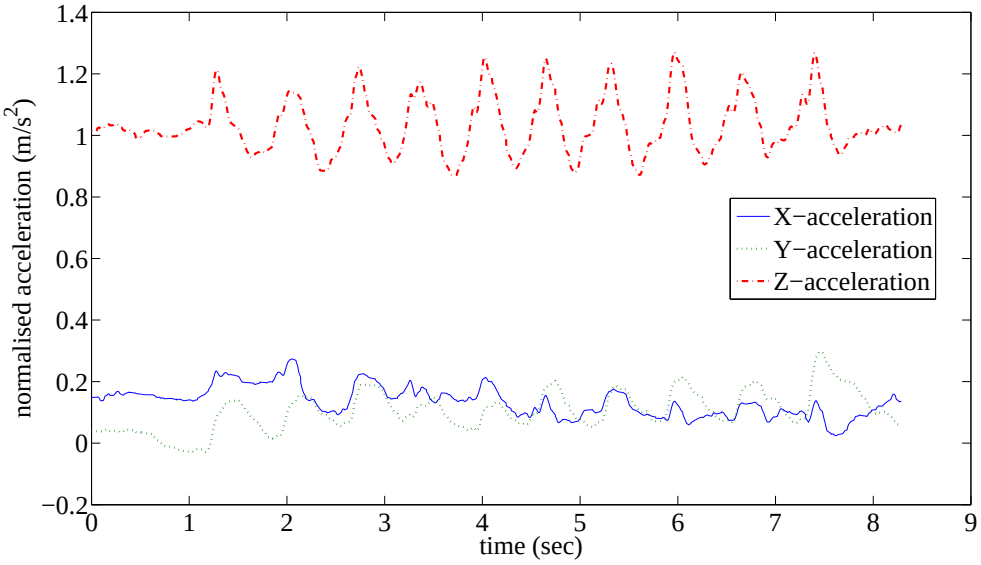


Figure 1.7: Acceleration waveforms for a person taking 10 steps

signals and further integrating the speed to obtain the distance. However, for indoor environments, the acceleration is very small and therefore it is very challenging to separate it from the noise due to the channel, offset drift, internal calibration, or even tilting. Consequently, integrating the acceleration results in integrating the error and leads to erroneous estimation of the distance. Alternatively, detecting the number of footsteps taken to walk a distance can help in knowing the distance covered if the stride size is known. In other words, the distance can be related to the number of steps and the step size as follows

$$distance = stride\ size \quad \times \quad number\ of\ steps \tag{1.1}$$

When walking with an accelerometer, the acceleration in the conventional z-direction fluctuates in a sinusoidal form as a result of the motion mechanism as shown in Fig. 1.7. Notice how clear the peaks are in the z-direction, and the acceleration waveforms in the x-, and y-directions seem to carry some useful information too. It turns out that adding the acceleration values in the three directions emphasizes the peaks and makes them more significant as is depicted in Fig. 1.8. Each significant peak represents one

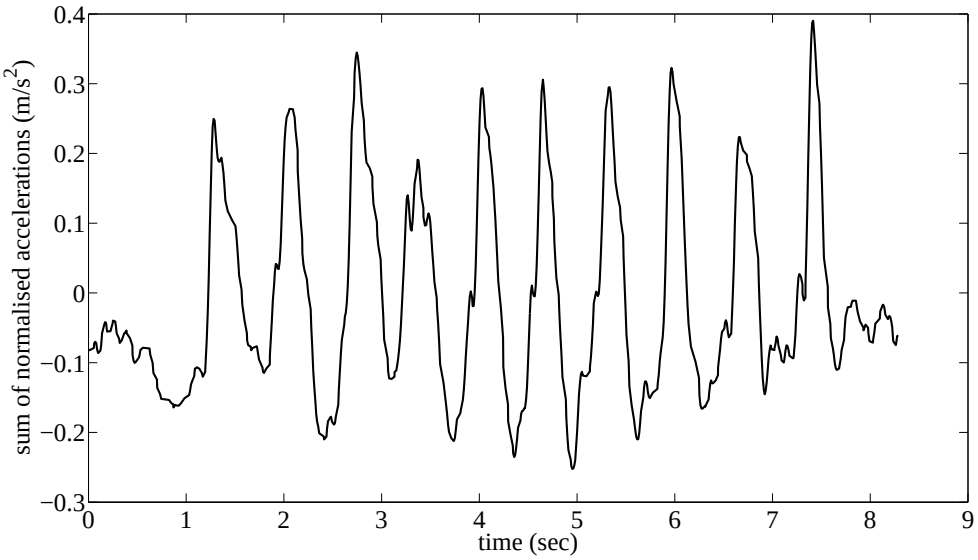


Figure 1.8: Equivalent acceleration waveform

step, and accordingly the subject walked 10 steps in order to cover the distance of 22 ft.

To estimate the number of steps, a simple zero crossing algorithm is used since it is obvious that the acceleration waveform crosses the zero line twice for each step. As a result, the number of steps can be found by detecting the number of zero crossings and dividing it by 2. As for the stride size, it is found empirically using the relationship proposed in [15] and which states that

$$stride\ size = \sqrt[4]{A_{max} - A_{min}} \times \quad K \tag{1.2}$$

where

- $A_{min}$ is the minimum acceleration measured in the z-axis in a single stride.
- $A_{max}$ is the maximum acceleration measured in the z-axis in a single stride.
- $K$ is a constant for unit conversion, obtained by walking training

Some more accurate, but more complex models for calculating the number of steps and step size are available in the literature [16] [17]. A simple distance calculating system

using a single 3-axis accelerometer is developed and simulated in Section 4.1 of Chapter 4. Simulations show the accelerometers can be of great significance and open the door to numerous applications.

## 1.4 Scope and Objectives

The objective of this work is to create a novel system that can automatically recognize hand gestures using a single 3-axis accelerometer and that utilizes the sparse nature of the gesture sequence. The proposed system can be implemented for user-dependent recognition, mixed-user recognition, and user-independent recognition. User-dependent recognition refers to the process of training a system using data from a single subject and using the system specifically for this subject only. Mixed-user recognition is more general in the sense that more than one subject can be used to train the system under the condition that the same subjects are involved in using the system. Finally, user-independent recognition, as the name implies, refers to the process of recognizing the input signals independently of the user. In other words, use of the system is not restricted to those involved in training the system. The proposed system evolves to be greatly competitive against the statistical models and the other existing gesture recognition systems in terms of computational complexity and recognition accuracy.

Chapter 2 presents the relevant background material. The details of how dynamic time warping and affinity propagation work are introduced. Then this chapter discusses random projection and how it can serve as an excellent approximation of signals with fewer samples. After that, the theory of Hidden Markov Models (HMMs) is presented followed by a literature review of previous systems that have been developed to recognize gestures using a single 3-axis accelerometer.

Chapter 3 introduces the proposed gesture recognition system. The system utilizes dynamic time warping to overcome the problem of different durations of gesture traces

and to compute a cost of similarity between the different gesture traces. Affinity propagation is then implemented to train the system and to create an exemplar or a set of exemplars for each gesture. Finally, the candidate gesture traces are randomly projected onto the same lower subspace to recognize an unknown gesture trace.

Chapter 4 presents a prototype implementation of the proposed system. The different parameters are defined to simulate the system. Results are presented and compared to other gesture recognition systems available in the literature that are based on a single 3-axis accelerometer. These include, continuous HMMs, uWave [1], and a system of discrete HMMs in [2].

Finally, in Chapter 5 we end this work with our concluding remarks and provide directions for future work.

## 1.5 Methodology

The first step in developing the gesture recognition system is to construct a dictionary of gestures on which the system will operate. So for a system of $N$ gestures, we collect $M$ traces for each gesture to create a database of the defined gestures. The database is then split into a training dataset and a testing dataset.

Gesture data for the proposed system refers to the acceleration of the hand, measured at different times $t$ using a single 3-axis accelerometer. However, hand gestures inherently suffer from temporal variations, and consequently, traces of the same gesture are of different lengths which poses the first major challenge in developing a gesture recognition system. In order to overcome this problem, dynamic time warping algorithm is used to compute a cost of similarity between the different gesture traces. Affinity propagation then operates on the costs of similarities to divide the training dataset into different clusters each represented by an "exemplar". This similarity cost computation and clustering using affinity propagation represent the core of the training stage. Therefore, the

output of the training stage is a set of exemplars for the different gestures defined in the database.

For testing, a user moves an accelerometer-equipped device to signal a particular gesture from the database. The objective of the gesture recognition system is to find out which gesture is intended by the user. To do that, the unknown trace is compared to the exemplars induced by affinity propagation to select a subset of coordinate traces. Based on the premise that the acquired gesture traces are sparse in nature, the candidate traces and the unknown gesture trace are projected onto the same lower-dimensional subspace in order to overcome the problem of different trace sizes. Projection is implemented using two definitions of a projection matrix: a matrix whose entities are independent realizations of Gaussian random variables and a matrix with sparse entities. Construction of random projection matrices is explained in full details in Section 2.4. Such definitions of the projection matrix are used since they satisfy the restricted isometry property (RIP) necessary for recovery of original data. After projection is done, the recognition problem is formulated as an $\ell_1$ minimization problem whose solution gives the label of the unknown trace.

## 1.6 Contributions

The main contribution of this work is the design of a novel gesture recognition system based solely on data from a single 3-axis accelerometer. The proposed system is very efficient in terms of recognition accuracy and computational complexity and is very competitive with other systems in the literature.

The following lists the contributions of this work, and any publications that refer to them:

- *Single 3-axis Accelerometer*: The system operates on data from only a single 3-axis accelerometer. Most of the systems in the literature combine data from an

accelerometer with data from other sensing devices like EMG sensors, or gyroscopes in order to enhance the system's performance. In addition, current systems that use only a single 3-axis accelerometer have limited applications, like being only user-dependent as is the case with uWave system, or have a small dictionary size. On the other hand, our proposed system uses only a single 3-axis accelerometer and functions competitively for any kind of recognition, user-dependent, mixed-user, or user-independent. Furthermore, our dictionary size is the largest in published studies to the best of our knowledge.

- *Dynamic Time Warping and Affinity Propagation*: Our system uses dynamic time warping to compute the cost of similarity between the different gesture traces followed by affinity propagation to split the training data into different clusters. Affinity propagation, being a very recent technique, has not been exploited in this field and according to the literature, it outperforms all the other clustering techniques. Besides, affinity propagation operates on a matrix of similarities between the gesture traces which makes it very suitable to the system set up (Chapter 2 and Chapter 3, [18], [19]).

- *Random Projection*: In the testing stage, comparing the unknown trace to the set of exemplars, found in the training stage, using dynamic time warping only does not suffice. Yet, the problem of the different gesture traces still poses a problem and hinders any further processing. Therefore, random projection proves to be an efficient way of projecting all the candidate traces and at the same time, preserving the similarities and the differences between the traces (Chapter 2 and Chapter 3, [18], [19]).

- $\ell_1$ *Minimization Formulation* : Formulating the recognition problem as an $\ell_1$ minimization problem is based on the premise that the gesture traces are sparse in nature. So, after the candidate traces are projected, the recognition problem is

transformed into an $\ell_1$ minimization problem and its solution leads to the classification of the unknown trace (Chapter 3, [18], [19]).

# Chapter 2

# Background

## 2.1 Dynamic Time Warping

The Euclidean distance is a very common metric used in many applications to represent the degree of similarity between any two sequences $\boldsymbol{p} = \{p_1, \ldots, p_n\}$ and $\boldsymbol{q} = \{q_1, \ldots, q_n\}$. The cost of similarity based on the Euclidean distance metric is computed as

$$d_{Euclid}(\boldsymbol{p}, \boldsymbol{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.1}$$

This kind of metric is applicable when the two sequences $\boldsymbol{p}$ and $\boldsymbol{q}$ are of the same length. However, in case $\boldsymbol{p}$ and $\boldsymbol{q}$ are not of the same length then the Euclidean distance is not applicable as a similarity measure. Instead, a more flexible method that can find the best mapping from elements in $\boldsymbol{p}$ to those in $\boldsymbol{q}$ must be sought in order to find a similarity cost between two sequences of different lengths.

Dynamic time warping (DTW) is an algorithm that measures the similarity between two time sequences of different durations. DTW matches two time signals by computing a temporal transformation causing the signals to be aligned. The alignment is optimal in the sense that a cumulative distance measure between the aligned samples is minimized

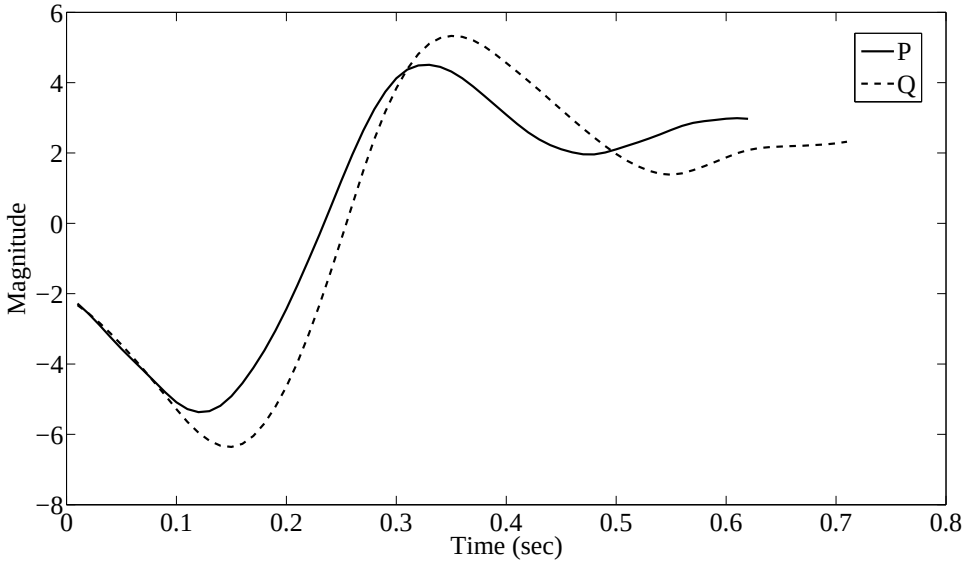


Figure 2.1: Two time sequences P and Q that are similar but out of phase

[20].

Assume that the two sequences, $\boldsymbol{p}$ and $\boldsymbol{q}$, are similar but are out of phase and are of length $n$ and $m$, respectively, where $\boldsymbol{p} = \{p_1, \ldots, p_n\}$ and $\boldsymbol{q} = \{q_1, \ldots, q_m\}$ as shown in Fig 2.1. The objective is to compute the matching cost: DTW($\boldsymbol{p}, \boldsymbol{q}$).

To align the two sequences using DTW, we construct an $n \times m$ matrix where the $(i, j)$-th entry of the matrix indicates the distance $d(p_i, q_j)$ between the two points $p_i$ and $q_j$, where $d(p_i, q_j) = (p_i - q_j)^2$. The cost of similarity between the two sequences is based on a warping path $W$ that defines a mapping between $\boldsymbol{p}$ and $\boldsymbol{q}$. The $k$th element of $W$ is defined as $w_k$ which is a pointer to the $k$-th element on the path, usually represented by the indices of the corresponding element. So, $W$ is defined as

$$W = \langle w_1, w_2, \ldots, w_k, \ldots, w_L \rangle \tag{2.2}$$

such that,

$$\max(m, n) \leq L < n + m - 1 \tag{2.3}$$

The warping path is subject to two main constraints [20]:

i. Boundary conditions: $w_1 = (1, 1)$ and $w_L = (n, m)$ and this entails that the warping path starts and ends in diagonally opposite corners of the matrix.

ii. Continuity and Monotocity: Given $w_k = (a, b)$, and $w_{k-1} = (a^{'}, b^{'})$, then $a^{'} \leq a \leq a^{'} + 1$ and $b^{'} \leq b \leq b^{'} + 1$. This casts a restriction on the allowable steps in the path to adjacent cells including diagonally adjacent cells, and forces the path's indices to be monotonically increasing.

There are exponentially many warping paths that satisfy the above constraints. However, we are seeking only the path that minimizes the warping cost. In other words,

$$\mathrm{DTW}(\boldsymbol{p}, \boldsymbol{q}) = \min\{\sqrt{\sum_{k=1}^{L} \mathrm{DTW}(w_k)}\} \tag{2.4}$$

The monotonically increasing warping path that minimizes the similarity cost between $\boldsymbol{p}$ and $\boldsymbol{q}$ is found by applying the dynamic programming formulation below, which defines the cumulative cost $D_{i,j}$ as the cost $d(p_i, q_j)$ in the current cell plus the minimum of the cumulative cost of the adjacent elements,

$$D_{i,j} = d(p_i, q_j) + min\left\{D_{i,j-1}, D_{i-1,j}, D_{i-1,j-1}\right\} \tag{2.5}$$

and consequently,

$$\mathrm{DTW}(\boldsymbol{p}, \boldsymbol{q}) = D_{n,m} \tag{2.6}$$

The resulting new matrix is depicted in Fig. 2.2 showing the optimum warping path between the two sequences and demonstrating the constraints that the warping path is satisfying.

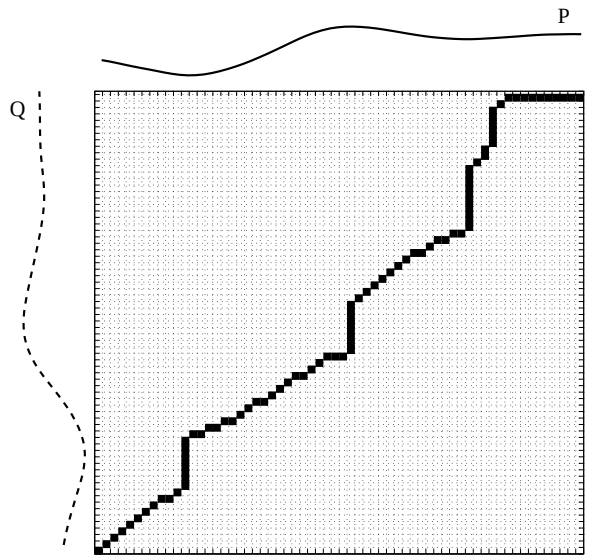


Figure 2.2: Aligning the two sequences by finding the optimum warping path

### 2.1.1 Example of Dynamic Time Warping

An example of how to use DTW to compute the similarity cost between two sequences $\boldsymbol{p}$ and $\boldsymbol{q}$ of different lengths is given here. Let

$$\boldsymbol{p} = [1\ 2\ 2\ 3\ 3\ 4\ 4]$$
$$\boldsymbol{q} = [1\ 2\ 3\ 4]$$

where $\boldsymbol{q}$ is a compressed version of $\boldsymbol{p}$. In this case, $n = 7$ and $m = 4$ and we start by constructing a matrix which is $7 \times 4$ and placing $\boldsymbol{p}$ and $\boldsymbol{q}$ on each side of the matrix. The $(i, j)$-th element of the matrix consists of the $d(p_i, q_j) = (p_i - q_j)^2$ as shown in the table to the left below. After the matrix to the left is filled, a second matrix based on it is constructed where each $(i, j)$-th element of the second matrix represents the cost in the current cell of the matrix to the left plus the minimum cost from the adjacent cells in the matrix to the right as per the formulation in( 2.5). The cost between the two sequences $\boldsymbol{p}$ and $\boldsymbol{q}$ is equal to the cost in the top right corner of the second matrix which is 0 in this example.

| | | | | |
|---|---|---|---|---|
| 4 | 9 | 4 | 1 | 0 |
| 4 | 9 | 4 | 1 | 0 |
| 3 | 4 | 1 | 0 | 1 |
| 3 | 4 | 1 | 0 | 1 |
| 2 | 1 | 0 | 1 | 4 |
| 2 | 1 | 0 | 1 | 4 |
| 1 | 0 | 1 | 4 | 9 |
| | 1 | 2 | 3 | 4 |

→

| | | | |
|---|---|---|---|
| 28 | 10 | 2 | 0 |
| 19 | 6 | 1 | 0 |
| 10 | 2 | 0 | 1 |
| 6 | 1 | 0 | 1 |
| 2 | 0 | 1 | 5 |
| 1 | 0 | 1 | 5 |
| 0 | 1 | 5 | 14 |

Table 2.1: Computation of the similarity cost using DTW

## 2.2 Affinity Propagation

Clustering data based on a measure of similarity is a critical step in engineering systems. A common approach is to use data to learn a set of centers such that the sum of squared errors between data points and their nearest centers is small. When the centers are selected from actual data points, they are called exemplars. A new technique for data clustering is the Affinity Propagation (AP) algorithm [21]. AP is an algorithm that simultaneously considers all data points as potential exemplars and recursively transmits real-valued messages until a good set of exemplars and clusters emerges. An exemplar is a term used to represent the center selected from the the actual data points. AP does not require that the number of clusters be known prior to clustering, instead, the clusters emerge naturally.

### 2.2.1 Similarity Function

AP algorithm takes an input function of similarities, $s(i, j)$, where $s(i, j)$ reflects how well suited data point $j$ is to be the exemplar of data point $i$. AP aims to maximize the similarity $s(i, j)$ for every data point $i$ and its chosen exemplar $j$. For example, the similarity function could be the negative Euclidean distance between data points. Negative Euclidean distance is used so that a maximum similarity corresponds to the closest data points.

In addition to the measure of similarity, AP takes as input a set of real numbers,

known as self-similarity $s(i,i)$, or *preference* $(p)$ for each data point. The *preference* $(p)$ influences the number of exemplars that are identified. Initializing a data point with a larger or smaller self-similarity, respectively increases or decreases the likelihood of the data point becoming an exemplar. If all the data points are initialized with the same constant self-similarity, then all data points are equally likely to become exemplars. The *preference* $(p)$ also controls how many clusters are produced. By increasing and decreasing this common self-similarity input, the number of clusters produced is increased and decreased respectively. If all data points are assigned the median of the input similarities, a moderate number of clusters is produced, and if they are assigned the minimum of the input similarities, the smallest number of clusters is produced. This algorithm can generate better clusters, compared to other clustering techniques like $K$-means clustering, because of its initialization-independent property [21].

### 2.2.2 Message Passing: Responsibility and Availability

Clustering is based on the exchange of two types of messages: the "responsibility" message to decide which data points are exemplars and the "availability" message, to decide which cluster a data point belongs to:

- the *responsibility message* $r(i,j)$ sent from data point $i$ to candidate exemplar $j$, reflects the accumulated evidence for how well-suited data point $j$ is to serve as the exemplar for $i$, taking into account other potential exemplars $j^{'}$ for data point $i$, that is

$$r(i,j) = s(i,j) - \max_{j^{'}s.t.j^{'}\neq j} \Big\{ a(i,j^{'}) + s(i,j^{'}) \Big\} \tag{2.7}$$

  where $i \neq j$, and $s(i,j)$ is the similarity between data point $i$ and data point $j$ and $a(i,j)$ is the availability message defined below.

- the *availability message* $a(i,j)$, sent from candidate exemplar $j$ to data point $i$, reflects the accumulated evidence for how appropriate it would be for data point $i$

to choose $j$ as its exemplar, taking into account the support from other data points that $j$ should be an exemplar, that is

$$a(i,j) = \min \left\{ 0, r(j,j) + \sum_{i^{'} s.t. i^{'} \neq i,j} \max\{0, r(i^{'},j)\} \right\} \tag{2.8}$$

- The self-responsibility, $r(j,j)$ and self-availability, $a(j,j)$ are two additional messages calculated for each data point, $j$. Both of these messages reflect accumulated evidence that $j$ is an exemplar, and they are used to find the clusters. The self-responsibility bases exemplar suitability on input preference and the maximum availability received from surrounding data points whereas the self-availability bases exemplar suitability on the number and strength of positive received responsibilities. Mathematically,

$$r(j,j) = s(j,j) - \max_{j^{'} s.t. j^{'} \neq i} \left\{ a(j,j^{'}) + s(j,j^{'}) \right\} \tag{2.9}$$

$$a(j,j) = \sum_{i^{'} s.t. i^{'} \neq j} \max\{0, r(i^{'},j)\} \tag{2.10}$$

### 2.2.3 Cluster Decisions

In AP algorithm, the exemplar of each data point $i$ is found using the following equation,

$$exemplar_i = arg\max_j \{a(i,j) + r(i,j)\} \tag{2.11}$$

This clustering procedure may be performed at any iteration of the algorithm, but final clustering decisions should be made once the algorithm stabilizes. The algorithm can be terminated once exemplar decisions become constant for some number of iterations, indicating that the algorithm has converged. It should be noted that the algorithm possesses another useful feature: it is possible to determine when a specific data point

has converged to exemplar status for a specific iteration. When a data point's self-responsibility plus self-availability becomes positive, that data point has become the exemplar for its cluster.

## 2.3 Compressive Sensing

Compressive sensing [22] is a method that allows us to recover signals from far fewer measurements than the traditional sampling methods. Assume that the received signal can be represented as a $d \times 1$ vector $\boldsymbol{x} = \boldsymbol{\Psi s}$ where $\boldsymbol{\Psi}$ is a $d \times d$ basis matrix and $\boldsymbol{s}$ is a $d \times 1$ sparse vector that has only $l_s \ll d$ non-zero elements. The locations of the non-zero elements in $\boldsymbol{s}$ are unknown. Signal $\boldsymbol{x}$ is compressed using a $k \times d$ sensing matrix $\boldsymbol{\Phi}$, which yields the measurement vector $\boldsymbol{y}$ of dimension $k$ as follows

$$\boldsymbol{y} = \boldsymbol{\Phi x} = \boldsymbol{\Phi \Psi s} \tag{2.12}$$

It has been shown that $\boldsymbol{s}$ can be recovered exactly if $k$ satisfies the following inequality

$$k \geq c l_s \log\left(d/l_s\right) \tag{2.13}$$

where $c$ is a constant and $l_s$ is the sparsity level [22].

The signal can be reconstructed by solving the following $\ell_1$ norm minimization problem

$$\begin{aligned} &\min_{\boldsymbol{s}} \|\boldsymbol{s}\|_1 \\ &\text{subject to } \ \boldsymbol{y} = \boldsymbol{\Phi \Psi s} \end{aligned} \tag{2.14}$$

### 2.3.1 Restricted Isometry Property (RIP)

To explain the concept of Restricted Isometry Property (RIP), we refer to the approach outlined in [23] by describing the concept of compressive sensing in terms of encoder/decoder and approximation error. In the discrete compressive sensing problem, we are interested in economically recording the information about a vector or a signal $\boldsymbol{x} \in \mathcal{R}^d$. We allocate a budget of $k$ nonadaptive questions to ask about $\boldsymbol{x}$. Each question takes the form of a linear functional applied to $\boldsymbol{x}$. Thus, the information we extract from $\boldsymbol{x}$ is given by

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{x}, \tag{2.15}$$

where $\boldsymbol{\Phi}$ is a $k \times d$ matrix and $y \in \mathcal{R}^k$. The matrix $\boldsymbol{\Phi}$ maps $\mathcal{R}^d$ into $\mathcal{R}^k$, where $d \gg k$.

To extract the information that $\boldsymbol{y}$ holds about $\boldsymbol{x}$, we use a decoder $\Delta$ that maps from $\mathcal{R}^k$ back into $\mathcal{R}^d$. The role of $\Delta$ is to provide an approximation $\bar{\boldsymbol{x}} := \Delta(\boldsymbol{y}) = \Delta(\boldsymbol{\Phi}\boldsymbol{x})$ to $\boldsymbol{x}$. The mapping $\Delta$ is typically nonlinear. The central question of compressive sensing is: What are the good encoder-decoder pairs $(\boldsymbol{\Phi}, \Delta)$?

To measure the performance of an encoder-decoder pair $(\boldsymbol{\Phi}, \Delta)$, we introduce a norm $\|\cdot\|_X$ in which we measure error. Then,

$$E(\boldsymbol{x}, \boldsymbol{\Phi}, \Delta)_X := \| \boldsymbol{x} - \Delta(\boldsymbol{\Phi}\boldsymbol{x}) \|_X \tag{2.16}$$

is the error of the encoder-decoder on $\boldsymbol{x}$. More generally, if $\mathcal{K}$ is any closed and bounded set contained in $\mathcal{R}^d$, then the error of this encoder-decoder on $\mathcal{K}$ is given by

$$E(\mathcal{K}, \boldsymbol{\Phi}, \Delta)_X = \sup_{\boldsymbol{x} \in \mathcal{K}} E(\boldsymbol{x}, \boldsymbol{\Phi}, \Delta)_X. \tag{2.17}$$

Thus, the error on the set $\mathcal{K}$ is determined by the largest error on $\mathcal{K}$. To address the question of what constitutes good encoder-decoder pairs, we introduce $\mathcal{A}_{k,d} := \{(\boldsymbol{\Phi}, \Delta) :$

$\mathbf{\Phi}$ is $k \times d\}$. The best possible performance of an encoder-decoder on $\mathcal{K}$ is given by

$$E_{k,d}(\mathcal{K})_X := \inf_{(\mathbf{\Phi},\Delta)\in\mathcal{A}_{k,d}} E(\mathcal{K},\mathbf{\Phi},\Delta)_X. \tag{2.18}$$

This is the so-called "minimax" way of measuring optimality that is prevalent in approximation theory, information-based complexity, and statistics [23].

The decoder $\Delta$ is important in practical applications in compressive sensing and also in the above formulation. Candès, Romberg, and Tao [24] showed that decoding can be accomplished by the linear program

$$\Delta(\boldsymbol{y}) := \arg\min_{\boldsymbol{x}:\mathbf{\Phi}\boldsymbol{x}=\boldsymbol{y}} \| \boldsymbol{x} \|_{\ell_1^d} . \tag{2.19}$$

Furthermore, Candès and Tao [25] introduced the isometry condition on matrices $\mathbf{\Phi}$ and established its important role in compressive sensing. Given a matrix $\mathbf{\Phi}$ and any set $\mathcal{T}$ of column indices with the number of elements $n_{\mathcal{T}}$, we denote by $\mathbf{\Phi}_{\mathcal{T}}$ the $k \times n_{\mathcal{T}}$ matrix composed of these columns. Similarly, for a vector $\boldsymbol{x} \in \mathcal{R}^d$, we denote by $\boldsymbol{x}_{\mathcal{T}}$ the vector obtained by retaining only the entries in $\boldsymbol{x}$ corresponding to the column indices $\mathcal{T}$. We say that $\mathbf{\Phi}$ satisfies the *Restricted Isometry Property* (RIP) of order $m$ if there exists a $\delta_m \in (0,1)$ such that

$$(1-\delta_m) \| \boldsymbol{x}_{\mathcal{T}} \|_{\ell_2^d}^2 \leq \| \mathbf{\Phi}_{\mathcal{T}}\boldsymbol{x}_{\mathcal{T}} \|_{\ell_2^k}^2 \leq (1+\delta_m) \| \boldsymbol{x}_{\mathcal{T}} \|_{\ell_2^d}^2 \tag{2.20}$$

holds for all sets $\mathcal{T}$ with $n_{\mathcal{T}} \leq m$. The condition (2.20) is equivalent to requiring that the Grammian matrix $\mathbf{\Phi}_{\mathcal{T}}^T\mathbf{\Phi}_{\mathcal{T}}$ has all of its eigenvalues in $[1-\delta_m, 1+\delta_m]$, where $\mathbf{\Phi}_{\mathcal{T}}^T$ denotes the transpose of $\mathbf{\Phi}_{\mathcal{T}}$.

The "good" matrices for compressive sensing should satisfy (2.20) for the largest possible $m$. For example, Candès and Tao [25] show that whenever $\mathbf{\Phi}$ satisfies the RIP

of order $3m$ with $\delta_{3m} < 1$, then

$$\| \boldsymbol{x} - \Delta(\boldsymbol{\Phi}\boldsymbol{x}) \|_{\ell_2^d} \leq \frac{C_2 \sigma_m(\boldsymbol{x})_{\ell_1^d}}{\sqrt{m}}, \tag{2.21}$$

where $\sigma_m(\boldsymbol{x})_{\ell_1^d}$ denotes the $\ell_1$ error of the best $m$-term approximation, and the constant $C_2$ depends only on $\delta_{3m}$. The original proof of (2.21) is given in [26] and the proof for this particular formulation is provided in [27].

The question before us now is how we can construct matrices $\boldsymbol{\Phi}$ that satisfy the RIP for the largest possible range of $m$. The most prominent matrices are the $k \times d$ random matrices $\boldsymbol{\Phi}$ whose entries $\phi_{i,j}$ are independent realizations of Gaussian random variables [23]

$$\phi_{i,j} \sim \mathcal{N}(0, \frac{1}{n}). \tag{2.22}$$

One can also use matrices where the entries are independent realizations of $\pm 1$ Bernoulli random variables [28]

$$\phi_{i,j} = \begin{cases} \frac{+1}{\sqrt{n}} \text{ with probability } \frac{1}{2}, \\ \frac{-1}{\sqrt{n}} \text{ with probability } \frac{1}{2}, \end{cases} \tag{2.23}$$

or related distributions such as

$$\phi_{i,j} = \begin{cases} +\sqrt{\frac{3}{n}} \text{ with probability } \frac{1}{6}, \\ 0 \quad\quad \text{with probability } \frac{2}{3}, \\ -\sqrt{\frac{3}{n}} \text{ with probability } \frac{1}{6}. \end{cases} \tag{2.24}$$

## 2.4 Random Projection

Random Projection (RP) has recently emerged as a powerful technique for dimensionality reduction [29] [30]. In RP, the original $d$-dimensional data is projected onto a $k$-dimensional ($k \ll d$) subspace using a $k \times d$ random matrix $\mathbf{A}$ whose columns have

unit lengths. Using matrix notation, let $\mathbf{X}_{d\times n}$ be the original set of $n$ $d$-dimensional observations, then the projection problem can be formulated as,

$$\mathbf{X}^{\mathrm{RP}}_{k\times n} = \mathbf{A}_{k\times d}\mathbf{X}_{d\times n} \tag{2.25}$$

where $\mathbf{X}^{\mathrm{RP}}_{k\times n}$ represents the projected data onto the lower $k$-dimensional subspace.The concept of RP is inspired by the Johnson-Lindenstrauss theorem [31].

Strictly speaking, (2.25) is not a projection because the projection matrix $\mathbf{A}$ is generally not orthogonal and such a linear mapping can result in significant distortion to the original data set. One solution is to orthogonalize $\mathbf{A}$ but this can be computationally very expensive. Alternatively, we can resort to the fact that in a high-dimensional space, the number of almost orthogonal directions is much larger than the number of orthogonal directions [32]. Therefore, vectors having random directions can be sufficiently close to orthogonal and thus can offer the necessary preservation of the original data set after projection.

In case of RP, the matrix $\mathbf{A}$ is the general case of $\mathbf{\Phi}$ in Sections 2.3 and 2.3.1. $\mathbf{A}$ is a sampling operator for $\mathbf{X}$, and is invertible if each $\boldsymbol{x} \in \mathbf{X}$ is uniquely determined by its sampled or projected data $\mathbf{A}\boldsymbol{x}$; this means if for every $\boldsymbol{u}, \boldsymbol{v} \in \mathbf{X}$, $\mathbf{A}u = \mathbf{A}v$ then $\boldsymbol{u} = \boldsymbol{v}$. In other words, $\mathbf{A}$ is a one-to-one mapping between $\mathbf{X}^{\mathrm{RP}}$ and $\mathbf{X}$ and this allows a unique identification for each $x \in \mathbf{X}$ from $\mathbf{A}\boldsymbol{x}$. However, practically, we want that a small change in $\boldsymbol{x}$ only result in a small change in its sampled or projected data $\mathbf{A}\boldsymbol{x}$. Therefore, we consider a stricter condition given by

$$\begin{aligned} \alpha \parallel \boldsymbol{u} - \boldsymbol{v} \parallel^2_{\mathcal{H}} &\leq \parallel \mathbf{A}u - \mathbf{A}v \parallel^2_{l_2} = \sum_n |\langle \boldsymbol{u} - \boldsymbol{v}, \psi_n^2 \rangle|^2 \\ &\leq \beta \parallel \boldsymbol{u} - \boldsymbol{v} \parallel^2_{\mathcal{H}} \end{aligned} \tag{2.26}$$

where $\alpha$ and $\beta$ are constants with $\alpha > 0$ and $\beta < \infty$, $\mathcal{H}$ is an ambient Hilbert space, and $\psi_n \in \mathcal{H}$ is a sampling vector [33].

The sampling condition (2.26) on $\mathbf{A}$ is related to the important concept of restricted isometry property (RIP) described in Section 2.3.1, and is interestingly the same as RIP if $\mathbf{X}$ has sparse columns and the columns come from the same subspace [33] [34]. As mentioned in Section 2.3.1, any distribution of zero mean and unit variance satisfies the sampling condition in (2.26). As far as this thesis is concerned, Gaussian distributions as well as a special case of the distribution in (2.24) given below

$$a_{ij} = \sqrt{3} \cdot \begin{cases} +1 \text{ with probability } \frac{1}{6} \\ 0 \text{ with probability } \frac{2}{3} \\ -1 \text{ with probability } \frac{1}{6} \end{cases} \tag{2.27}$$

will be investigated. The distribution (2.27) has been reported to result in further computational savings since computations can be done using integer arithmetics [28].

## 2.5 Hidden Markov Models

Consider a system which may be described at any time as being in one of a set of $N$ distinct states, $S_1$, $S_2$, ..., $S_N$. At regularly spaced discrete times, the system undergoes a change of state according to a set of probabilities associated with the state. We denote the time instants associated with the sate changes as $t = 1, 2, \ldots$, and we denote the actual state at time $t$ as $q_t$. Generally, a full probabilistic description of such a system would require specification of the current state (at time $t$), as well as all the predecessor states. However, a system is first-order Markov if the conditional probability density of the current state $q_t$, given all present and past states, depends only on the most recent state, represented in the following formulation:

$$P[q_t = S_j | q_{t-1} = S_i, q_{t-2} = S_k, \cdots] = P[q_t = S_j | q_{t-1} = S_i]. \tag{2.28}$$

Furthermore, we only consider those systems in which the right-hand side of (2.28) is independent of time, thereby leading to the set of state transition probabilities $a_{ij}$ of the form

$$a_{ij} = P[q_t = S_j | q_{t-1} = S_i], \;\; 1 \leq i, \;\; j \leq N \tag{2.29}$$

with the state transition coefficients having the properties

$$\begin{aligned} a_{ij} &\geq 0 \\ \textstyle\sum_{j=1}^{N} a_{ij} &= 1 \end{aligned} \tag{2.30}$$

A Hidden Markov Model (HMM) is a double stochastic process governed by 1) an underlying Markov chain with a finite number of states, and 2) a set of random functions associated with each state. HMM has proven to be extremely efficient in modeling time-series with spatial and temporal variability as well as handling undefined patterns. The states in an HMM are connected by transitions. Each transition is associated with a pair of probabilities:
(i) transition probability, the probability of transitioning from the current state to a different state, and
(ii) output probability, the probability of producing an output symbol from a finite alphabet given a state. An HMM is formally characterized by the following [35]:

i. $N$, the number of states in the model. Although the states are hidden, there is frequently some physical significance associated with the states or the sets of states in the model. States can be interconnected in different ways. Ergodic model interconnection is the simplest of all, and it represents the interconnection in which any states can be reached from any other state. The individual states are denoted as $S = \{s_1, s_2, s_3, \ldots, s_N\}$, and the state at time $t$ is denoted by the random variable $q_t$.

ii. $M$, the number of distinct observation symbols per state, i.e. the discrete alphabet

size. The observation symbols, in other words, model the physical output of the system. The individual symbols are denoted as $V = \{v_1, v_2, v_3, \ldots, v_M\}$, and the observation at time $t$ is denoted by the random variable $O_t$.

iii. The state transition probability distribution $\mathbf{A} = \{a_{ij}\}$ where

$$a_{ij} = P(q_{t+1} = S_j | q_t = S_i), \quad 1 \leq i, j \leq N \tag{2.31}$$

For the special case where any state can reach any other state in a single step, we have $a_{ij} > 0$ for all $i$,$j$. For other types of HMMs, $a_{ij} = 0$ for one or more $(i, j)$ pairs.

iv. The observation symbol probability distribution in state $j$, $\mathbf{B} = \{b_j(k)\}$, where

$$\begin{aligned} b_j(k) = P(v_k \ \ at \ t | q_t = S_j), \quad & 1 \leq j \leq N \\ & 1 \leq k \leq M \end{aligned} \tag{2.32}$$

v. The initial state distribution $\pi = \pi_i$ where

$$\pi_i = P(q_1 = S_i) \quad 1 \leq i \leq N \tag{2.33}$$

Given appropriate values of $N, M, \mathbf{A}, \mathbf{B}$, and $\pi$, the HMM can be used as a generator to give an observation sequence

$$O = O_1 O_2 \cdots O_T \tag{2.34}$$

where each observation $O_t$ is one of the symbols from $V$, and $T$ is the number of observations in the sequence. The sequence $O$ is generated as follows

1. Choose an initial state $q_1 = S_i$ according to the initial state distribution $\pi$.

2. Set $t = 1$.

3. Choose $O_t = v_k$ according to the symbol probability distribution in state $S_i$, i.e. $b_i(k)$.

4. Transit to a new state $q_{t+1} = S_j$ according to the state transition probability distribution for state $S_i$, i.e. $a_{ij}$.

5. Set $t = t + 1$; return to step 3 if $t < T$, otherwise terminate the procedure.

For convenience, an HMM is represented in a compact form as

$$\lambda = (\mathbf{A}, \mathbf{B}, \pi) \tag{2.35}$$

to indicate the complete parameter set of the model.

### 2.5.1 Three Basic Problems for HMMs

Given the form of the HMM, there are three basic problems of interest that must be solved for the model to be useful in real-world applications [35]. These problems are the following:

**Problem 1** : Given the observation sequence $O = O_1 O_2 \cdots O_T$, and a model $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$, how do we efficiently compute $P(O|\lambda)$, the probability of the observation sequence, given the model?

**Problem 2** : Given the observation sequence $O = O_1 O_2 \cdots O_T$, and the model $\lambda$, how do we choose a corresponding state sequence $Q = q_1 q_2 \cdots q_T$ which is optimal is some meaningful sense?

**Problem 3** : How do we adjust the model parameter $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ to maximize $P(O|\lambda)$?

### 2.5.2 Solutions to the Three Basic Problems of HMMs

As far as this thesis is concerned, we are interested only in solving Problems 1 and 3. Problem 1 is an evaluation problem, namely given a model and a sequence of observations, how do we compute the probability that the observed sequence was produced by the model. We can also view the problem as one of scoring how well a given model matches a given observation sequence. The latter concept is extremely useful, especially in the case when we are trying to choose among several competing models. The solution to Problem 1 allows us to choose the model which best matches the observations. As for Problem 3, it is a one in which we attempt to optimize the model parameters so as to best describe how a given observation sequence comes about. The observation sequence used to adjust the model parameters is called a training sequence since it is used to train the HMM.

A. *Solution to Problem 1*

In Problem 1, we wish to calculate the probability of the observation sequence, $O = O_1 O_2 \cdots O_T$, given the model $\lambda$, i.e. $P(O|\lambda)$. The most straightforward way of doing this is through enumerating every possible state sequence of length $T$. Considering one such fixed state sequence

$$Q = q_1 q_2 \cdots q_T \tag{2.36}$$

where $q_1$ is the initial state. The probability of the observation sequence O for the state sequence of (2.36) is

$$P(O|Q, \lambda) = \prod_{t=1}^{T} P(O_t|q_t, \lambda) \tag{2.37}$$

where we have assumed statistical independence of observations. Thus we get

$$P(O|Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdots b_{q_T}(O_T) \tag{2.38}$$

The probability of such a state sequence $Q$ can be written as

$$P(Q|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \tag{2.39}$$

The joint probability of $O$ and $Q$, i.e. the probability that $O$ and $Q$ occur simultaneously, is simply the product of (2.38) and (2.39),

$$P(O, Q|\lambda) = P(O|Q, \lambda) P(Q, \lambda). \tag{2.40}$$

The probability of $O$ given the model is obtained by summing the joint probability over all possible state sequences $q$ giving

$$\begin{aligned} P(O|\lambda) &= \sum_{allQ} P(O|Q, \lambda) P(Q|\lambda) \\ &= \sum_{q_1, q_2, \cdots, q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \cdots a_{q_{T-1} q_T} b_{q_T}(O_T) \end{aligned} \tag{2.41}$$

Calculating $P(O|\lambda)$ according to this direct definition in (2.41) involves on the order of $2TN^T$ calculations since at every $t = 1, 2, \cdots, T$, there are $N$ possible states which can be reached, i.e. there are $N^T$ possible state sequences and for each such state sequence about $2T$ calculations are required for the term in the sum of (2.41). Clearly, a more efficient method is required to solve Problem 1. The commonly used procedure is the Forward-Backward Procedure [36] [37].

B. *Solution to Problem 3*

Problem 3 is the most difficult problem among the three problems which is to determine a method to adjust the model parameters $(\mathbf{A}, \mathbf{B}, \pi)$ to maximize the probability of the observation sequence given the model. Given any finite observation sequence as training data, there is no optimal way of estimating the model parameters [35]. However, we can choose $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$ such that $P(O|\lambda)$ is locally

maximized using an iterative procedure or using gradient techniques [38]. As far as this thesis is concerned, local maximization of $P(O|\lambda)$ and training of the HMM is done using Expectation-Modification method [39].

### 2.5.3 Types of HMMs

An HMM can be of different types depending on how the states are connected and depending on the application. The two most common types of HMMs are the ergodic model and the left-right model.

i. *Ergodic or fully-connected Model*

This is the most general form of a HMM and is the case in which all the states are connected. In other words, every state can be reached from every other state in a finite number of steps. Fig. 2.3 depicts how an ergodic HMM looks like. Notice how

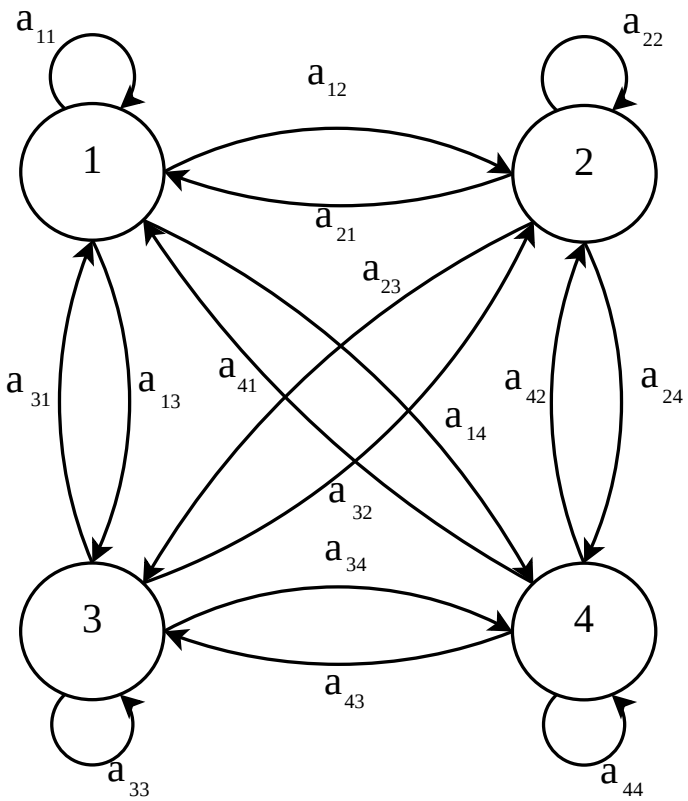


Figure 2.3: A four-state ergodic HMM

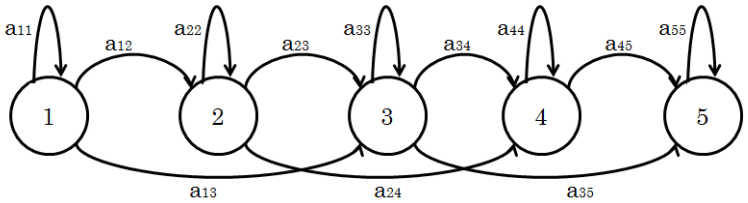


Figure 2.4: A five-state left-right HMM

this model has the property that every $a_{ij}$ coefficient exists and is of course positive. Hence, for ergodic model in Fig. 2.3, the state transition probability distribution **A** would have the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{2.42}$$

ii. *Left-right Model*

The left-right HMM is good for modelling order-constrained time-series whose properties sequentially change over time. Fig. 2.4 depicts how a left-right HMM looks like. The model gets its name from the property that as time increases the state index increases or stays the same, and as a result, the states proceed from left to right and thus the name. The fundamental property of all left-right HMMs is that the state transition coefficients have the property

$$a_{ij} = 0, \quad j < i \tag{2.43}$$

which implies that no transitions are allowed to states whose indices are lower than the current state. Moreover, the initial state probabilities have the property

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \tag{2.44}$$

In this case, the state transition probability distribution $\mathbf{A}$ of the left-right HMM in Fig. 2.4 would have the form

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix} \tag{2.45}$$

# Chapter 3

# Gesture Recognition Systems

The majority of the available literature on gesture or action recognition combines data from a 3-axis accelerometer with data from another sensing device like a biaxial gyroscope [40] or EMG sensors [10] in order to improve the system's performance and to increase the recognition accuracy. An accelerometer-based gesture recognition system using continuous Hidden Markov Models (HMMs) [41] has been developed. However, the computational complexity of statistical or generative models like HMMs is directly proportional to the number as well as the dimension of the feature vectors [41]. Therefore, one of the major challenges with HMMs is estimating the optimal number of states and thus determining the probability functions associated with the HMM. Besides, variations in gestures are not necessarily Gaussian and perhaps, other formulations may turn out to be a better fit. The most recent gesture recognition system that is solely accelerometer-based is the uWave [1].

## 3.1 uWave

uWave is a user-dependent system that supports personalized gesture recognition. Liu et al. developed uWave system on the premise that human gestures can be characterized by the time series of the forces measured by a handheld device [1]. Fig. 3.1 illustrates

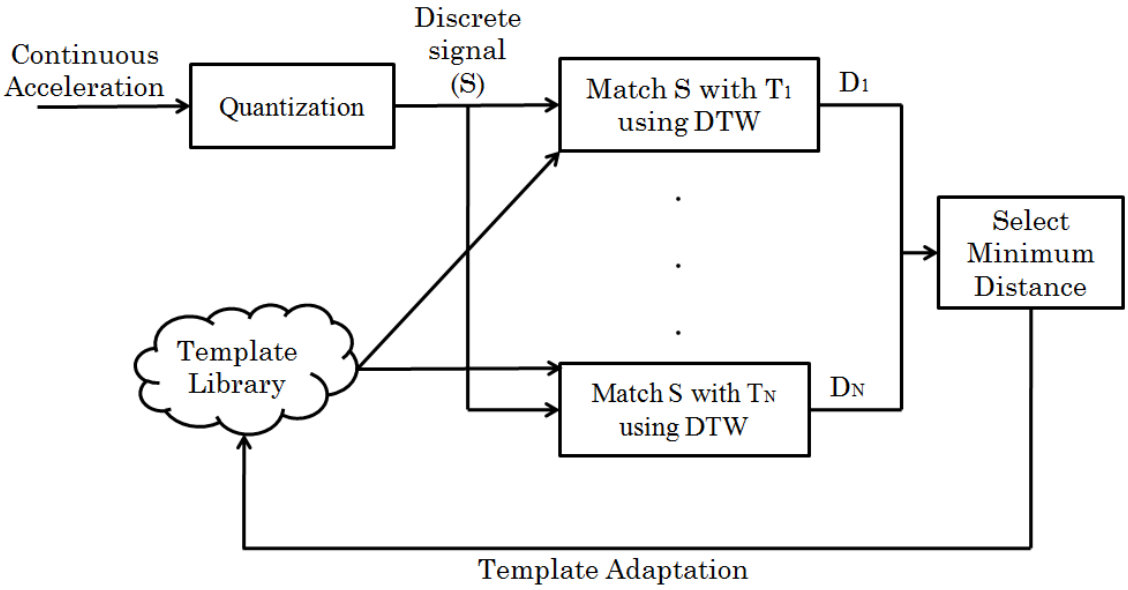


Figure 3.1: Block diagram of the uWave System from [1]

how uWave functions. The input to uWave is a time series of acceleration provided by a 3-axis accelerometer. Each time sample is a vector of three elements, corresponding to the acceleration along the three axes. uWave starts by quantizing the acceleration values into discrete values. Quantization of Library templates is also done. The quantized input time series is then compared to the library templates by dynamic time warping (DTW) and then the time series are recognized as the gesture whose template yields the lowest cost. The recognition results, confirmed by the user as correct or incorrect, can be used to adapt the existing templates to accommodate gesture variation over time.

uWave quantization consists of two steps. In the first step, the time series of the acceleration is temporally compressed by an averaging window of 50ms that moves at a 30ms step. This significantly reduces the length of the time series for DTW. The rationale behind the averaging window is that intrinsic acceleration produced by hand movement does not change erratically; and rapid changes in acceleration are often caused by noise and minor hand shaking or tilting. In the second step, the acceleration data is converted into one of 33 levels, as summarized in Table 3.1. uWave implements a non-linear quantization because most of the samples are between -$g$ and +$g$ and very few

go beyond $+2g$ or below $-2g$.

| Acceleration Data ($a$) | Converted Value |
|---|---|
| $a > 2g$ | 16 |
| $g < a < 2g$ | 11 – 15 (five levels linearly) |
| $0 < a < g$ | 1 – 10 (ten levels linearly) |
| $a = 0$ | 0 |
| $-g < a < 0$ | -1 – -10 (ten levels linearly) |
| $-2g < a < -g$ | -11 – -15 (five levels linearly) |
| $a < -2g$ | -16 |

Table 3.1: uWave Quantization Table

The core of the uWave is DTW since it is very effective in coping with limited training data and small vocabulary gestures. However, for a larger vocabulary, HMM-based methods are the chosen techniques since they are more scalable and can create better models from a large set of training data.

Liu et al. claim that there are considerable variations between gesture samples by the same user collected over different days. Ideally, uWave should adapt its templates to accommodate such time variations. Two simple schemes are proposed to adapt the templates: uWave keeps two templates generated on two different days for each gesture. It matches gesture input with both templates of each gesture and takes the smaller matching cost of the two as the matching cost between the input and the gesture. In the first scheme, if both templates for a gesture in the library are at least one day old and the input gesture is correctly recognized, the older one is replaced by the newly correctly recognized input gesture. This is referred to as *Positive Update*. The second scheme differs from the first one in that the older template is replaced with the input gesture when it is incorrectly recognized. This scheme is referred to as *Negative Update*. Positive update only requires the user to notify uWave when recognition result is incorrect whereas negative update requires the user to point out the correct gesture when a recognition error happens.

uWave is evaluated with a gesture vocabulary identified by a Nokia research [42].

uWave achieves an accuracy of 98.6% and 93.5% with and without template adaptation, respectively, for user-dependent recognition. However, uWave's database adaptation resembles continuous training and in some cases, removing an older template every other day might lead to replacing a very good representative of a gesture, which is best avoided. Although uWave proves to perform incredibly efficiently in terms of computational cost as well as recognition accuracy when compared to other systems and other statistical methods, being user-dependent limits the applications of uWave. Besides, researchers on accelerometer-based gesture recognition are envisaging a universal system that, given a dictionary of gestures, can recognize different gesture repetitions with a competitive accuracy and with minimal dependence on the user.

## 3.2 System of Discrete HMMs

Another system that is solely dependent on a single 3-axis accelerometer is the HMM-based system in [2]. The system implements a discrete HMM since the authors claim that a discrete HMM delivers reliable results for patterns with spatial and temporal variations. Fig. 3.2 depicts a block diagram of the system.

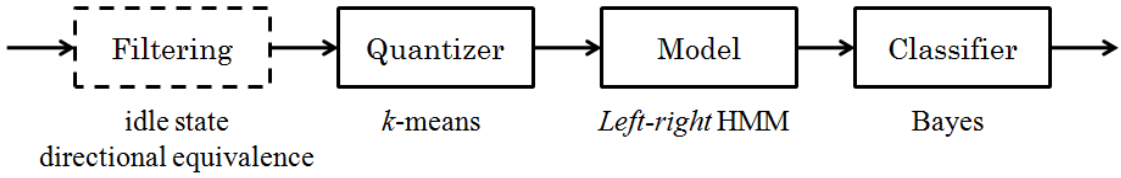


Figure 3.2: Block diagram of the HMM-based System in [2]

The system implements a preprocessing stage which constitutes two filters, an "idle state" and a "directional equivalence" filter. Schlömer et al. claim that two filters serve the purpose of reducing and simplifying the incoming acceleration data before it is forwarded for quantization and training of an HMM. The first filter is a simple threshold filter eliminating all vectors which do not contribute to the characteristic of a gesture in a significant way, i.e. all $\boldsymbol{a}$ for which $|\boldsymbol{a}| < \Delta$. For this system, the value of $\Delta$ is found

empirically to be equal to $1.2g$ where $g$ represents the acceleration due to gravity. The second filter eliminates all the vectors that are roughly equivalent to their predecessors and thus keeping only those that contribute to the characteristics of the gesture. In other words, vectors are eliminated if their components, $c \in \{x, y, z\}$ at time $t$ is very close to the vector's component at time $t$ - 1, i.e. if $|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}| \leq \epsilon$. For this system, $\epsilon$ is set to 0.2.

After the data is filtered, the system implements $k$-means in order to quantize the data according to a number of vectors set a priori. After vector quantization, an HMM is trained for each gesture. The system is developed for a gesture dictionary of five gestures, comprising a square, a circle, gesture representing roll, letter Z, and gesture representing bouncing off ground a tennis ball. For classification, a Bayesian classifier is used. The system yields an accuracy of 89.7% for both mixed-user and user-dependent recognitions.

## 3.3 Proposed Gesture Recognition System

### 3.3.1 Problem Setup

Suppose that a system consists of $N$ hand gestures and for each gesture $M$ traces are stored in a database. Gesture complexity ranges from very simple ones, as simple as the hand moving either to the right or to the left or up or down, to more complex ones such as gestures representing letters or numbers. The acceleration of the hand is used as the data to represent a gesture rather than the hand position. The acceleration of the hand is measured at different times $t$ using a single 3-axis accelerometer. Therefore, a trace of a gesture is basically a three column matrix representing the acceleration of the hand in the x-, y-, and z-directions. However, hand gestures inherently suffer from temporal variations. In other words, they differ from one person to another and even the same person cannot perfectly replicate the same gesture. This means that gesture traces can differ in duration depending on the user and the speed of the hand movement.

Consequently, traces of the same gesture are of different lengths which poses the first major challenge in developing a gesture recognition system.

Mathematically, the gesture recognition problem can be formulated as follows: The system consists of $N$ gestures, each having $M$ traces, tabulated as the following sets,

$$\begin{aligned} \mathcal{G}_1 &= \{\mathbf{G}_{1,1}, \mathbf{G}_{1,2}, \ldots, \mathbf{G}_{1,M}\}, \\ \mathcal{G}_2 &= \{\mathbf{G}_{2,1}, \mathbf{G}_{2,2}, \ldots, \mathbf{G}_{2,M}\}, \\ &\vdots \\ \mathcal{G}_N &= \{\mathbf{G}_{N,1}, \mathbf{G}_{N,2}, \ldots, \mathbf{G}_{N,M}\}. \end{aligned} \tag{3.1}$$

Each $\mathbf{G}_{i,j}$ is a $l_{i,j}\times 3$ matrix, where each column represents the acceleration in the x-, y-, or z-direction. Notice that $l_{i,j}$ is different even for traces of the same gesture $\mathcal{G}_i$ since traces of the same gesture can have different durations and thus different number of rows.

The gesture database (3.1) is produced in an off-line procedure and stored for later use, which constitutes the training stage. In a test stage, a user moves his/her accelerometer-equipped device, such as a smart phone or a wiimote, to signal a particular gesture from the above database in (3.1). The accelerometer readings are formed into an $l_y\times 3$ matrix $\mathbf{Y}$. Again, note that $l_y$ may not be equal to any $l_{i,j}$. The objective of a gesture recognition system is to find out which gesture is intended by the user.

Fig. 3.3 depicts the general overview of the proposed gesture recognition system. Notice that the block diagram implicitly represents a two-stage system: the first stage being the training stage is represented by the top part of the block diagram, whereas the second stage being the testing stage is represented by the bottom part of the block diagram.

The training stage comprises two parts. A sliding window, which acts as a moving average filter, is applied to the acquired data to remove any noise that might have been accumulated, which is due to internal sampling, or accelerometer calibration or sensitivity, or hand shaking during gesture acquisition. The smoothing step is followed by a clustering

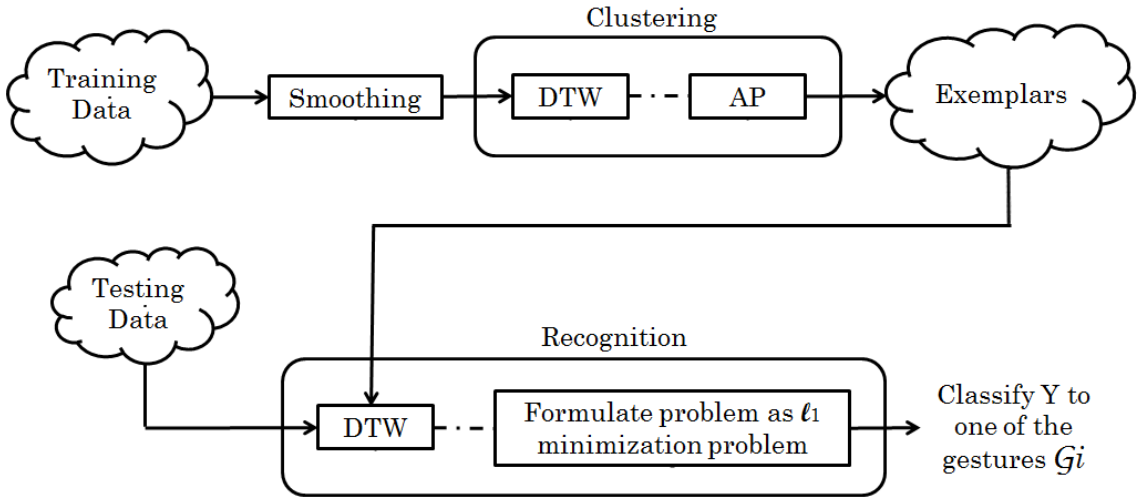


Figure 3.3: General Overview of the Gesture Recognition System

process which is broken into two sub-blocks. The first clustering sub-block deals with the unequal durations of the gesture traces $\mathbf{G}_{i,j}$. This sub-block uses DTW to compute a measure of similarity between vectors of unequal lengths. The measure of similarity is then used in the AP sub-block to decompose the training data into multiple clusters. Clustering in essence represents the core of the training stage. Members of the same cluster should share the same characteristics; coming from the same gesture is the most desirable one. Each cluster is represented by one of its members called an "exemplar". So, the output of the clustering stage, and in turn the training stage, is a set of exemplars $\mathscr{E}$ each representing a cluster of gesture traces. The details of each sub-block will be discussed in the following section.

### 3.3.2 Training Stage

Recall that gesture traces suffer from inherent temporal variations, and therefore the conventional Euclidean distance is not applicable as a similarity measure between the gesture traces. Consequently, in our gesture recognition system, we resort to dynamic time warping to compute the similarities between the different gesture traces.

In the proposed 3-axis accelerometer gesture recognition system, since each gesture

trace is defined by three acceleration waveforms, the similarity cost between gesture trace $\mathbf{G}_i$ of size $n\times3$ and gesture trace $\mathbf{G}_j$ of size $m\times3$ is computed as:

$$\mathrm{DTW}(\mathbf{G}_i, \mathbf{G}_j) = \sqrt{D^2_{n,m}(x) + D^2_{n,m}(y) + D^2_{n,m}(z)} \tag{3.2}$$

where $D_{n,m}(x)$, $D_{n,m}(y)$, $D_{n,m}(z)$ are the DTW costs computed between the traces in the x, y, and z axes respectively.

Importing the definitions of the similarity and the availability messages from Section 2.2 and applying them to our proposed gesture recognition system, the "responsibility" message assists in deciding which traces are exemplars, and the "availability" message assists in deciding which cluster a trace belongs to. The responsibility message is given by

$$r(i,j) = s(i,j) - \max_{j' s.t. j' \neq j} \left\{ a(i,j') + s(i,j') \right\} \tag{3.3}$$

where $i \neq j$, and $s(i,j)$ is the pairwise similarity that indicates how well the trace $\mathbf{G}_j$ is suited to be the exemplar for the trace $\mathbf{G}_i$ and is defined as

$$s(i,j) = \mathrm{DTW}(\mathbf{G}_i, \mathbf{G}_j) \ \ \forall \ \ i,j \in \{1,2,...,L\} \tag{3.4}$$

where $L$ is the total number of gesture traces, and the availability message is given by

$$a(i,j) = \min \left\{ 0, r(j,j) + \sum_{i' s.t. i' \neq i,j} \max\{0, r(i',j)\} \right\} \tag{3.5}$$

In addition to the measure of similarity, AP takes as input a set of real numbers, known as self-similarity or *preference* ($p$) for each gesture trace, so that traces with larger values of $p$ are more likely to be chosen as exemplars. For the proposed gesture recognition system, the self-similarity $p$ is proportional to the median of the input similarities, that

is

$$p = \beta * \text{median}\{s(i,j),\ \forall\ i,j \in \{1,2,...,L\}\} \tag{3.6}$$

where $\beta$ is a constant that controls the number of clusters to be generated in an inversely proportional manner. In other words, as the value of $\beta$ decreases, more clusters will be generated.

AP is chosen as the clustering technique since it does not operate on feature vectors or raw data but rather operates on a matrix of similarities between data points. This configuration of AP eliminates the requirement of forcing all gesture traces to be of the same length or generating feature vectors of equal lengths as is the case in [40] [10] [42]. AP can generate better clusters, compared to other clustering techniques like $K$-means clustering, because of its initialization-independent property [21].

The output of AP is a set of exemplars $\mathscr{E}$ for the $N$ gestures in our system, such that

$$\mathcal{E} = \{\mathbf{E}_1, \mathbf{E}_2, \ldots, \mathbf{E}_H\} \tag{3.7}$$

where $H \geq N$. Notice that the number of exemplars $H$ obtained is greater than or equal to the number of gestures $N$. The reason is due to the fact that the gesture traces are collected from different subjects. Consequently, the number of exemplars per gesture, $H_i$, satisfies the inequality

$$1 \leq H_i \leq P \quad \forall\ i \in \{1,2,\ldots,N\} \tag{3.8}$$

where $P$ is the number of subjects included in training the system. Figure 3.4 shows a complete block diagram of the training stage.

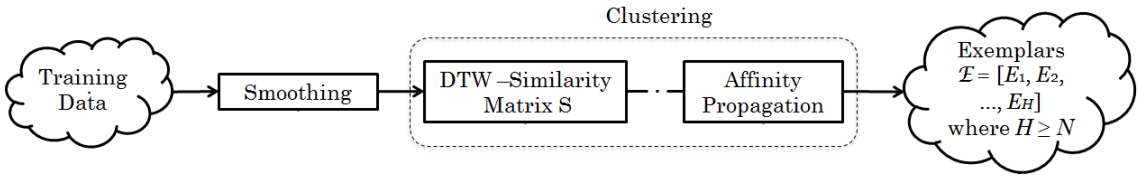


Figure 3.4: Block Diagram of Training Stage

### 3.3.3 Testing Stage

In order to recognize an unknown gesture trace $\mathbf{Y}$, it is intuitive to compare it to the set of exemplars in $\mathcal{E}$ and classify $\mathbf{Y}$ to the gesture whose exemplar gives the lowest cost. However, since our clustering algorithm does not yield a unique exemplar for each gesture, this approach does not suffice in yielding a high recognition accuracy. We make the following observations on the clustering technique used above.

First, we note that, although not observed in our simulations, the affinity propagation technique does not guarantee that all members of a cluster and its exemplar are traces of the same gesture. The problem becomes more significant when gesture traces from different subjects are combined in the same database.

Second, although an exemplar is a representative of its cluster, it cannot be used to detect the corresponding gesture of a testing trace due to the fact that a unique exemplar cannot be extracted per gesture. However, exemplars are useful in removing outliers, and reducing the size of the search space, hence reducing the computational complexity.

Fig. 3.5 shows a complete block diagram of the testing stage. The proposed recognizer comprises mainly of two steps. In the first step, the unknown gesture trace $\mathbf{Y}$ is compared to the set of exemplars obtained in the training stage to find those that are closest to $\mathbf{Y}$. Then, a $\mathcal{R}$ is created by grouping together all the members of the clusters whose exemplars are chosen to be closer to $\mathbf{Y}$. Further processing of the data is very difficult since the different gesture trace duration still poses problem.

One solution is to project all the traces onto the same lower dimensional subspace and thus solve the problem of different durations and simultaneously reduce the compu-

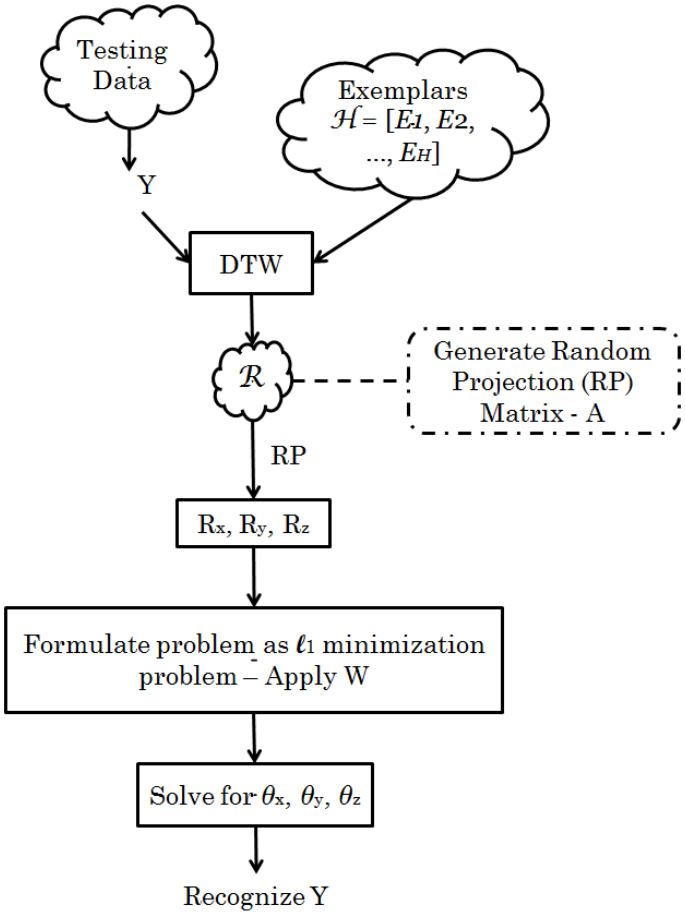


Figure 3.5: Block Diagram of Testing Stage

tational cost. This proposition is motivated by the premise that, as seen in Fig. 1.3, the defined hand gestures appear to be sparse since the hand follows a smooth trajectory while performing a gesture. Therefore, gesture traces can be represented using fewer samples as per the theory of compressive sensing as explained in Section 2.3.

Therefore, one solution to overcome the different gesture repetitions sizes is to project all the repetitions onto the same lower subspace. An ideal dimensionality reduction technique has the capability of efficiently reducing the data into lower-dimensional subspace while preserving the properties of the original data. Random projection is the approach sought to project the repetitions onto the same lower subspace. The proposed gesture recognition system will be tested against both forms of the sampling or projection matrix $\mathbf{A}$: the Gaussian distribution and the distribution in (2.27).

In the second step, the set $\mathcal{R}$ is then converted into three matrices and the random projection matrix is constructed. After that, the data is projected onto the same lower dimensional subspace. Then the problem is formulated as $\ell_1$ minimization problem and is solved to estimate the three sparse vectors $\hat{\boldsymbol{\theta}}_x$, $\hat{\boldsymbol{\theta}}_y$, and $\hat{\boldsymbol{\theta}}$ which are then combined to recognize the unknown trace $\mathbf{Y}$.

For recognition, the following approach is followed. As per earlier notation, let $\mathbf{Y}$ represent an unknown gesture trace to be recognized, which is a matrix of size $l_y \times 3$, and let $\boldsymbol{y}$ denote one of the columns of $\mathbf{Y}$ and let $\mathcal{R}$ be the set of all traces that are of close resemblance to $\mathbf{Y}$. This resemblance is determined by computing the DTW cost between $\mathbf{Y}$ and every exemplar $\mathbf{E}_j \in \mathscr{E}$, and choosing the clusters whose DTW cost is below a certain threshold $\alpha$. In other words,

$$\mathcal{R} = \{\mathbf{C}_j \mid \forall\, j : \mathrm{E}_j \in \mathscr{E} \ \textit{and}\ \mathrm{DTW}(\mathrm{E}_j, \mathrm{Y}) < \alpha\} \tag{3.9}$$

where $\mathbf{C}_j$ is any member of the $j$th cluster with the exemplar $\mathbf{E}_j$. By empirical exami-

nation, we have found that the threshold

$$\alpha = 2 \cdot \min\{\mathrm{DTW}(\mathbf{E}_j, \mathbf{Y}),\ \forall\, \mathbf{E}_j \in \mathscr{E}\} \tag{3.10}$$

gives the best results.

In order to proceed with the recognition process, the set $\mathcal{R}$ is converted into three matrices by forcing all traces as well as the unknown gesture trace $\mathbf{Y}$ to be in the same space. This is done by finding the maximum length $l_{max}$ to be

$$l_{max} = \max\{l_y, l_1, \ldots, l_L\} \tag{3.11}$$

where $L$ is the total number of traces in $\mathcal{R}$ and $l_1, \ldots, l_L$ represent their lengths, i.e. the number of rows in each trace. After $l_{max}$ is found, all the traces including the unknown gesture trace $\mathbf{Y}$, which are shorter than $l_{max}$ are padded by zeros forcing them to be of length $l_{max}$. In other words, all traces are transformed to the largest subspace by assuming that the shorter traces have zero components in the higher subspaces. Zero padding can take any possible form, i.e. adding zeros to the beginning of the trace, adding zeros in between the trace samples, or adding zeros to the end of the trace. This is due to the fact that the random projection matrix $\mathbf{A}$ satisfies the RIP condition and therefore, it makes no difference which columns of $\mathbf{A}$ are chosen to compress the trace [25]. However, for simplicity of application, we pad zeros to the end of the traces, and in this case,

$$\mathbf{R}_x = [\boldsymbol{r}_1^x, \boldsymbol{r}_2^x, \cdots, \boldsymbol{r}_L^x] = \begin{bmatrix} r_{1,1}^x & r_{2,1}^x & \cdots & r_{L,1}^x \\ r_{1,2}^x & r_{2,2}^x & \cdots & r_{L,2}^x \\ \vdots & \vdots & \ddots & \vdots \\ r_{1,l_1}^x & r_{2,l_2}^x & \cdots & r_{L,l_L}^x \\ 0_1 & 0_2 & \cdots & 0_L \end{bmatrix} \tag{3.12}$$

and,

$$\boldsymbol{y}_x = \begin{bmatrix} y_1^x \\ y_2^x \\ \vdots \\ y_{l_y} \\ 0_y \end{bmatrix} \tag{3.13}$$

where $\mathbf{R}_x$ is a matrix whose columns represent the x-components of the padded traces, $\boldsymbol{y}_x$ is the x-component of the padded unknown gesture trace, and $0_i$ and $0_y$ are zero vectors of length $(l_{max} - l_i)$ and $(l_{max} - l_y)$ respectively. $\mathbf{R}_y$, $\mathbf{R}_z$, $\boldsymbol{y}_y$, and $\boldsymbol{y}_z$ are constructed in a similar manner.

In order to project the data onto the lower dimensional subspace, the projection matrix $\mathbf{A}$ is constructed based on the distributions defined earlier and would be of size $l_k \times l_{max}$, where $l_k$ is the dimension of the new common lower dimensional subspace. According to Fig. 1.3, gesture waveforms are smooth curves, and one of the transformations which would give a sparse representation of the waveforms is the Fourier transform. So, for a sparse sequence $\boldsymbol{r}$, let $\tilde{\boldsymbol{r}}$ and $k_{\boldsymbol{r}}$ denote the Fourier transform and the sparsity level of the sequence $\boldsymbol{r}$ respectively. Moreover, let $r_m$ denote the maximum magnitude in $\tilde{\boldsymbol{r}}$. The sparsity level $k_{\boldsymbol{r}}$ of $\boldsymbol{r}$ is defined as

$$k_{\boldsymbol{r}} \geq K \cdot B_\gamma \tag{3.14}$$

where $K$ is a constant and $B_\gamma$ is the number of samples in $\tilde{\boldsymbol{r}}$ that are greater than a threshold $\gamma$ defined as

$$\gamma = c \cdot r_m \tag{3.15}$$

where $c$ is a constant $\in (0, 1)$ to preserve only the significant samples. In practice, $K$ can be either 3 or 4 making the sparsity level $k_{\boldsymbol{r}}$ three or four times $B_\gamma$ [22]. Accordingly,

the Fourier transform of a trace $\mathbf{R}$ is defined as

$$\tilde{\mathbf{R}} = [\tilde{\boldsymbol{r}}_x \;\; \tilde{\boldsymbol{r}}_y \;\; \tilde{\boldsymbol{r}}_z] \tag{3.16}$$

The sparsity of $\mathbf{R}$ is then given by

$$k_{\mathbf{R}} = \max\{k_{\boldsymbol{r}_x}, k_{\boldsymbol{r}_y}, k_{\boldsymbol{r}_z}\} \tag{3.17}$$

The sparsity level $k_{\mathbf{R}}$ is computed for each trace in (3.1) and stored in the database as well. Consequently,

$$l_k = \max\{k_{\mathbf{R}_i};\;\; \forall\, i \in \{1, 2, \ldots, L\}\} \tag{3.18}$$

After $\mathbf{A}$ is constructed, the data in the x-direction is projected as

$$\overline{\mathbf{R}}_x = \mathbf{A}\mathbf{R}_x = [\mathbf{A}\boldsymbol{r}_1^x, \mathbf{A}\boldsymbol{r}_2^x, \cdots, \mathbf{A}\boldsymbol{r}_L^x] \tag{3.19}$$

and

$$\overline{\boldsymbol{y}}_x = \mathbf{A}\boldsymbol{y}_x \tag{3.20}$$

where $\overline{\mathbf{R}}_x$, represents the projected data in the x-direction onto the new subspace and $\overline{\boldsymbol{y}}_x$ represents the projected x-component of the unknown gesture trace.

The relationship between $\overline{\mathbf{R}}_x$ and $\overline{\boldsymbol{y}}_x$ can be formulated as

$$\overline{\boldsymbol{y}}_x = \overline{\mathbf{R}}_x \boldsymbol{\theta}_x \tag{3.21}$$

where $\boldsymbol{\theta}_x$ is theoretically a 1-sparse $L \times 1$ vector whose elements are all zeros except $\boldsymbol{\theta}_x(n) = 1$, such that $\boldsymbol{r}_n^x$ best resembles $\boldsymbol{y}_x$. Namely,

$$\boldsymbol{\theta}_x = [0, , 0, 1, 0, , 0]^T \tag{3.22}$$

where $T$ denotes transposition. However, gesture traces suffer from inherent temporal variations and therefore, the above ideal scenario of having a perfect match to the unknown gesture trace is impossible and therefore, the problem can be reformulated as

$$\overline{\boldsymbol{y}}_x = \overline{\mathbf{R}}_x \boldsymbol{\theta}_x + \boldsymbol{\varepsilon}_x \tag{3.23}$$

where $\boldsymbol{\varepsilon}_x$ is the measurement noise.

Using the same formulation as in [43], we introduce the preprocessor $\mathbf{W}$, which is defined as

$$\mathbf{W}_x = \mathbf{Q}_x \overline{\mathbf{R}}_x{}^{\dagger} \tag{3.24}$$

where $\mathbf{Q}_x = \text{orth}(\overline{\mathbf{R}}_x^T)^T$, and $\text{orth}(\overline{\mathbf{R}}_x)$ is an orthogonal basis for the range of $\overline{\mathbf{R}}_x$, and $\overline{\mathbf{R}}_x^{\dagger}$ is the pseudo-inverse of the matrix $\overline{\mathbf{R}}_x$. The gesture recognition problem takes on a new formulation as,

$$\boldsymbol{h}_x = \mathbf{W}_x \overline{\boldsymbol{y}}_x = \mathbf{Q}_x \boldsymbol{\theta}_x + \boldsymbol{\varepsilon}_x^{'} \tag{3.25}$$

where $\boldsymbol{\varepsilon}_x^{'} = \mathbf{W}_x \boldsymbol{\varepsilon}_x$. $\boldsymbol{\theta}_x$ can be well recovered from $\boldsymbol{h}_x$ with a high probability through the following $\ell_1$-minimization formulation:

$$\hat{\boldsymbol{\theta}}_x = \arg\min \parallel \boldsymbol{\theta}_x \parallel_1, \;\; s.t. \; \boldsymbol{h}_x = \mathbf{Q}_x \boldsymbol{\theta}_x + \boldsymbol{\varepsilon}_x^{'} \tag{3.26}$$

This represents recognition of the unknown trace based on data in the x-direction only. $\boldsymbol{\theta}_y$ and $\boldsymbol{\theta}_z$ are solved for using the same approach.

In order to recognize the unknown gesture trace, the three $\hat{\boldsymbol{\theta}}_x, \hat{\boldsymbol{\theta}}_y, \hat{\boldsymbol{\theta}}_z$ vectors are combined together in the following manner,

$$\hat{\boldsymbol{\theta}}_{eq} = \hat{\boldsymbol{\theta}}_x^2 + \hat{\boldsymbol{\theta}}_y^2 + \hat{\boldsymbol{\theta}}_z^2 \tag{3.27}$$

The unknown gesture trace is then recognized as the gesture to which the trace $\mathbf{R}_i$ belongs

such that $\hat{\boldsymbol{\theta}}_{eq}(i)$ is maximum.

# Chapter 4

# Implementation Results

## 4.1 Distance Calculation System

A simple distance estimating system using a single 3-axis accelerometer is developed. For training, five individuals are asked to walk a distance of 48 ft while carrying an accelerometer. The chosen individuals vary in height causing them to have different stride sizes. The path the individuals follow consists of turns as well as U-turns to verify how well the system performs in such setting. Each person is asked to repeat the experiment three times, and each time the number of steps taken is noted. The collected data is used to train the system by estimating the constant $K$ in (1.2). The system is tested by asking two of the individuals to walk a distance of 44 ft and a distance of 22 ft and the estimated distance are compared to the actual distances.

Table 4.1 shows a summary of the performance of the system with the percentage error. According to the obtained results, the system estimates the walked distance within $\pm 8\%$ of the actual distance which is a very promising result. In conclusion, 3-axis accelerometer can be of great significance and their embodiment with many personal electronic devices opens a gate to numerous applications that need to be explored.

| User | Actual Distance (ft) | Calculated Distance (ft) | Percentage Error (%) |
|---|---|---|---|
| Individual 1 | 44 | 45.98 | 4.5 |
| | 22 | 23.72 | 7.8 |
| Individual 2 | 44 | 42.25 | 3.9 |
| | 22 | 20.99 | 4.6 |

Table 4.1: Performance of distance estimating system

## 4.2 Gesture Recognition System

The acceleration data corresponding to the different gestures is collected using a wiimote, which has a built-in 3-axis accelerometer. A dictionary of 18 gestures is created as shown in Fig. 4.1. To the best o our knowledge, our dictionary of gestures is the largest in published studies for accelerometer-based gesture recognition. The defined gestures are not limited to one plane only as is the case in other studies [1] [42], but span the two planes: $XZ$ and $YZ$ planes. The dictionary contains a variety of gestures ranging from the simple right, left, up, down gestures to more complex gestures resembling letters and numbers. This definition of gestures is to increase the robustness of the gesture recognition system.

The database consists of 3,780 traces and is built by acquiring gestures from 7 subjects (2 females and 5 males) using the wiimote. Each subject is asked to repeat each gesture 30 times resulting in a total of 540 traces for all gestures per participant or a total of 210 traces from all participants per gesture. A gesture acceleration waveforms from the same person can differ drastically if the tilting angle of the accelerometer is large. Therefore, all participants are asked to try their best to perform the gestures without any, or with minimal, tilting of the remote.

For system evaluation, the database is split into two datasets: a training set and a testing set. The training dataset is generated by choosing traces from three users (2 males and 1 female) out of the seven users, i.e. $P = 3$. Specifically, 5 traces are ***randomly*** chosen for each gesture from each of the three users resulting in a total of 15 traces per gesture, i.e. $M = 15$. The testing dataset comprises all the remaining traces from the

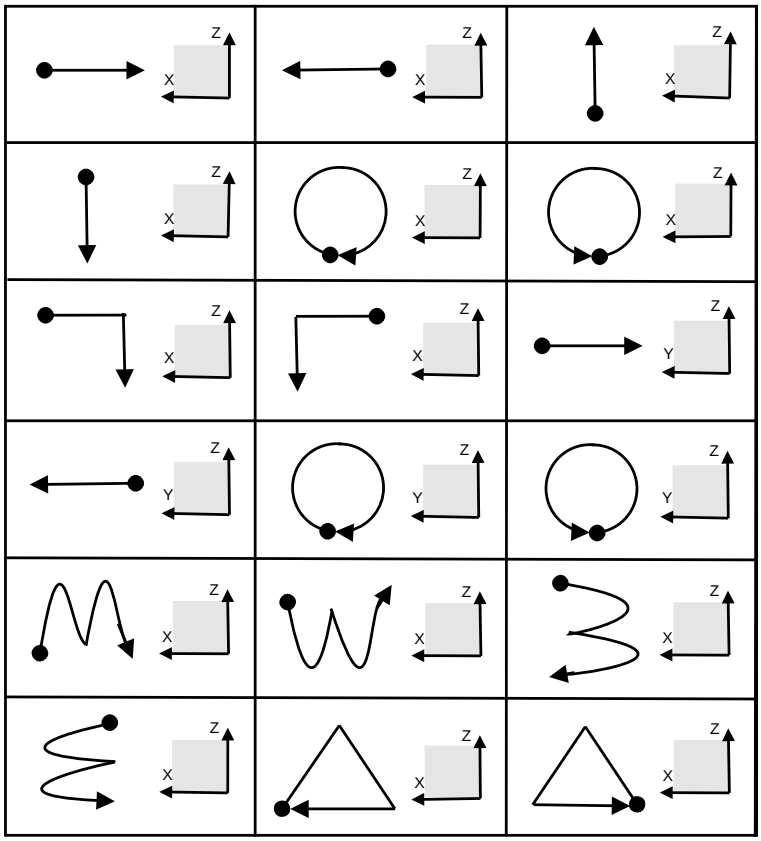


Figure 4.1: The dictionary of 18 gestures

three users plus the entire set of traces from the remaining four users (3 males and 1 female). Simulations are run for $N = \{8, 10, 12, 14, 16, 18\}$ gestures. The system's performance is compared to Hidden Markov Models (HMMs), the system in [1], and furthermore to a system in literature developed using HMMs [2]. Fig. 4.2 and Fig. 4.3 show the system's performance in terms of recognition accuracy against the number of gestures for a projection matrix $\mathbf{A}$, formed with a Gaussian distribution and the sparse distribution in (2.27) respectively, and is compared to HMMs.

In order to develop the system of HMMs, the system is set up in an identical manner to [41]: a left to right HMM with a continuous gaussian distribution is used to model each gesture. The output distributions are assumed to have diagonal covariance matrices. Consequently, each HMM can be described as an $8m$-parameter model, where $m$ is the number of states. The eight parameters comprise the two state transition probabilities, and one Gaussian output distribution which constitutes a mean vector $\boldsymbol{\mu} \in \mathcal{R}^3$ and the three diagonal elements of the covariance matrix. For comparison, simulations are run with $m = 10$.

Fig. 4.2 and Fig. 4.3 show that the system's performance is almost identical for both definitions of the projection matrix $\mathbf{A}$ which confirms that the sparse distribution in (2.27) is a very good approximation of the Gaussian distribution. The system yields a very competitive performance for a system of 8 gestures, giving a recognition accuracy of 96.84%. The dashed lines show the system's performance only on traces in the testing dataset from the three subjects whose data is used in training the system. In other words, this type of recognition can be referred to as mixed-user recognition. The solid lines show the system's performance on the entire testing dataset which includes traces from all the seven subjects, and this type of recognition is referred to as user-independent recognition. As shown, the system greatly outperforms the simple HMM-system for all dictionary sizes.

The proposed system results in great computational savings in terms of training. Fig.

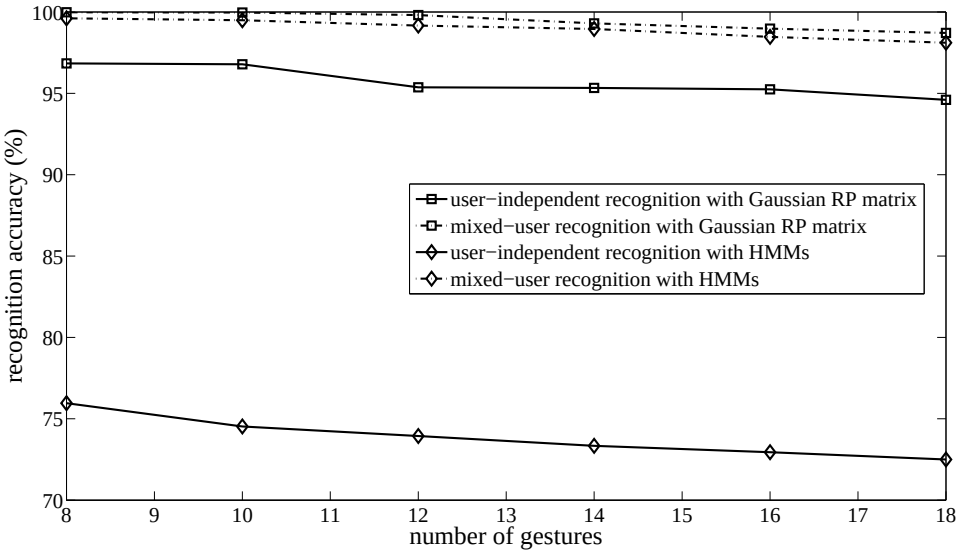


Figure 4.2: System's performance against the number of gestures using a Gaussian random projection matrix compared to HMM

4.4 shows the average time taken to train the system for different dictionary sizes. The dark bars in Fig. 4.4 pertain to the proposed system whereas the lighter bars pertain to the system of HMMs. The proposed system takes much less time to train the system for all dictionary sizes compared to the system of HMMs. This is due to the nature of the Expectation-Maximization (EM) algorithm adopted by HMM for training. EM is an iterative method which alternates between performing an expectation (E) step and a maximization (M) step to estimate the parameters that describe the distribution of the model. Therefore, depending on the starting points, the algorithm might converge to a local maximum and never reach the global maximum. Affinity propagation, on the other hand, is immune to the choice of the starting point, since it considers all data points as exemplars. Exemplars are chosen based on the exchange of the responsibility and the availability messages between the data points. In other words, affinity propagation implements a recursive algorithm to create the exemplars, not an iterative algorithm as is the case with HMM.

As for testing, Fig. 4.5 shows the average time taken to recognize an unknown gesture

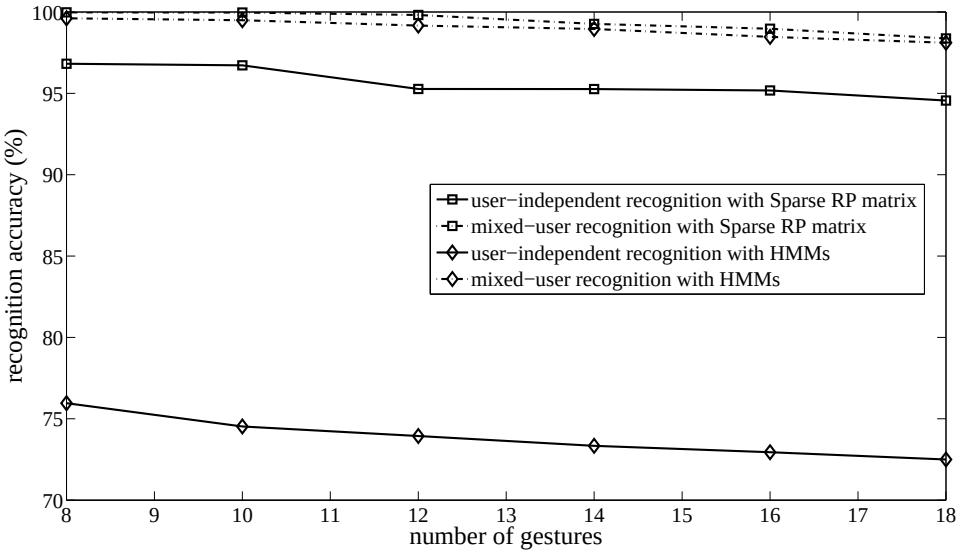


Figure 4.3: System's performance against the number of gestures using a sparse random projection matrix compared to HMM

trace. The system of HMMs executes an unknown gesture trace recognition in an average of 0.08 seconds whereas the proposed system recognizes an unknown gesture trace in 0.18 seconds. Although the system of HMMs takes about half the time taken by the proposed system for an unknown trace recognition, the difference is not noticeable since both times are less than one-twentieth of a second. Besides, the unnoticeable saving in the testing computational cost is minor at the expense of the tremendous outperformance of the proposed system to the system of HMMs for user-independent recognition.

In order to give a deeper insight into the system's performance, Fig. 4.6 depicts the cumulative density functions (cdfs) of the system's performance using a Gaussian projection matrix for dictionary sizes of 8, 10, 16, and 18 gestures respectively. The cdfs are generated by running the code 1000 times and recording the system's recognition accuracy for each run. According to Fig. 4.6, an increase in the dictionary size from 8 gestures to 18 gestures results in a reduction of 2% in the system's performance. This shows that the system is robust to large dictionary sizes unlike the case with the uWave system. For uWave system, the dictionary size is limited to 8 gestures since the core of

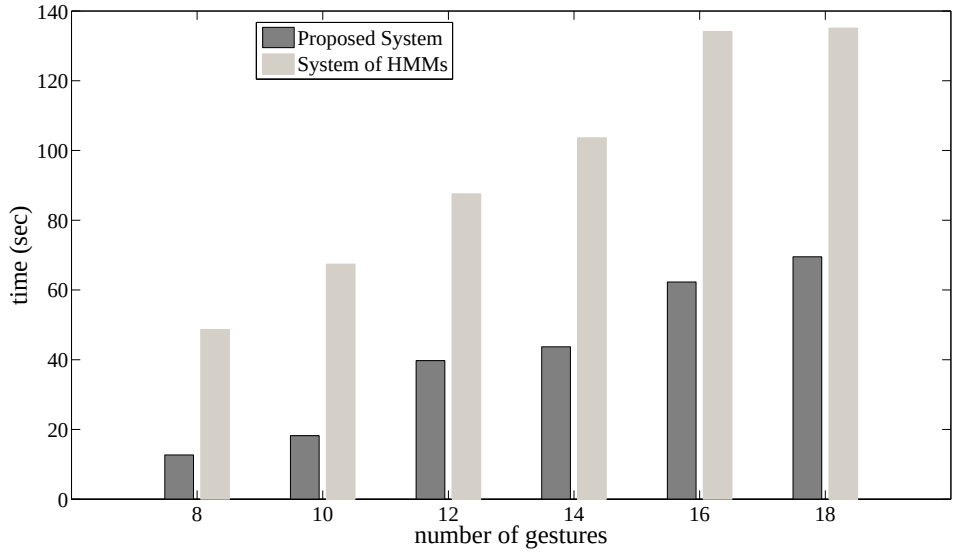


Figure 4.4: Comparison of training computational cost between proposed system and system of HMMs

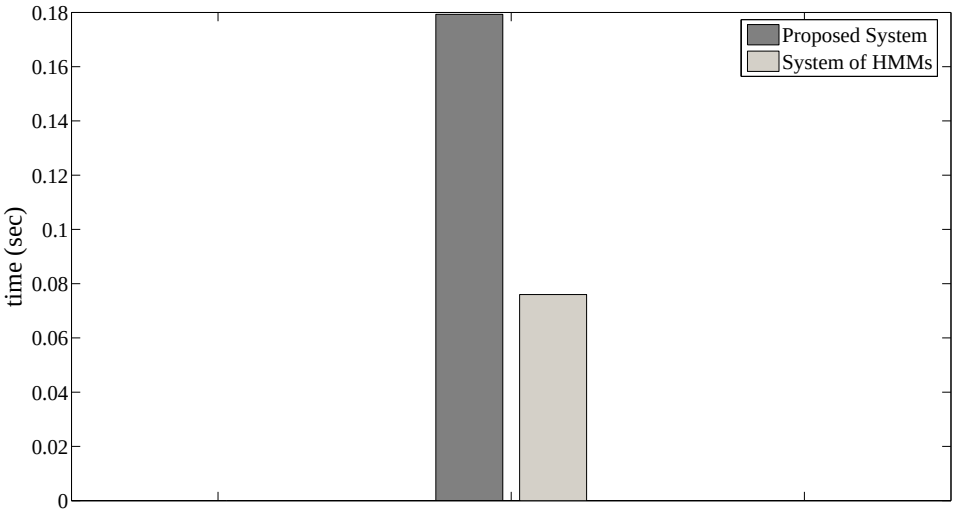


Figure 4.5: Comparison of testing computational cost between proposed system and system of HMMs

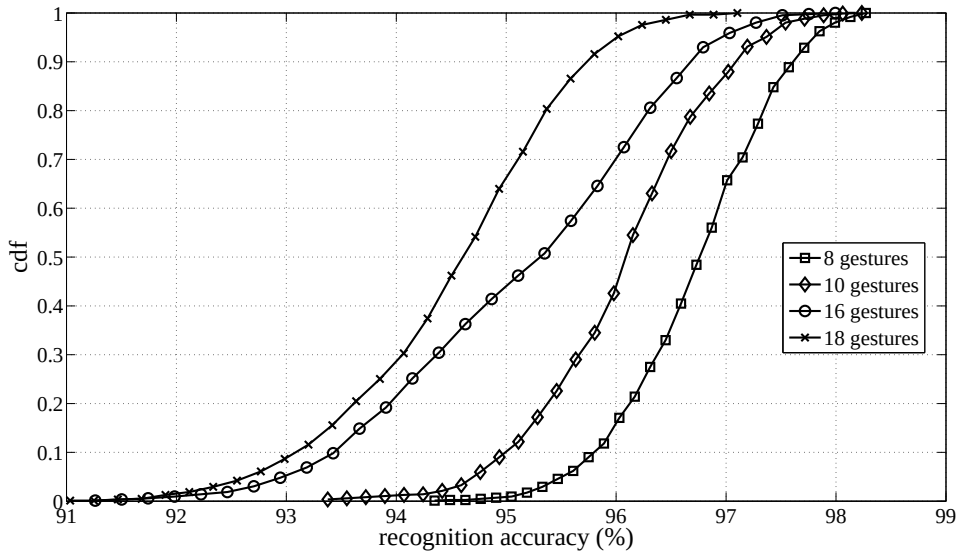


Figure 4.6: Cdfs of the system's performance for $N = 8$, 10, 16, and 18 gestures using a Gaussian random projection matrix

the system is dynamic time warping which copes effectively with limited training data and smaller dictionaries of gestures.

For further comparison between the proposed system and the other systems, Table 1 shows a comparison of performance between the proposed system, HMMs, uWave, and the system in [2]. From Table 1, we can deduce that our proposed system outperforms uWave system in [1] for user-dependent recognition. The system yields a perfect recognition accuracy for a dictionary of 8 gestures using the gestures defined compared to an accuracy of 98.4% by uWave. Finally, our system outperforms the system in [2] which yields an average accuracy of 90% for a dictionary size of five gestures only compared to our dictionary size of 18 gestures.

Table 1: Comparison of Performance of proposed system, HMMs, uWave, and system in [2]

| Technique | no. of gestures | Accuracy(%) | | |
|---|---|---|---|---|
| | | User-Dependent | Mixed-User | User-Independent |
| Proposed System | 8 - 18 | 100 - 99.81 | 99.98 - 98.71 | 96.84 - 94.60 |
| HMMs | 8 - 18 | 99.97 - 99.54 | 99.61 - 98.11 | 75.96 - 71.50 |
| uWave | 8 | 98.6 | - | 75.4 |
| System in [2] | 5 | 89.7 | 89.7 | - |

# Chapter 5

# Conclusions and Future Work

A primary goal of virtual environments is to provide natural, efficient, and flexible interaction between human and computer. Gestures as input modality can help meet these requirements. Human gestures which are natural can be efficient and powerful when compared to other interaction modes. Gestures are expressive and meaningful body motions which involve physical movements of the fingers, hands, arms, face, or head with the intent to convey information or interact with the environment. Gestures can be static, where the user assumes a certain pose or configuration, or dynamic, defined by movement.

Gesture recognition is the process by which gestures made by the user are made known to the system. To support gesture recognition, body parts position and movement must be tracked and interpreted in order to recognize meaningful gestures. Numerous sensing devices such as magnetic field trackers, instrumented gloves, and datasuits, which are attached to the user, have been utilized to develop gesture recognition systems as well as cameras and other computer vision techniques. Each sensing technology differs along several dimensions, including accuracy, resolution, latency, range of motion, user comfort, and cost. Other gesture recognition systems have been developed by integrating multiple sensors with the intent of improving the system's performance and increasing

the recognition accuracy.

Motivated by the proliferation of the accelerometers on personal electronic devices such as Apple iPhone, iPad, and other smart phones, this work presented a novel gesture recognition system based solely on data from a single 3-axis accelerometer. The system employs dynamic time warping and affinity propagation algorithms for efficient training. Dynamic time warping is used to calculate a cost of similarity between the gesture traces after the conventional methods failed due to the different gesture trace durations. Affinity propagation operates on the costs of similarities between the gesture traces and divides the training data into different clusters each represented by an exemplar.

In the testing phase, the unknown trace is compared to the exemplars induced by affinity propagation to select a subset of coordinate traces. The sparse nature of the gesture traces is exploited to project candidate traces and the unknown gesture trace onto the same lower-dimensional subspace. Projection is implemented using two definitions of a projection matrix: a matrix whose entities are independent realizations of Gaussian random variables and a matrix whose entities are independent realizations of Bernoulli random variables.

The system is tested on a dictionary of 18 gestures whose database contains over 3,700 traces collected from 7 subjects. The system's performance is evaluated by comparing it to the performance of a system of continuous HMMs, a system of discrete HMMs, and uWave. Systems of HMMs are used for comparison since HMM is a powerful technique which manifested itself in virtual environments and recognition problems. The system achieves almost perfect recognition for user-dependent recognition and extremely competitive accuracies for mixed-user and user-independent recognitions. Compared to HMMs, the system outperforms a system of continuous HMMs as well as a system of discrete HMMs for all dictionary sizes. Furthermore, the system requires much less time to train the system compared to HMMs. Although uWave, the most recent gesture recognition system that is solely accelerometer-based, is primarily user-dependent, our proposed

system outperforms uWave for user-dependent recognition. In summary, our proposed gesture recognition system provides an outstanding performance in terms of recognition accuracy and computational complexity when compared to other systems in literature.

## 5.1 Future Work

Future work involves implementing the proposed gesture recognition system on a smart phone or any other personal device with a built-in accelerometer. The system proves a very competitive performance computationally and in terms of recognition accuracy by running the code in Matlab on a computer. However, it is more reasonable to validate the system's novelty and performance by testing how it performs in real life. This entails running the code on a different platform which means converting the code to a different programming language such as JAVA, C, C++, or any language supported by the phone.

Another interesting topic for future work is to extend the proposed system to incorporate gesture spotting. In our proposed system, the starting point of a gesture trace was marked by pressing and holding the 'trigger' button or 'B' button on the bottom of the remote. This way of acquiring gesture traces by assuming known starting and end points is not realistic. Therefore, a more realistic scenario would be to detect meaningful gesture traces from a stream of hand movements and recognizing the gesture traces accordingly.

One more interesting topic to research is the problem of tilting. As mentioned earlier, tilting of the remote can lead to erroneous recognition if not taken into account. Therefore, in our proposed system, subjects were requested to hold the remote in a natural way while performing the gestures and to avoid any tilting of the remote as much as possible. However, this way of holding the remote can result in some inconvenience to users of the system. Consequently, a system which is immune to tilting of the accelerometer is definitely a desirable one.

# Bibliography

[1] J. Liu, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 657 – 675, 2009, perCom 2009.

[2] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a Wii controller," *TEI'08 - Second International Conference on Tangible and Embedded Interaction - Conference Proceedings*, pp. 11–14, 2008.

[3] "The global Wii experience website," 2010. [Online]. Available: http://us.wii.com/iwata_asks/wii_remote/

[4] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics - Part C*, vol. 37, no. 3, pp. 311–324, 2007.

[5] J. Alon, V. Athitsos, Q. Yuan, and S. Sclaroff, "A unified framework for gesture recognition and spatiotemporal gesture segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 1685–1699, 2009.

[6] H.-K. Lee and J. H. Kim, "An HMM-based threshold model approach for gesture recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 961–973, 1999.

[7] J. Jia, J. Jiang, and D. Wang, "Recognition of hand gesture based on Gaussian mixture model," in *Content-Based Multimedia Indexing, 2008. CBMI 2008. International Workshop on*, 18-20 2008, pp. 353 –356.

[8] S. Zhao, W. Tan, S. Wen, and Y. Liu, "An improved algorithm of hand gesture recognition under intricate background," in *ICIRA '08: Proceedings of the First International Conference on Intelligent Robotics and Applications*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 786–794.

[9] J. Alon, V. Athitsos, and S. Sclaroff, "Accurate and efficient gesture spotting via pruning and subgesture reasoning," in *In Proc. IEEE ICCV Workshop on Human Computer Interaction*, 2005, pp. 189–198.

[10] X. Zhang, X. Chen, W.-h. Wang, J.-h. Yang, V. Lantz, and K.-q. Wang, "Hand gesture recognition and virtual game control based on 3D accelerometer and EMG sensors," in *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2009, pp. 401–406.

[11] Y. Fujiki, "iPhone as a physical activity measurement platform," in *CHI EA '10: Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*. New York, NY, USA: ACM, 2010, pp. 4315–4320.

[12] C. A. Wingrave, B. Williamson, P. D. Varcholik, J. Rose, A. Miller, E. Charbonneau, J. Bott, and J. J. L. Jr., "The wiimote and beyond: Spatially convenient devices for 3D user interfaces," *IEEE Computer Graphics and Applications*, vol. 30, pp. 71–85, 2010.

[13] ADXL330, *Small, Low Power, 3-Axis* $\pm 3g$ *iMEMS Accelerometer,* Analog Devices, 2007.

[14] "Wiimote - WiiBrew," 2010. [Online]. Available: http://wiibrew.org/wiki/Wiimote

[15] H. Weinberg, "Using the ADXL202 in pedometer and personal navigation applications," *application note AN602, Analog Device*.

[16] Q. Ladetto, "On foot navigation: Continuous step calibration using both complementary recursive prediction and adaptive Kalman filtering," in *ION GPS 2000, Salt Lake City, Utah, USA*, 2000.

[17] S. Y. Cho and C. G. Park, "MEMS based pedestrian navigation system," *The Journal of Navigation*, vol. 59, no. 01, pp. 135–153, 2006.

[18] A. Akl and S. Valaee, "Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing," *2010 IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP)*, pp. 2270 –2273, March 2010.

[19] A. Akl, C. Feng, and S. Valaee, "A novel accelerometer-based gesture recognition system," *Submitted to IEEE Transactions on Signal Processing*, 2010.

[20] E. Keogh, "Exact indexing of dynamic time warping," in *VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases*. VLDB Endowment, 2002, pp. 406–417.

[21] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, February 2007. [Online]. Available: http://dx.doi.org/10.1126/science.1136800

[22] E. Candès and M. Wakin, "An introduction to compressive sampling," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 21 –30, March 2008.

[23] R. Baraniuk, M. Davenport, R. Devore, and M. Wakin, "A simple proof of the restricted isometry property for random matrices," *Constr. Approx*, vol. 2008, 2007.

[24] E. J. Candès, J. K. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, August 2006. [Online]. Available: http://dx.doi.org/10.1002/cpa.20124

[25] E. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203 – 4215, dec. 2005.

[26] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, February 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.2005.862083

[27] A. Cohen, W. Dahmen, and R. Devore, "Compressed sensing and best $k$-term approximation," Tech. Rep., 2006.

[28] D. Achlioptas, "Database-friendly random projections," *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 274–281, 2001.

[29] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: Applications to image and text data," *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 245–250, 2001.

[30] J. Lin and D. Gunopulos, "Dimensionality reduction by random projection and latent semantic indexing," *of the Text Mining Workshop, at the 3rd SIAM International Conference on Data Mining*, May 1-3, 2003.

[31] W. B. Johnson and J. Lindemstrauss, "Extensions of Lipshitz mapping into Hilbert space," *Conference in Modern Analysis and Probability, volume 26 of Contemporary Mathematics,* pp. 189-206, 1984.

[32] R. Hecht-Nielsen, "Context vectors: general purpose approximate meaning representations self-organized from raw data," in J. M. Zurada, R.J. Marks II, and C.J. Robinson, editors, *Computational Itelligence: Imitating Life*, 1994, pp. 43–56.

[33] Y. Lu and M. Do, "Sampling signals from a union of subspaces," *Signal Processing Magazine, IEEE*, vol. 25, no. 2, pp. 41 –47, Mar. 2008.

[34] E. Candès and T. Tao, "Near-optimal signal recovery from random projections: Universal encoding strategies?" *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406 –5425, Dec. 2006.

[35] L. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257 –286, feb 1989.

[36] L. E. Baum and J. A. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a markov process and to a model of ecology," *Bulletin of the American Meteorological Society,* vol. 73, pp. 360-363, 1967.

[37] L. E. Baum and G. R. Sell, "Growth functions for transformations on manifolds," *Pacific Journal of Mathematics,* vol. 27, no. 2, pp. 211-227, 1968.

[38] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition," *Bell System Technical Journal,* vol. 62, no. 4, pp. 1035-1074, April 1983.

[39] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society,* vol. 39, no. 1, pp. 1-38, 1977.

[40] A. Yang, S. Iyengar, S. Sastry, R. Bajcsy, P. Kuryloski, and R. Jafari, "Distributed segmentation and classification of human actions using a wearable motion sensor

network," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, June 2008, pp. 1 –8.

[41] T. Pylvänäinen, "Accelerometer based gesture recognition using continuous HMMs," in *Pattern Recognition and Image Analysis*. New York, NY, USA: Springer Berlin / Heidelberg, 2005, pp. 639–646.

[42] J. Kela, P. Korpip, J. Mantyjarvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca, "Accelerometer-based gesture control for a design environment," *Personal Ubiquitous Comput.*, vol. 10, no. 5, pp. 285–299, 2006.

[43] C. Feng, S. W. A. Au, S. Valaee, and Z. H. Tan, "Orientation-aware localization using affinity propagation and compressive sensing," *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing , CAMSAP*, 2009.