ACCELEROMETER BASED CALCULATOR FOR VISUALLY-IMPAIRED PEOPLE USING MOBILE DEVICES

by

Doğukan Erenel

B.S. in Computer Engineering, Istanbul Technical University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2011

ACCELEROMETER BASED CALCULATOR FOR VISUALLY-IMPAIRED PEOPLE USING MOBILE DEVICES

APPROVED BY:

Assoc. Prof. Haluk Bingöl (Thesis Supervisor)	.....
Prof. Lale Akarun	.....
Assoc. Prof. Zehra Çataltepe	.....

DATE OF APPROVAL: 20.07.2011

ACKNOWLEDGEMENTS

This work is partially supported by Boğaziçi University Research Fund Project BAP-2011-6017.

ABSTRACT

ACCELEROMETER BASED CALCULATOR FOR VISUALLY-IMPAIRED PEOPLE USING MOBILE DEVICES

Within the popularity of new interface devices such as accelerometer based game controllers or touch-screen smartphones, the need of new accessibility options for these interfaces have become emergent. Previous studies gave the idea of using accelerometer based gesture recognition system on touch-screen smartphones with accelerometer as a new interface for visually-impaired people to use touchscreen keyboards. However, almost all studies, which have high accuracy results, are used user-dependent classifications or very limited gesture sets.

In this study, our aim is to find an alternative approach to classify 20 different gestures captured by iPhone 3GS's built-in accelerometer and make high accuracy on user-independent classifications. Our method is based on Dynamic Time Warping (DTW) with dynamic warping window sizes. The first experimental result, which is obtained from collected data set, gives 96.7% accuracy rate among 20 gestures with 1062 gesture data totally. The second experimental result, which is obtained from 4 visually-impaired people with implemented calculator as end-user test, gives 95.5% accuracy rate among 15 gestures with 720 gesture data totally. Within this work, a design of accelerometer-based recognition system is given as well as its implementation as a gesture recognition based talking calculator and experimental results.

ÖZET

GÖRME ENGELİLER İÇİN MOBİL CİHAZLARDA İVMEÖLÇER TEMELLİ HESAP MAKİNASI

İvmeölçerli oyun kumandaları ve dokunmatik ekranlı akıllı telefonların popülerlik kazanması ile birlikte; bu yeni arayüzlerin erişilebilirlik seçeneklerine ihtiyaçlar önem kazanmaktadır. Önceki çalışmalar, ivme ölçer temelli el hareket tanıma sistemlerinin, ivmeölçerli dokunmatik ekran telefonlarda görme engellilerin dokunmatik ekran klavye kullanımları için yeni bir arayüz olarak kullanılması fikrini vermiştir. Ancak, yüksek tanıma oranlarına sahip hemen bütün çalışmalar kullanıcı bağımlı tanımalar yapmakta veya sınırlı bir hareket kümesine sahip bulunmaktadırlar.

Bu çalışmada amacımız, Apple iPhone 3GS mobil cihazın ivmeölçeri kullanılarak 20 adet el hareketinin tanınmasına alternatif bir çözüm ve kullanıcı bağımsız yüksek tanıma oranı sunabilmektir. Metodumuz temeli, dinamik zaman bükmesi ile birlikte dinamik sarmal çerçeve boyutu kullanılmasına dayanmaktadır. Toplanan el hareket verileri üzerinde yapılan birinci deneye göre; 20 farklı el hareketi için 1062 veri toplanmış ve %96.7 tanıma başarısı elde edilmiştir. İkinci deneyde, sistem yapılan hesap makinası ile 4 görme engelli tarafından denenmiştir. Toplanan 15 farklı el hareketi için toplam 720 örnek alınmış ve ortalama %95.5 tanıma başarısı sağlanmıştır. Bu çalışma ile birlikte; ivmeölçer temelli el hareket tanıma sistemi, sistemin pratik bir uygulaması olarak konuşan ve 4 işlem yapabilen basit bir hesap makinası ve deney sonuçları verilmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xii
LIST OF SYMBOLS xiii
LIST OF ACRONYMS/ABBREVIATIONS xiv
1. INTRODUCTION 1
2. BACKGROUND 4
2.1. Mathematical Notations 4
2.2. Acceleration 5
2.3. Accelerometer 8
2.3.1. Built-in Accelerometer of iPhone 3GS 8
2.3.2. Usage of Accelerometer 11
2.4. Gesture 11
2.5. Gesture Recognition Systems 12
2.5.1. Brief History 12
2.5.2. Gesture Recognition 12
2.5.3. Accelerometer Based Gesture Recognition 13
2.6. Digital Signal Processing 14
2.6.1. Digital Filter 14
2.6.1.1. Low-Pass Filter 15
2.6.2. Sampling 15
2.6.2.1. Downsampling 17
2.7. Dynamic Time Warping 17
2.7.1. General DTW Algorithm 18
2.7.1.1. Example of DTW Algorithm 19
2.7.1.2. Conditions of DTW Algorithm 21
2.7.2. DTW Algorithm in Thesis 24

2.7.3. LBKeogh Distance 25
2.8. Statistical Information 28
2.8.1. Basic Information 28
2.8.2. Cross Validation 28
2.8.3. Confusion Matrix 29
2.9. Previous Works 30
3. PROPOSED METHOD 37
3.1. Data Acquisition 37
3.2. General Model 39
3.2.1. Template Generation 40
3.2.2. Recognition Processes 43
3.3. Theory 44
3.3.1. Definitions 44
3.3.2. Assumptions 45
3.4. Data Validation 46
3.4.1. Gesture Length Data Validation 46
3.4.2. Average Amplitude Data Validation 47
3.5. Filtering Accelerometer Data 48
3.6. Preprocess 49
3.6.1. Mean Adjustment 50
3.6.2. Variance Adjustment 52
3.7. Downsampling 53
3.8. DTW Training Processes 55
3.8.1. DTW Training 55
3.8.1.1. Template Library Generation 55
3.8.1.2. Best Warping Window Size 57
3.9. Recognition Processes 60
3.9.1. Finding Closest Class with Threshold Control 60
3.9.2. Threshold Control 60
3.10. Paramater Tuning 62
3.11. Experimental Results 62
3.11.1. Implementation Information 62

3.11.2. Experimental Information 63
3.11.3. Results 64
4. CONCLUSION AND FUTURE WORK 68
APPENDIX A: EFFECTS OF PROCEDURES 70
APPENDIX B: FINDINGS FROM TRAINING PART 72
APPENDIX C: ALGORITHMS 76
APPENDIX D: CASE STUDY 82
REFERENCES 88

LIST OF FIGURES

Figure 2.1. Acceleration example. 6

Figure 2.2. Acceleration movement characteristics. 7

Figure 2.3. iPhone axes orientation. 8

Figure 2.4. Example gesture data illustration. 10

Figure 2.5. Example of signal sampling. 16

Figure 2.6. DTW matching example. 20

Figure 2.7. Example adjustment window conditions. 24

Figure 2.8. LB Keogh Distance definitions. 27

Figure 2.9. Gesture set from Jie Yang and Yangsheng Xu's work. 30

Figure 2.10. Gesture set from Keskin's work. 32

Figure 2.11. Relevant coordinate systems from Mizell's work. 32

Figure 2.12. Gesture set from Kela's work. 33

Figure 2.13. Gesture set from Thomas Schlömer's work. 34

Figure 2.14. Gesture set from Jiahui Wu's work. 34

Figure 2.15. Gesture set from Tian Seng Leong's work. 35

Figure 2.16. Gesture set from Ahmad Akl and Shahrokh Valaee's work. 35

Figure 3.1. Gestures need for 4-operations calculator. 37

Figure 3.2. Gesture Set used in this project. 39

Figure 3.3. Legend used in template generation and recognition processes. . . 40

Figure 3.4. Template generating processes. 41

Figure 3.5. Template generation algorithm. 42

Figure 3.6. Gesture recognition processes. 43

Figure 3.7. Gesture recognition algorithm. 44

Figure 3.8. Low-Pass Filter algorithm. 48

Figure 3.9. Illustration of preprocess effects. 50

Figure 3.10. Downsampling algorithm. 54

Figure 3.11. General DTW Training algorithm. 55

Figure 3.12. Forward search algorithm. 59

Figure 3.13. Finding Closest Class Index algorithm. 61

Figure 3.14. First Experiment's Result. 64

Figure 3.15. Second Experiment's Result. 67

Figure A.1. Effects of procedures on classification accuracy. 70

Figure C.1. Simple DTW Distance with Dynamic Warping Window Size algorithm. 76

Figure C.2. Best Warping Window Size algorithm. 77

Figure C.3. DTW evaluation algorithm. 78

Figure C.4. Finding Best Warping Window Sizes algorithm. 79

Figure C.5. LBKeogh Distance calculation algorithm. 80

Figure C.6. DTW Distance calculation algorithm. 81

Figure D.1. Gesture types used in case study. 82

Figure D.2. Raw gesture data sequences. 83

Figure D.3. Filtered gesture data sequences. 83

Figure D.4. Preprocessed gesture data sequences. 84

Figure D.5. Sampled gesture data sequences. 85

Figure D.6. Template data sequences with upper and lower bounds. 85

Figure D.7. Raw, Filtered, Sampled validation data for Class A and Class B. . 86

Figure D.8. Template Matched data. 87

LIST OF TABLES

Table 2.1. Mathematical notations used in thesis. 4

Table 2.2. Proper acceleration of rightward movement. 7

Table 2.3. Proper acceleration of downward movement. 7

Table 2.4. Simple confusion matrix with 2 class labels. 29

Table 2.5. Comparison Table of The Previous Works and Proposed System. . 36

Table 3.1. Calculations used in second experiment. 64

Table 3.2. Confusion matrix of experimental result. 66

Table A.1. Effects of procedures on classification accuracy. 71

Table B.1. DTW distance threshold values. 72

Table B.2. Warping window sizes of x axis. 73

Table B.3. Warping window sizes of y axis. 74

Table B.4. Warping window sizes of z axis. 75

Table D.1. Template Matching result with confusion matrix. 87

LIST OF SYMBOLS

$\mathbf{C}$	Candidate gesture data
FN	The number of False Negative classification
FND	DTW distance between falsely predicted and actual class
$\mathbf{G}$	Gesture data matrix
$\mathcal{G}$	Gesture set containing all gestures collected
$\mathcal{G}_m$	Gesture set containing all gestures of gesture type m
$\{\overrightarrow{\mathbf{g}}(k)\}_{k=1}^{K}$	Gesture data definition
$\overrightarrow{\mathbf{g}}(k)$	Gesture data value at time k
I	The number of maximum iteration on DTW training
K	Length of a single discrete-time data sequence
$\mathbf{L}$	Lower limit row vector used in LB Keogh distance calculation
M	Number gesture types (gesture class)
N	Number of samples
P	Number of person who participated on gesture data collection
Pr	Precision rate from confusion matrix
$\mathbf{Q}$	Template (Query) Library (matrix)
R	Number of range limit using in LB Keogh distance calculation
Rc	Recall rate from confusion matrix
$\mathbf{T}$	Training Data Set
$\mathbf{TL}$	Template Library (Set)
TP	The number of True Positive classification
TPD	DTW distance between accurately predicted and actual class
$\mathbf{U}$	Upper limit row vector used in LB Keogh distance calculation
$\mathbf{W}$	Warping window size matrix
Φ	Threshold value

LIST OF ACRONYMS/ABBREVIATIONS

DTW	Dynamic Time Warping
HMM	Hidden Markov Model

1. INTRODUCTION

Within the popularity of new interface devices such as accelerometer based game controllers or touch-screen smartphones, the need of new accessibility options for these interfaces have become emergent. For example, some mobile devices with just touch-screens serve features like text-to-speech, speech-to-text, magnifier etc. for handicapped people or some games are designed for rehabilitation of people with physical disability or mental retardation. However, touchscreen keyboards are still in need of some improvement for visually-impaired people. More precisely, the accessibility options have served by touch-screen smartphones for text editing are insufficient for visually impaired people. This work aims to present an alternative solution for this problem.

Several research works on accelerometer based gesture recognition systems in computer science and on the usage of accelerometer based devices in medical area gave the idea of new interface for accessibility [1–5]. For example, the Wii controller is used for patients recovering from strokes, broken bones, surgery and even combat injuries with some specific games [6–8]. However, there are limited research works on these mentioned interfaces for visually-impaired people on mobile devices.

In this work, we aim to serve a reliable, fast and simple gesture recognition model and its implementation as a new interface, which is desperately needed for visually-impaired people for writing text with touchscreen smart-phones. Our motivation is the ability to serve a new interface to the current smart-phone industry without making any hardware changes; therefore the integration of handicapped people to the society is much increased.

The main contribution of this work is the new interface to write text by capturing accelerometer data of hand gestures for touch-screen smartphones with accelerometer. Our core approach (the training part) is based on the technique originally proposed in Ratanamahatana and Keogh's work [9]. In previous studies, there are few research

works tested by end-users, while the others use a limited gesture set. In this study, our proposed system is capable of classifying 20 different gestures and also it has tested by end-users with a handled device, which is iPhone 3GS.

The proposed system is separated into two parts; training (template generation) and classification (recognition process) parts and works as follows. In the training part; each gesture movement is captured by iPhone built-in accelerometer and saved into a dataset. Then each individual gesture data in the dataset is validated, preprocessed and the system is trained to generate key data, called template data, of each gesture type. After templates are ready, each individual gesture data in the dataset is validated and trained to find warping window size array and threshold values. In the classification part; a candidate gesture data is matched with each template using warping window size array and threshold values and the closest template is found as the classification result. The matching of two gesture data is based on Dynamic Time Warping (DTW) and LBKeogh algorithms, which determine the distance between two gesture data as a similarity metric [10, 11]. Here, in the training part; all operations, except the data collection, are made by PC because of it's computational power and iPhone 3GS is used in classification part with the finding from training part.

The proposed recognition system has been implemented as an iPhone application and tested with two experiments. In the first experiment, we collected training data for 20 different gesture from 15 participants with 55 samples for each gesture types. The data was used to train and validate our system. According to result of first experiment, user-independent gesture recognition accuracy rate is 96.7%. In the second experiment, we made a gesture based talking calculator with 15 different gestures by using proposed system and first experiment's training data. The calculator was tested by 4 visually-impaired people with 40 different calculations having all simple operations ('+', '−', '×', '/') and numbers. This experiment was repeated day by day for an adaptation period of 7 days. On the last day of experiment (7th day), the average accuracy rate has become 95.5%. Based on the user-independent classification accuracy rates and the number of gesture types, our proposed method seems to be one of the best among previous works (see: Comparison Table 2.5).

The remainder of this thesis is organized as follows. Chapter 2 gives the related background information with gesture recognition and preprocessing steps. Also, it gives some account of previous works on different gesture recognition approaches, relevant academic publications and their gesture sets and results. The proposed recognition system is described in Chapter 3 with the data collection part in Section 3.1, while our new and exciting results are given in Section 3.11. Finally, Chapter 4 gives conclusion for the project and ideas for future works. In the Appendix part; effects of procedures on classification accuracy, findings from training part, algorithms used in the thesis and a case study with simple gesture set are given with informative figures and tables.

2. BACKGROUND

This chapter gives an overview and related works on gesture recognition systems as well as their experimental results. Mathematical notations are given in the Section 2.1. From Section 2.2 to 2.4, definitions of acceleration, accelerometer and gesture, the usage of the accelerometer and gesture recognition systems are explained in order to understand the area of the field and future expectations. Then a brief information about gesture recognition systems in Section 2.5, digital signal processing in Section 2.6 with detailed DTW explanations in Section 2.7 and statistical information on some related topics in Section 2.8 are given. Then the gesture data sets used in previous research works with experimental results are given in Section 2.9 to give an idea on comparison with previous works.

2.1. Mathematical Notations

The mathematical notations used in this thesis are given with table below:

Table 2.1. Mathematical notations used in thesis.

Notation	Meaning of the Notation
$\mathbf{X}$	TBoldface capitalized characters denote matrices.
X	Capitalized characters (non-bold, non-italic) denote sets.
$\overrightarrow{\mathbf{x}}$	Boldface lowercase characters with arrow on top denote vectors.
$\mathbf{x}$	Boldface lowercase characters denote time series.
$x(k)$	Value of time serie x at time k.
$\mathbf{X}(k)$	k^{th} column of matrix $\mathbf{X}$.
$\{x(k)\}_{k=1}^{T}$	A sequence definition with type of x and size of T.
$x \triangleq y$	Equality operator with a small triangle denotes a definition. Here, x is defined as y.

2.2. Acceleration

Acceleration is defined to be the rate change of change of velocity over time. Acceleration is a vector in 3D, represented by $(x(t),\ y(t),\ z(t))$ at any given time t, where $x(t)$, $y(t)$, $z(t)$ are x, y, z coordinates of the acceleration vector. In generally, velocity is measured in meters per second (m/s), so acceleration is measured in meters per second squared (m/s^2), which can be positive or negative. For motion along an axis, the *average acceleration* a_{avg} over a time interval Δt is:

$$a_{avg} \triangleq \frac{v_2 - v_1}{t_2 - t_1} = \frac{\Delta v}{\Delta t} \tag{2.1}$$

where an object has velocity v_1 at time t_1 and v_2 at time t_2 $(t_2 > t_1)$ [12]. In addition, the *instantaneous acceleration* a, which is commonly used for acceleration, is the derivative of the velocity with respect to time where velocity is the change in displacement with time at a given instant:

$$a \triangleq \frac{\mathrm{d}v}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right) = \frac{\mathrm{d}^2x}{\mathrm{d}t^2} \tag{2.2}$$

As a special case; *constant acceleration* a_{const} is the acceleration, when the average and instantaneous accelerations are equal:

$$a_{const} \triangleq a = a_{avg} = \frac{v - v_0}{t - 0} \tag{2.3}$$

where v_0 is the velocity at time $t = 0$ and v is the velocity at any later time t. In addition to these essential definitions about acceleration, *free-fall acceleration* is the constant acceleration of an object while it is in free fall and which is caused by the gravitational force. Free-fall acceleration's amplitude is represented by g and it's value

near Earth's surface is $9.8m/s^2$, where acceleration is:

$$a = -g = -9.8m/s^2 \tag{2.4}$$

Note that, free-fall acceleration is negative that shows the direction of the acceleration, which is downward on the y axis and towards Earth's center. Lastly, the acceleration of an object relative to an observer, whose net acceleration is zero, is called *proper acceleration* and it is measured by an accelerometer. The illustrations, given in Figure 2.1, are some example movements to show proper acceleration values. Here, the big circle with a small mark on bottom is an glass sphere containing an object to show the effect of the acceleration. The values of acceleration of the circle are given with Table 2.2 and Table 2.3, while velocity and instantaneous acceleration is given with Figure 2.2.

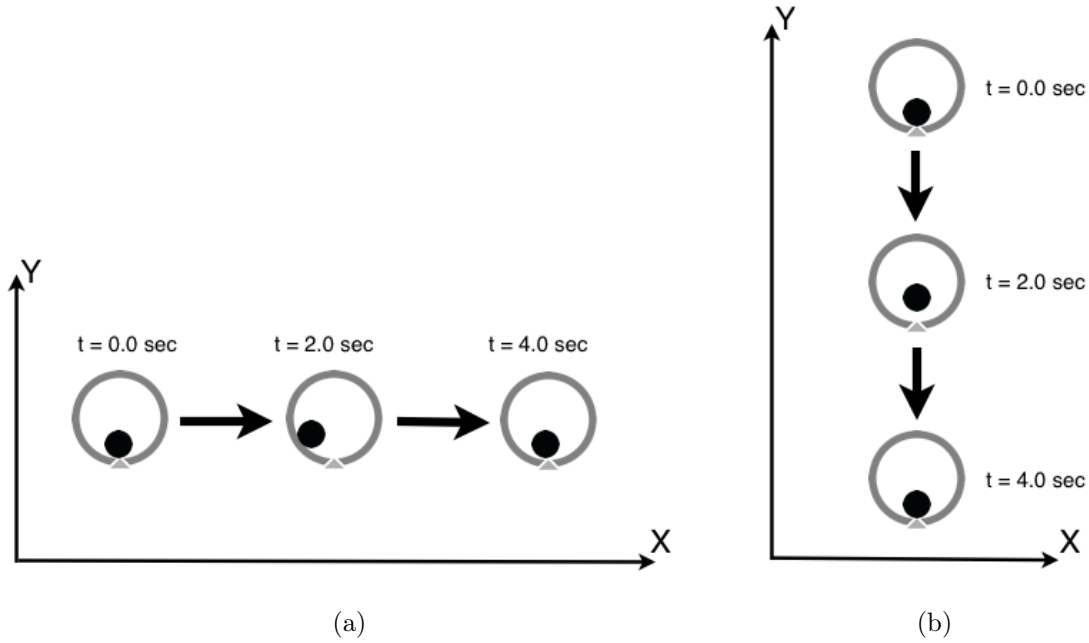


Figure 2.1. Example movements of glass sphere containing an object. (a) Rightward movement. (b) Downward movement.

Note that, the acceleration vector values are given with respect to accelerometer frame of reference. Here, there are two frame of references; one of them is inertial frame of reference, which is stationary and not accelerated and other is the frame of reference of glass sphere (or accelerometer), which is moving with an external force. In the right-

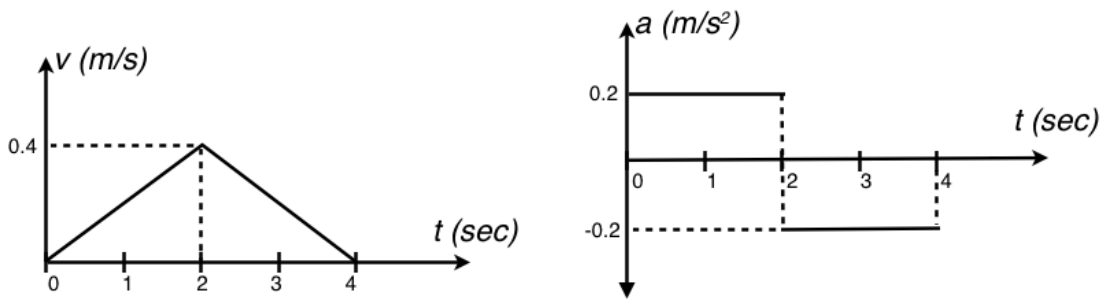
ward movement, the glass sphere is moved to the right by increasing it's velocity with a constant acceleration $0.2m/s^2$ till time $t = 2sec$ then slowing with the same acceleration value and stopped at time $t = 4sec$. Similarly, in the downward movement the glass sphere is moved downard by increasing it's velocity with a constant acceleration $0.2m/s^2$ till time $t = 2sec$ then slowing with the same acceleration value and stopped at time $t = 4sec$.

Table 2.2. Proper acceleration values of the sphere with rightward movement.

Acceleration on axis (m/s^2) \ Time (sec)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
x	0.0	0.2	0.2	0.2	0.0	-0.2	-0.2	-0.2	0.0
y	-9.8	-9.8	-9.8	-9.8	-9.8	-9.8	-9.8	-9.8	-9.8
z	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 2.3. Proper acceleration values of the sphere with downward movement.

Acceleration on axis (m/s^2) \ Time (sec)	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
x	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
y	-9.8	-7.8	-7.8	-7.8	-9.8	-11.8	-11.8	-11.8	-9.8
z	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0



(a) (b)

Figure 2.2. (a) shows the velocity values, while (b) gives the acceleration values at a given time t of movements in Figure 2.1. Note that, the directions of velocity and acceleration are parallel with the movement directions.

2.3. Accelerometer

Accelerometer is a sensor device which measures the proper acceleration of the device, producing time series of each axes in 3D ($x(t)$, $y(t)$, $z(t)$) for each clock cycle. In generally, an accelerometer has a test mass in order to measure the force per unit of mass, so this kind of accelerometers give an output in units of g (gravity), which is equal to approximately $9.8m/s^2$. In this kind of accelerometers; if there is no change in velocity (not accelerated), the measurement value is g. Accelerometers can measure: vibrations, shocks, tilt, impacts and motion of an object (exp. falling is a case where the proper acceleration changes, since it tends toward zero).

2.3.1. Built-in Accelerometer of iPhone 3GS

In this project, we used iPhone 3GS and its built-in accelerometer[1] [13]. It is capable to work on 10 Hz to 100 Hz and gives values on x, y, z axes and time stamp data t. Although all technical details are described in LIS302DL guide; the accelerometer has a sensitivity of approximately $0.02g$ and a range of $\pm 2g$ [14]. The Figure 2.3 gives an idea of the axes of built-in accelerometer of iPhone.

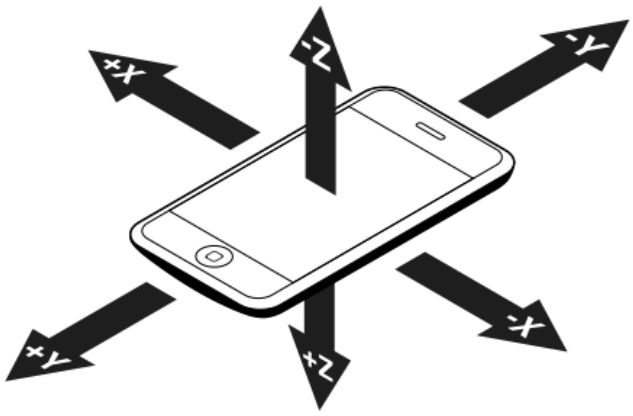


Figure 2.3. Orientation of the iPhone built-in accelerometer axes (image from Klingmann [2]).

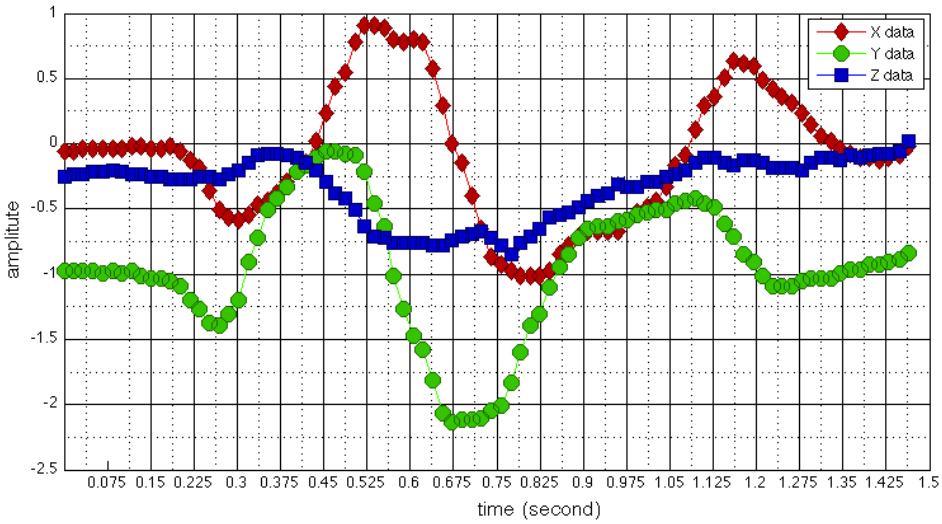
[1]At time at writing, iPhone 4 has newly released, but it was not used it in our research.

The acceleration data are specified as g-force values, where the value 1.0 corresponds to the normal acceleration caused by gravity at the Earths surface. If the device is laying still with its back on a horizontal surface, acceleration values (in 3D) will be approximately the following values: $x = 0$, $y = 0$ and $z = -1$.

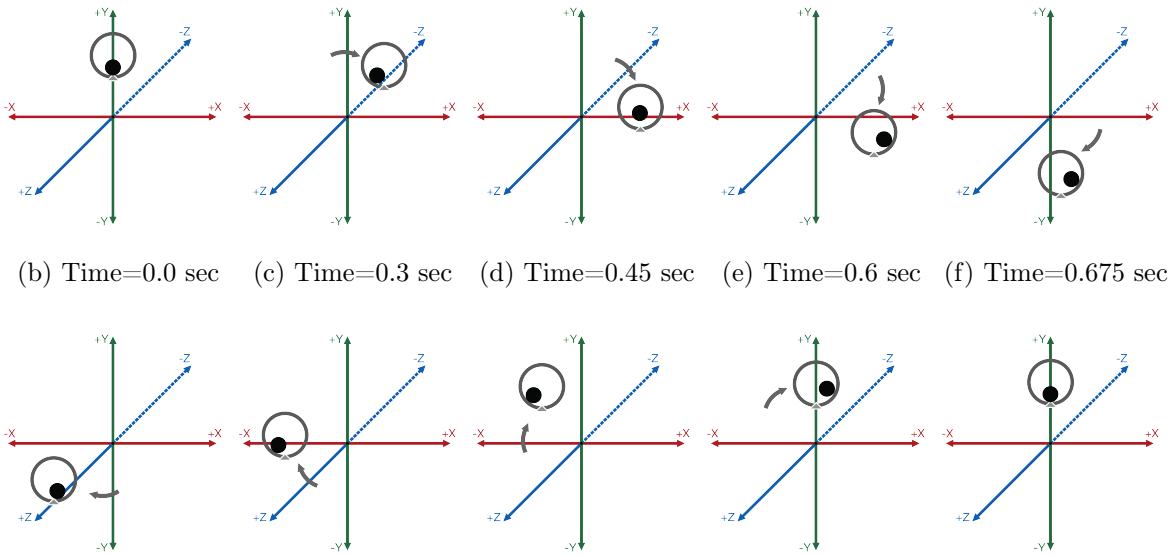
In brief, at each clock time (set by user, for example 100 Hz) accelerometer produces three (double precision) data values for acceleration on x, y and z axes with time information given in following order below.

$$\mathbf{x} = \{x(k)\}_{k=1}^{K}, \qquad \mathbf{y} = \{y(k)\}_{k=1}^{K}, \qquad \mathbf{z} = \{z(k)\}_{k=1}^{K}, \qquad \boldsymbol{\tau} = \{\tau(k)\}_{k=1}^{K} \tag{2.5}$$

where $\boldsymbol{\tau}$ contains the exact time in seconds from the starting time where $k = 1$ and $\tau(1) = 0.0$. Here, there are four data sequences starting from 1 to time K, which shows the end of the movement (length of a single discrete-time data sequence). In addition, in Figure 2.4 three data sequences can be seen as an example of a circle gesture movement with time information on x axis and amplitude values on y axis. Here, user makes a circle movement in clockwise direction starting from top point. The sphere containing an object represent the accelerometer and the object determines the effects of the acceleration. At time $t = 0$ the accelerometer produces the constant acceleration as a proper accelerations, then user starts to make the movement and stops at time $t = 1.5$.



(a)



(b) Time=0.0 sec (c) Time=0.3 sec (d) Time=0.45 sec (e) Time=0.6 sec (f) Time=0.675 sec

(g) Time=0.75 sec (h) Time=0.825 sec (i) Time=1.05 sec (j) Time=1.2 sec (k) Time=1.5 sec

Figure 2.4. Example of circle gesture data captured by iPhone 3GS accelerometer at 60Hz and its illustration with given time. (a) Data sequence in x, y and z axes combined.

Here, the Figure 2.4 (a) shows the proper acceleration values measured by accelerometer of each axis (x, y and z) at given time t. A full circle movement is provided with (b) to (k). (b) is the starting position, where time $t = 0sec$ and (k) is the end position of the sphere at time $t = 1.5sec$

2.3.2. Usage of Accelerometer

Accelerometers have wide variety of application areas from industry to education such as triggering airbag deployments or monitoring of nuclear reactors. In addition to the extreme areas, in generally; they are used to measure static acceleration (gravity), dynamic acceleration (acceleration of the device itself), orientation, vibration and tilt of an object as well as shock to an object. Nowadays, accelerometers are becoming more and more ubiquitous such that cell phones, computers and washing machines now contain accelerometers [15]. There are a couple of different usage on research area given below:

- Acceleration to Text. PhonePoint Pen work shows the possible usage of 3D accelerometer data [16]. Here, the data are converted to an image in order to give the user the ability to note down small pieces of information, quickly and ubiquitously, which can be considered as writing on air with mobile phone.
- Rehabilitation. Judith E. Deutsch et al. used Wii for rehabilitation of an adolescent with cerebral palsy, which is a group of disorder in brain and nervous system functions affecting a person's ability to move and to maintain balance and posture [6, 17]. Within this research work; the improvements in visual-perceptual processing, postural control, and functional mobility were measured after training.
- Gesture Recognition. Juha Kela et al. used accelerometer-based gesture controls for a design environment and their research work gives the experimental result of controlling the basic functionality of three different home appliances: TV, VCR and lighting [18]. This work shows the possible implementations to our daily devices and control it within a accelerometer-based interface.

2.4. Gesture

Gesture is a movement of the body or a part of the body such as hands, arms, or face used to express an idea or emotion [19]. Gestures are used to convey meaningful information as well as to interact with the environment. The movement could include

a wave, handshake, head nod, shaking of the fist, etc. involving a posture, a static configuration without the movement of the body part and a gesture, a dynamic movement of the body part. In addition, gestures are often language and culture specific. There are mainly three types of gestures; hand and arm gestures, head and face gestures and body gestures. In this thesis, we used hand and arm gestures, which are captured by iPhone built-in accelerometer. In hand and arm gestures; the static part of the body is the arm, while dynamic part is hand such as sign language or hand shaking.

2.5. Gesture Recognition Systems

2.5.1. Brief History

In 1963, light-pen gestures were used by a sketchpad, RAND tablet and one year later in 1964, the first trainable gesture recognizer developed by Teitelman as well as RAND tablet's early demonstration [20, 21]. At that time, It was quite common some gesture recognition included light-pen-based systems, for example in the AMBIT/G system [22, 23]. In 1969, Michael Coleman at CMU developed a gesture-based text editor using proof-reading symbols [24]. Commercially, CAD systems used gesture recognition since the 1970s, and came to universal notice with the Apple Newton in 1992 [25]. From that time, university and commercial researches on gesture recognition systems have been done with various other disciplinary researches.

2.5.2. Gesture Recognition

Gesture Recognition is a topic in computer science and language technology with the goal of interpreting human gestures via mathematical algorithms. There are different tools for gesture recognition, based on the approaches ranging from statistical modeling, computer vision and pattern recognition, image processing, connectionist systems, etc. Most of the problems have been addressed based on statistical modeling, such as Kalman filtering, PCA, FFT, DTW, HMM, SVM, more advanced particle filtering and condensation algorithms [10, 19, 26–32]. There are mainly two types of gesture recognition. One of them is *computer vision based gesture recognition* and the

other is *accelerometer based gesture recognition.* In brief, these types are separated from each other by the input type. In vision based recognition, the input type is some type of images (2D, 3D, etc.) whereas in accelerometer based recognition, the input is a type of accelerometer based controller whether it is a basic accelerometer or an advanced accelerometer with extra sensor information like gyroscope, etc.

2.5.3. Accelerometer Based Gesture Recognition

Accelerometer Based Gesture Recognition is a subset of pattern recognition using user controller's acceleration data as input and interprets the symbols using the feature extraction, clustering and classification methods as well as acceleration data specific data operations. The main input is a controller having at least one accelerometer, but there may additional sensor such as gyroscope or camera etc. Controller acts as an extension of the body so that when user performs a gesture, acceleration of the motion generated and identified by the recognition system. For example; Wii Remote controller or Play Station Move Motion controller are the famous examples for controller, which are used as a game controller in games [33–35]. They have their built-in accelerometers and the game console uses this data directly or after processing in a gesture recognition system.

(a) in Figure 2.4 shows an example gesture data, as 3D discrete-time data sequences, captured from iPhone 3GS built-in accelerometer. User makes a circle movement in clockwise direction starting from top point, while holding the phone facing to user. Here, the gesture $\mathbf{G}$ is a sequence;

$$\mathbf{G} \triangleq \{\overrightarrow{\mathbf{g}}(k)\}_{k=1}^{K} \tag{2.6}$$

where $\overrightarrow{\mathbf{g}}(k)$ is;

$$\overrightarrow{\mathbf{g}}(k) \triangleq \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} \tag{2.7}$$

Note that, $\mathbf{G}$ can be interpreted as an $3 \times K$ matrix. In brief, the accelerometer based gesture recognition system takes a gestures data given in Figure 2.4 and produce the symbol of circle.

2.6. Digital Signal Processing

Digital Signal Processing (DSP) is the science of using computers to understand data represented by discrete time signals and the processing of these signals [36]. DSP has a wide variety of goals: filtering, speech recognition, image enhancement, data compression, neural networks, and much more. In addition, the signal has come from numerous sources; images from remote space probes, voltages generated by the heart and brain, radar and sonar echoes, seismic vibrations, and countless other applications.

2.6.1. Digital Filter

Digital Filter is a system that performs mathematical operations on a discrete-time signal in order to enhance or reduce certain aspects of a digital signal [37]. Digital filtering is a topic related with electronics, computer science and mathematics, which means that we need to know all about signal characteristics such that; how the signal is produced (electronic component details), which aspects are need to reduce or enhance (requirements), what we need to do mathematically (mathematics) and how well we represent the algorithm as a software (computer science). In this thesis, we used a simple low-pass filter explained below.

2.6.1.1. Low-Pass Filter. A filter that reduces the amplitude (attenuates) of signals with frequencies higher than the cutoff frequency but passes low-frequency signals is called *Low-Pass Filter* [38]. Low-pass filters exist in many different forms such as digital filters for smoothing sets of data like blurring of images or electronic circuits for acoustic barriers etc. In this work, a simple low-pass filter is used which has the same logic with analog low-pass RC filter given with Equation 2.8 below:

$$y(i) = \alpha x(i) + (1 - \alpha)y(i - 1) \tag{2.8}$$

where x is input signal and y is the filtered signal and α is *Smoothing Factor* which is defined as:

$$\alpha = \frac{F_{cutoff}}{F_{cutoff} + F_{update}} \tag{2.9}$$

Here, F_{cutoff} and F_{update} are the cutoff and update frequency, respectively. Cutoff frequency is the boundary frequency for filter, where update frequency is the accelerometer update frequency.

2.6.2. Sampling

Sampling is a process that reduction of a continuous signal to a discrete signal with given sampling rate. *Sampling Rate*, which is a constant value defined by the user, is used to determine number of samples (discrete sequence length) in one second. For example; if the rate is 20 Hz, it means that sampler produce 20 data values in one second. In this project, the accelerometer gives the data in a sampled form with a sampling rate of 60 Hz, so we don't need to use a sampling but downsampling defined below. The example in Figure 2.5a shows a continuous time function starting at time $t = 0$ and ended at time $t = 29$ and Figure 2.5b gives the sampled function of given continuous time function with 1 Hz.

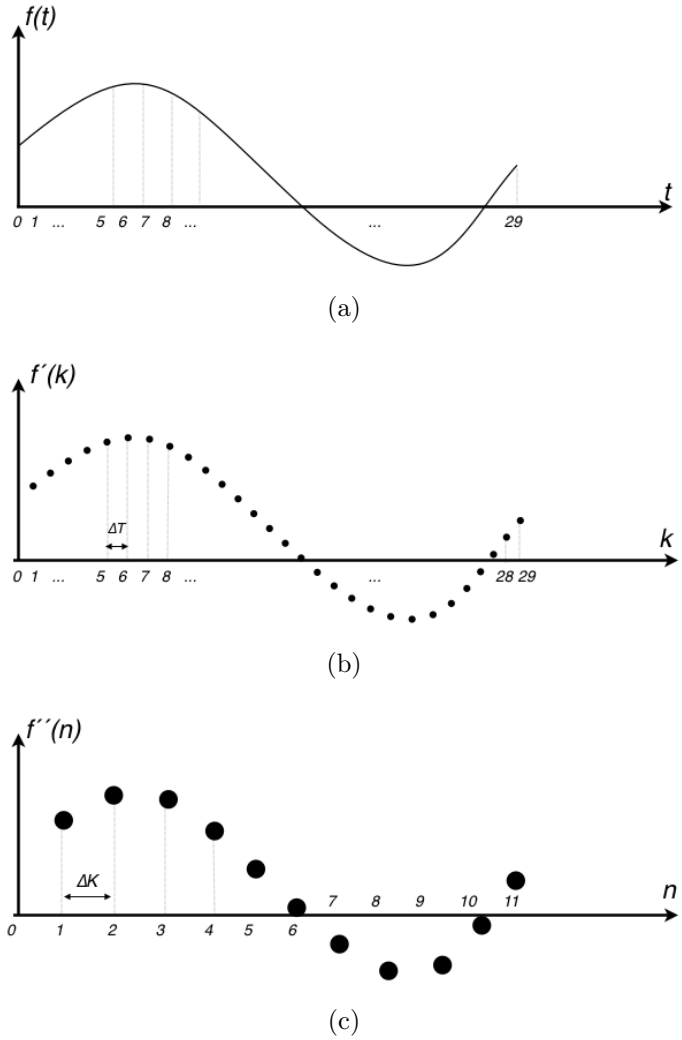


Figure 2.5. Example of signal sampling. (a) Continuous time function. (b) Sampled function (discrete time series) with 1 Hz. (c) Down-sampled sequence with scaling factor 0.4 (12 / 30).

Here, t is the time value in continuous and k and n is the time index in discrete time series. The continuous time function $f(t)$ is segmented from $t = 0$ to $t = 29$ and sampled with 1 Hz sampling rate, so the new function is named as $f'(k)$ where k is the time index between 0 and 29. Then, the sampled time series is down-sampled with 0.4 Hz downsampling rate, while the discrete signal is sampled previously with 1 Hz; therefore the downsampling factor is $0.4/0.1 = 0.4$. The down-sampled time series is called $f''(n)$ where n is the time index between 0 and 11.

2.6.2.1. Downsampling. *Downsampling* is a process that reduction the sampling rate of a discrete signal to a given downsampling rate. This process is similar to sampling but sampling operates on continuous signal whereas downsampling operates on discrete signal (sequence). In addition, in sampling; the values are directly selected from the given signal with sampling rate, in downsampling; the values are selected from an interval of values of given discrete signal according to *downsampling factor*, which defines the length of data selection interval (selection group). Downsampling factor is calculated with division of given discrete signal sampling rate and downsampling rate. For example, if a discrete time signal is sampled with 5 Hz previously and need to down-sampled with 2 Hz; the downsampling factor is $2/5 = 0.4$. In order to down-sample a discrete sequence; firstly, the given discrete signal is split into group of values, called *selection group*, according to downsampling factor. Then, a value is selected or calculated to generate down-sampled sequence for each selection group. There are number of methods in determining the value of selection groups; *simple downsampling*, *max downsampling*, *min downsampling*, *average downsampling* and *median downsampling*. In this thesis, we used average downsampling. In average downsampling; the value of a selection group is calculated by averaging the values in the groups. In addition, in this study, downsampling factor is changed according to each gesture data in order to make each gesture data with same length but the downsampling rate is constant 30 Hz, while sampling rate is 60 Hz. For example, if a gesture data has 40 data points (captured at 60 Hz) and other one is 70 data points, after downsampling each gesture data has exactly 30 data points with downsampling factors are $30/40 = 0.75$ and $30/60 = 0.5$, respectively.

2.7. Dynamic Time Warping

In this study, we have 20 different gestures and we need to classify any unknown gesture data captured by iPhone 3GS. According to computational power and memory limitation or iPhone 3GS, we need a fast classification method. In addition, we collected 55 gesture data (see: Data Acquisition in Section 3.1) of each gesture type, which is not a big data set according to previous works; therefore, the classification method should not be too dependent on the variety of training data set, otherwise classification

accuracy become very low in the end-user test. Moreover, the classification method is expected to work user-independently, which means that it should produce high accuracy recognition results in user-independent work. In order to work user-independently, our classification method can handle with problems such as the lengths of the gesture data are not the same or one part of the motion is faster then other ones etc. According to these reasons, we chose to use *Dynamic Time Warping* (DTW), which is an algorithm for finding the optimal match between two given sequences (e.g. time series) which may vary in time or speed with some certain criteria [39]. DTW is also used for measuring the similarity distance between two sequences after finding optimal match.

2.7.1. General DTW Algorithm

DTW matches two time sequence and calculate the minimum distance in warped sequence from matching result [10]. In other words, DTW matches two time signals by computing a temporal transformation, called *warping function*, causing the signals to be optimally aligned in terms of the cumulative distance between the aligned samples is minimized. In generally; DTW is implemented by calculating the matching cost and using a dynamic programming algorithm with two-dimensional matrix.
Let $\mathbf{q} = \{q(i)\}_{i=1}^{I}$ is the one known sequence called *query* (or *template*) with length of I, $\mathbf{c} = \{c(j)\}_{j=1}^{J}$ is the one unknown sequence called *candidate* with length of J and $\mathbf{D}$ is the two-dimensional matrix for matching cost of each point pairs and d is the distance between two points of given sequences. The matching cost is computed based on dynamic programming using the following formulation:

$$D(i,j) \triangleq d(q(i), c(j)) + \min \begin{pmatrix} D(i, j-1), \\ D(i-1, j-1), \\ D(i-1, j) \end{pmatrix} \tag{2.10}$$

where the boundary conditions are:

$$D(0,0) = 0, \qquad \{D(i,0)\}_{i=1}^{I} = \infty, \qquad \{D(0,j)\}_{j=1}^{J} = \infty, \tag{2.11}$$

The distance calculation is generally same as the Euclidean distance, which is the distance between points p and q is the length of the line segment connecting them ($\overline{pq}$); so $d(q(i), c(j)) \triangleq |q(i) - c(j)|$ if the sequences are one dimensional and $d(q(i), c(j)) \triangleq \sqrt{(q(i)_1 - c(j)_1)^2 + (q(i)_2 - c(j)_2)^2}$ if the sequences are two dimensional etc. In this thesis, we used one dimensional sequences (see: DTW Algorithm in Thesis in Section 2.7.2), so our distance calculation is given below:

$$d(q(i), c(j)) \triangleq |q(i) - c(j)| \tag{2.12}$$

In summary, the value of matching cost matrix $\mathbf{D}$ for given points i, j between two sequences is calculated by:

$$D(i,j) \triangleq \begin{cases} \infty, & \text{if } i = 0,\ j > 0 \\ \infty, & \text{if } i > 0,\ j = 0 \\ 0, & \text{if } i = 0,\ j = 0 \\ |q(i) - c(j)| + \min \begin{pmatrix} D(i, j-1) \\ D(i-1, j-1) \\ D(i-1, j) \end{pmatrix}, & \text{otherwise} \end{cases} \tag{2.13}$$

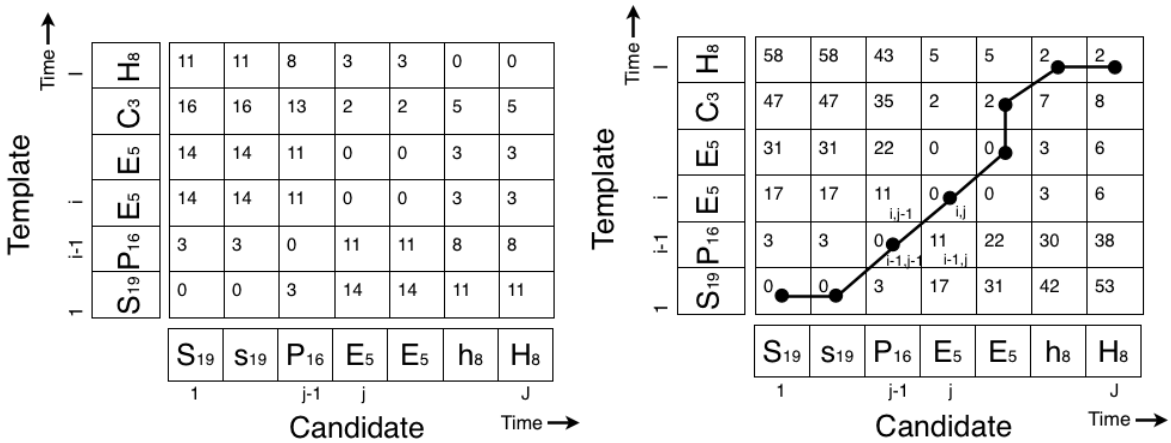
where $i \leq I$ and $j \leq J$. In addition, the *DTW distance*, which is the similarity distance between two sequences after finding optimal match between query and template data sequences, is the final point of the matching cost matrix and computed as:

$$DTW(\mathbf{q}, \mathbf{c}) \triangleq D(I, J) \tag{2.14}$$

where I is the length of sequence $\mathbf{q}$ and J is the length of sequence $\mathbf{c}$.

2.7.1.1. Example of DTW Algorithm. Figure 2.6gives an example of usage of DTW Algorithm. After taking query sequence (or template) and candidate sequence, DTW

arranges two given sequences on the sides of grid with the unknown sequence on the bottom and template on the left side. Here, our template is word “SPEECH” and our candidate is “SsPEEChH”. Now, we are determine a distance measure comparing the corresponding elements of the two sequences for each cell, and write inside the corresponding cell.



(a) Distance matrix (b) Cost matching matrix

Figure 2.6. (a) shows the Euclidean distances, while (b) gives the matching cost and optimal path between “SPEECH” and “SsPEEChH” sequences according to letter lexical order. Here, the value of lexical order is given as subscript of each letter.

Here, the aim is finding the best match between two given sequences, which is the same as finding path through the grid minimizing the total distance. In generally, the distance refers to the Euclidean distance in DTW algorithms, but here it means that the distance in lexical ordering in Latin alphabet without considering the upper or lower cases. For example; the distance between ‘B’ and ‘a’ is 1. In the figure, the lower script letter for each character show the lexical ordering index and each cell shows the calculated matching cost as well as the optimal path for minimum DTW distance.

The total distance minimizing path should start from the lower-left corner (the initial sequence points) and end upper-right corner (the last sequence points). In the given example in Figure 2.6, the first elements ‘S’ of each sequences match together while the second element of the candidate ‘s’ also matches best against the first element of the template, because the distance is 0. This corresponds to a section of template

being stretched in the input. After stretching, we can see that from the second element to fourth element of the template 'PEE' are matched with the candidates from third to fifth element 'PEE'. Then, the sixth element of the candidate 'h' matches with the sixth element of the template 'H' and in order to minimize the distance we are passing the fifth element of the template 'C', it means that this section of template has been compressed in the input sequence. Within this operations, an overall best path has been found for this template and the total distance between the two sequences is calculated by adding each cell pair distance of the optimal path and found as 2, which is the upper-left value of the matching cost matrix.

2.7.1.2. Conditions of DTW Algorithm. In DTW algorithm, there some condition that we assume such as; distance from the 0th row and column are infinite in the cost matching matrix. In addition, we need some limitations for optimum path in order to speed up the DTW distance calculation. These are some examples to show the need of optimizations on DTW which are defined in Sakoe and Chiba work [10]. Observations on the nature of good paths through the grid gives the idea of the major optimizations to the DTW algorithm which are explained briefly below. Here, $\mathbf{q}$ and $\mathbf{c}$ are two given data sequence with length of I and J respectively, and i and j are any index points where $0 < i \leq I$ and $0 < j \leq J$, and k is the any time index where $0 < k \leq I$ and $0 < k \leq J$.

- Monotonic Condition. In warping function (optimum path), the i and j indexes either stay the same or increase; they can't turn back on itself.

$$i_{k-1} \leq i_k \qquad\qquad j_{k-1} \leq j_k \tag{2.15}$$

- Continuity Condition. In warping function (optimum path), the i and j indexes can only increase by 1 on each step along the path, which means that warping

function advances one step at a time.

$$i_k - i_{k-1} \leq 1 \qquad\qquad j_k - j_{k-1} \leq 1 \tag{2.16}$$

- Sum of Monotonic and Continuity conditions. In brief; there are three choice at each step while stepping in optimum path. For any point at (i_k, j_k) on optimal path, the previous point may be (i_k, j_{k-1}) or (i_{k-1}, j_k) or (i_{k-1}, j_{k-1}), which are called *insertion*, *deletion* and *matching*, respectively. Let's assume, optimal path in DTW is $path = \{path(k)\}_{k=1}^{K}$ where K is the length of the optimal path which is $\max(I, J)$ and any point of optimal path $path(k) = (i, j)$ contains the index values. Insertion (if $path(k-1) = (i_k, j_{k-1})$) is used for the distance if we use an element from template once more while stepping into time on candidate sequence, deletion (if $path(k-1) = (i_{k-1}, j_k)$) is used for the distance if we use an element from candidate once more while stepping into time on template sequence and matching (if $path(k-1) = (i_{k-1}, j_{k-1})$) is used for the distance if we use both element from template and candidate (most probably there are matching) and stepping in time both on template and candidate sequences.

$$path(k-1) = \begin{cases} (i_k, j_{k-1}), \\ (i_{k-1}, j_{k-1}), \\ (i_{k-1}, j_k) \end{cases} \tag{2.17}$$

- Boundary Condition.The warping function (optimum path) starts at the lower-left and ends at the upper-right.

$$i_1 = 1, \qquad j_1 = 1, \qquad i_{final} = I, \qquad j_{final} = J, \tag{2.18}$$

- Warping Window Condition. In generally, the warping function (optimum path) can't be wander very far from the diagonal. There are several approaches for satisfying this condition (see: Figure 2.7);

(i) Sakoe-Chiba Band. A constant distance r is defined as a window length in order to restrict the possible paths with a diagonal band [10]. The restriction path of Sakoe-Chiba Band is given in Figure 2.7a.

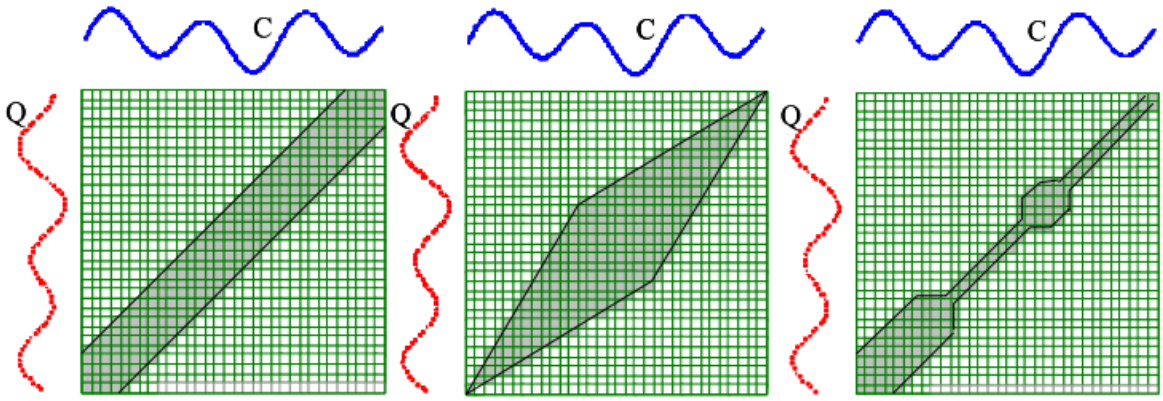
$$|i_k - j_k| \leq r \tag{2.19}$$

(ii) Itakura Parallelogram. A global contraint r is defined as a window length of given index [40]. Here, the difference from Sakoe-Chiba band is the band width is changing with given index and the window shape is a symmetric parallelogram as seen in Figure 2.7b.

(iii) Ratanamahatana-Keogh Band (R-K Band). A variable w is defined as warping window for each index according to classification result in order to find highest accuracy result [9]. An example R-K band can be seen in Figure 2.7c. In this study, R-K band is used in order to increase the classification accuracy, so our system is trained in order to find best warping window size (R-K band) for each gesture type.

$$|i_k - j_k| \leq w(k) \tag{2.20}$$

- Slope constraint condition. The path should be in a range of slope angles to prevent being too steep or too shallow as well as matching very short sequences with very long ones. There should be a ratio i/j as a slope constraint condition and according to this condition; after j steps in x we must make a step in y and vice versa (see: Figure 2.6).



(a) (b) (c)

Figure 2.7. Example adjustment window conditions, from Ratanamahatana work [9]. (a) Sakoe-Chiba band. (b) Itakura parallelogram. (c) R-K band.

2.7.2. DTW Algorithm in Thesis

In this thesis, we used DTW distance with satisfying all conditions defined above. Here, R-K Band has been selected as the warping window condition. A locality constraint as warping window size matrix $\mathbf{W}$ has been defined, as given in R-K band definition, for each axis of the gesture data $\mathbf{G}$. Note that, in this study; a template gesture data $\mathbf{Q}$, a candidate gesture data $\mathbf{C}$ and warping window size array $\mathbf{W}$ are both the same size matrices with $3 \times N$. Here N is the length of each sequences, because after downsampling all sequences become of the same length (see: Downsampling in Section 2.6.2.1). The matrices are;

$$\mathbf{Q} \triangleq \{\overrightarrow{\mathbf{g}}(n)\}_{n=1}^{N}, \qquad \mathbf{C} \triangleq \{\overrightarrow{\mathbf{c}}(n)\}_{n=1}^{N}, \qquad \mathbf{W} \triangleq \{\overrightarrow{\mathbf{w}}(n)\}_{n=1}^{N}, \tag{2.21}$$

where n^{th} column of $\overrightarrow{\mathbf{q}}(n)$, $\overrightarrow{\mathbf{c}}(n)$ and $\overrightarrow{\mathbf{w}}(n)$ are;

$$\overrightarrow{\mathbf{q}}(n) \triangleq \begin{bmatrix} x(n) \\ y(n) \\ z(n) \end{bmatrix}, \qquad \overrightarrow{\mathbf{c}}(n) \triangleq \begin{bmatrix} x(n) \\ y(n) \\ z(n) \end{bmatrix}, \qquad \overrightarrow{\mathbf{w}}(n) \triangleq \begin{bmatrix} x(n) \\ y(n) \\ z(n) \end{bmatrix}, \tag{2.22}$$

Therefore, in order to calculate the DTW distance between $\mathbf{Q}$ and $\mathbf{C}$, the matching cost is calculated for each dimension (x, y and z) and added to each other. The matching cost for x axis sequence is given below with DTW distance calculation. Here, the distance calculation is the same as finding one dimensional Euclidean distance $d(q(i), c(j)) = |q(i) - c(j)|$ as given in general DTW algorithm.

$$D_x(i,j) \triangleq \begin{cases} d(q_x(i), c_x(j)) + M_x, & \text{if } \mathbf{w}_x(\max(i,j)) \leq |i-j| \\ \infty, & \text{otherwise} \end{cases} \tag{2.23}$$

where M_x is the minimum distance coming from previous route and is defined:

$$M_x \triangleq \min(D_x(i, j-1), D_x(i-1, j-1), D_x(i-1, j)) \tag{2.24}$$

The DTW distance calculation becomes:

$$DTW(\mathbf{Q}, \mathbf{C}) \triangleq D_x(N,N) + D_y(N,N) + D_z(N,N) \tag{2.25}$$

Note that, the pseudocode of DTW distance calculation is given in the Appendix C in Figure C.6. Here, there are three dimensions but the distance calculation is not the same with Euclidean distance calculation for three dimensions, but a simple addition operation. In our experiments, we tested Euclidean distance calculation, weighted addition (multiplying each dimension with a variable and sum all) and simple addition and found that simple addition operation gives higher accuracy rate in classification.

2.7.3. LBKeogh Distance

In this thesis, we mainly used DTW distance to matching two data sequences defined above. However, in order to decrease time consumption in recognition processes using DTW distance we used one more distance metric before calculating the DTW distance. *Lower-Bound Keogh (LBKeogh) Distance* is a tool for lower bounding various

time series distance measures [11]. It is used for indexing DTW in a fast way. Here, we are defining two more time series for each templates and a constant value R for determining the lower and upper bound range. Lower bounds and upper bounds are defined in order to make indexing fast, which are:

$$\mathbf{L} \triangleq \{\overrightarrow{\mathbf{l}}(n)\}_{i=1}^{N}, \qquad \mathbf{U} \triangleq \{\overrightarrow{\mathbf{u}}(n)\}_{i=1}^{N}, \tag{2.26}$$

where n^{th} column of $\overrightarrow{\mathbf{l}}(n)$ and $\overrightarrow{\mathbf{u}}(n)$ are;

$$\overrightarrow{\mathbf{l}}(n) \triangleq \begin{bmatrix} x(n) \\ y(n) \\ z(n) \end{bmatrix}, \qquad \overrightarrow{\mathbf{u}}(n) \triangleq \begin{bmatrix} x(n) \\ y(n) \\ z(n) \end{bmatrix}, \tag{2.27}$$

where i^{th} value of $\overrightarrow{\mathbf{l}}(n)$ and $\overrightarrow{\mathbf{u}}(n)$ for x axis are:

$$\begin{aligned} l(i)_x &\triangleq \min\{q_x(j) | j \in [\max\{1, i-R\}, \min\{N, i+R\}]\} \\ u(i)_x &\triangleq \max\{q_x(j) | j \in [\max\{1, i-R\}, \min\{N, i+R\}]\} \end{aligned} \tag{2.28}$$

As an example; consider a sequence with indices $\{1, 2, ..., k-R, k-R+1, ..., k, k+1, ..., k+R, k+R+1, ..., N\}$, where k is an arbitrary index ($0 < k \leq N$) and N is the length of the sequence and R is the range value of lower and upper bounds. Here, the upper and lower bounds at k index are calculated by the maximum and minimum value in subsequence $\{k-R, ..., k, ..., k+R\}$, respectively.

In the given Figure 2.8; a protective envelope is built around one dimension of template $\mathbf{Q}$ with the upper bound $\mathbf{U}$ and lower bound $\mathbf{L}$. The LBKeogh distance calculation for one dimension (x axis) between candidate sequence $\mathbf{C}$ and template sequence $\mathbf{Q}$ is

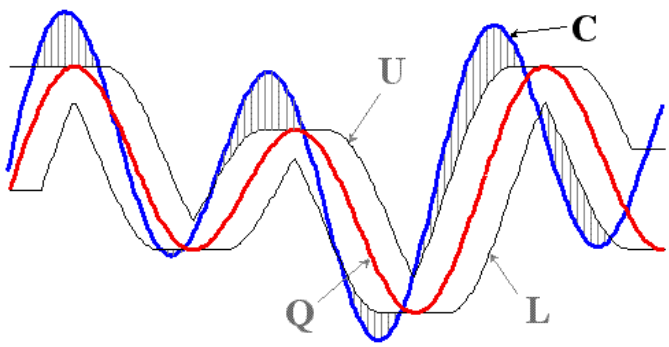


Figure 2.8. LB Keogh Distance definitions from Keogh's work [11].

given below:

$$LB_{Keogh}(\mathbf{Q}_x, \mathbf{C}_x) \triangleq \sum_{i=1}^{N} \begin{cases} (q_x(i) - u_x(i))^2, & \text{if } q_x(i) > u_x(i) \\ (q_x(i) - l_x(i))^2, & \text{if } q_x(i) < l_x(i) \\ 0, & \text{otherwise} \end{cases} \tag{2.29}$$

In this study, we used 3 dimensional sequences as described before; so our distance definition becomes;

$$LB_{Keogh}(\mathbf{Q}, \mathbf{C}) \triangleq LB_{Keogh}(\mathbf{Q}_x, \mathbf{C}_x) + LB_{Keogh}(\mathbf{Q}_y, \mathbf{C}_y) + LB_{Keogh}(\mathbf{Q}_z, \mathbf{C}_z) \tag{2.30}$$

Note that, the pseudocode of LB Keogh distance calculation is given in the Appendix C in Figure C.5.

2.8. Statistical Information

2.8.1. Basic Information

Mean, indicated by μ, is a term equivalent to the average value of a signal as given below [36]:

$$\mu \triangleq \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.31}$$

Variance, indicated by v, is a term equivalent to the average value of squared difference of mean and the sequence points, which is given below:

$$v \triangleq \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2 \tag{2.32}$$

2.8.2. Cross Validation

Cross validation (CV) is a popular technique for selecting the model or estimating the risk of an estimator [41]. The main idea on cross validation is to split data into two parts (once or several times till all data are used in validation set once) for estimating how accurately a predictive model will perform in practice. One split of data, called *training set*, is used for training the algorithms and the remaining part of the data, called *validation set* or *testing set*, is used for estimating the risk of the algorithm. According to type of CV; validation may be performed multiple times by using different partitions and the results from validation of algorithm are averaged in order to reduce variability. Then, CV selects the algorithm with the smallest estimated risk [42]. Most common types of cross validation techniques are; leave-one-out Validation (LOOV), K-fold cross validation and random sub-sampling. In this thesis, we used LOOV. In LOOV, system is trained with whole samples except for a single observation and repeating the procedure till all data are used as validation once. Here, the single

observation is used as validation data whereas the all others are used as training data.

2.8.3. Confusion Matrix

Confusion Matrix is a matrix containing all classification result of actual and predicted classes done by a classification system [43]. It shows the predicted and actual classification result. The given Table 2.4 is an example with 2 different class labels.

Table 2.4. Simple confusion matrix with 2 class labels.

Actual \ Predicted	Negative	Positive
Negative	TN	FP
Positive	FN	TP

There are some definitions related with confusion matrix and used in this project given below:

- True Positive (TP).The number of correct predictions that an instance is positive.
- True Negative (TN). The number of correct predictions that an instance is negative.
- False Positive (FP). The number of incorrect predictions that an instance is positive.
- False Negative (FN). The number of incorrect predictions that an instance is negative.
- Accuracy. The proportion of the predictions that were correct to the total prediction number.

$$Accuracy \triangleq \frac{TP + TN}{TP + TN + FP + FN} \tag{2.33}$$

- Precision (Pr). The proportion of the predicted positive cases that were correct

to the total prediction of given class.

$$Pr \triangleq \frac{TP}{TP + FP} \tag{2.34}$$

- True Positive Rate (Recall or Rc). Recall (true positive rate - Rc) is the proportion of positive cases that were correctly identified to the total actual class numbers.

$$Rc \triangleq \frac{TP}{TP + FN} \tag{2.35}$$

2.9. Previous Works

Until this section, the general background information has been given in order to make clear all field of this research project. Now, we are looking the previous works related with our proposed method (see: Chapter 3) and result of them in order to determine our method and our expectations from our method. Here, some gesture collections used in previous works with their experimental results are given in a chronological order.

In 1994, Jie Yang and Yangsheng Xu made a gesture recognition system by using HMM classifier with a mouse controller as gesture input [44]. There were 9 gestures (see: Figure 2.9) and they collected 100 training, 50 validation samples for each gesture. The general accuracy rate was 99.78%.

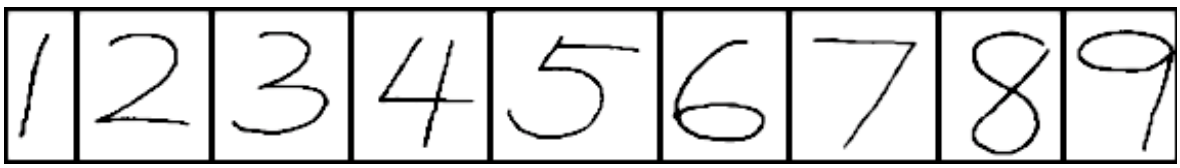


Figure 2.9. Gesture set from Jie Yang and Yangsheng Xu's research [44].

In 1995, Donald O. Tanguay, Jr. worked on a hand gesture recognition for identifying some basic letters by using HMM with a mouse controller as gesture input

(working at 20Hz) [45]. He made some experiments and lastly, he used five letters to distinguish them each other; ‘a’, ‘b’, ‘c’, ‘d’ and ‘e’. The training set contains 5 examples for each gesture as well as 5 examples for validation. Then, HMM state number and number of code book clusters are changed and found maximum 88% accuracy on validation data.

In 1995, Thad Eugene Starner used a camera (5 Frames/ sec) and colored gloves to identify sentences from American Sign Languages by HMM classifier [46]. He used 395 sentences as training data and 99 sentences as validation with totally 6000 samples, and found 97% accuracy with strong grammar and 91% accuracy rate without grammar.

In 2000, Mantyla et al. used acceleration sensors for recognition static and dynamic gestures by a HMM classifier [3]. Static gestures are; SG1 as ‘display up’, SG2 as ‘display down’, SG3 as ‘on the left/ right ear’, SG4 as ‘in the pocket’, where as dynamic gestures are DG1 as ‘checking the phone from pocket and turning the same position’, DG2 as ‘checking the phone and bringing phone to right ear ’, DG3 as ‘reverse in time of DG2’, DG4:‘checking the phone from desk and turning the same position’, DG5:‘checking the phone from desk and bringing phone to right ear’, DG6:‘reverse in time of DG5’. There were 4050 training and 6 testing samples for each static gesture and 100 training (from 1 person) and 100 testing samples (50 from same person, 50 from other person) and the average accuracy is 96.46% for static gestures where as 96% for dynamic gestures from second person.

In 2003, Keskin et al. tried to identify 8 different 3D gestures (see: Figure 2.10) captured by a camera with reconstruction 2D images (from two camera) into 3D image [47]. There were 469 elements in training set totally and 160 elements in test set. The average accuracy is 98.75% by using HMM classifier.

In same year with Keskin’s work, 2003, Tracy Westeyn et al. gave an idea about using multiple sensors on gesture recognition with ‘Georgia Tech Gesture Toolkit’ [48, 49]. There were a 5 gesture set of words {‘my’, ‘computer’, ‘helps’, ‘me’, ‘talk’} as vocabulary and 72 annotated sequences of signed sentences as test data. The indepen-

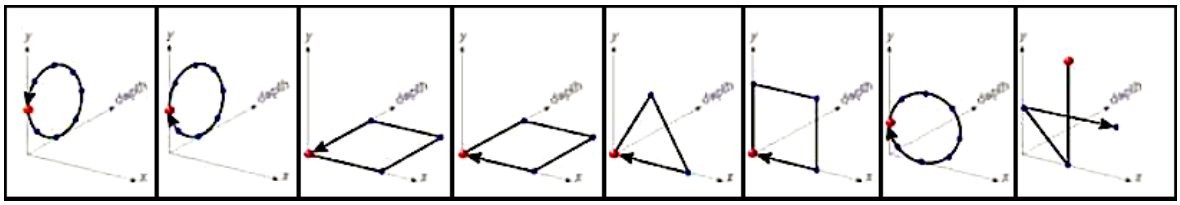


Figure 2.10. Gesture set from Keskin's work [47].

dent user recognition rate was 90.48% and user-dependent was 94.26% by combining accelerometer and vision information. In addition to gesture recognition systems, in 2003, David Mizell proposed a solution for the acceleration axis calculation problem by an estimate of the constant gravity vector from accelerometer samples average [50].

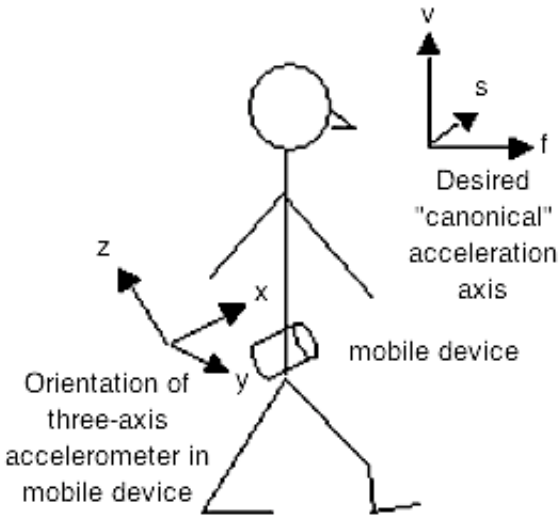


Figure 2.11. Relevant coordinate systems, image from Mizell's work [50].

In 2004, Ralph Niels made a handwriting recognition system by using DTW with two different approaches [51]. The gesture set contains 62 latin letters (all upper case [A-Z], lower case [a-z] and number [0-9] letters) with 37,683 lowercase letters, 16,451 uppercase letters and 9,195 digits. User independent classification accuracy rate was; 88.20% for lowercase letters, 90.64% for uppercase letters, and 95.26% for digits with 'Resample and Average' prototype set and 90.32% for lowercase letters, 94.28% for uppercase letters, and 97.16% for digits with 'Mergesamples' prototype set.

In addition to accelerometer based gesture recognition system, in 2004, Choti-

rat Ann Ratanamahatana and Eamonn Keogh proposed a dynamic window distance calculation as an addition of Sakoe-Chiba Band and Itakura Parallelogram, called Ratanamahatana-Keogh Bands (or *R-K band*), for improving the accuracy (see: Section 2.7.1) [9]. In this work, they used 3 different gesture data set ('Gun' contains 2 different classes with 100 samples for each class, 'Trace' contains 4 different class with 50 samples for each class, 'Word Spotting' contains 4 class with 272 samples totaly) and found 99.5% accuracy for 'Gun' data set, 100% accuracy for 'Trace' data set and 99.67% accuracy for 'Word Spotting' data set.

In 2006, Juha Kela et al. used acceleration data for controlling some daily operations, such as TV, VCR and lighting [18]. In accuracy experiments, they used 8 gestures (see: Figure 2.12), which are determined according to questionnaires results, with 240 samples for each gestures totally. All gestures were captured by a wireless sensor device equipped with accelerometers. The training set contains 240 samples for each gesture, while 28 samples for validation set and the experiment is repeated with different training set sizes. A recognition rate of over 90% was achieved with four training vectors by using HMM classifier.

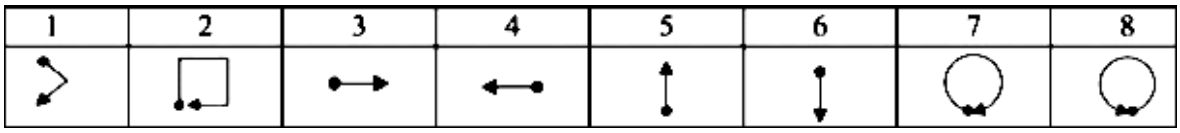


Figure 2.12. Gesture set from Kela's work [18].

In 2008, Thomas Schlömer et al. made a research on a gesture recognition system with a wii controller using HMM classifier [8]. There were 5 different gestures (see: Figure 2.13) with 90 samples for each gesture by 6 participants. The classifier was HMM and average accuracy on recognition by gestures were; square = 88.8%, circle = 86.6%, roll = 84.3%, z = 94.3%, and tennis = 94.5%.

In the same year, 2008, Zoltan Prekopcsak proposed a real-time hand gesture recognition system by using two different classifiers; HMM and SVM classifiers [52]. The experimental results showed that the two different classifiers reach 97.4% (with HMM) and 96% (with SVM) accuracy on a personalized gesture set containing 10 dif-

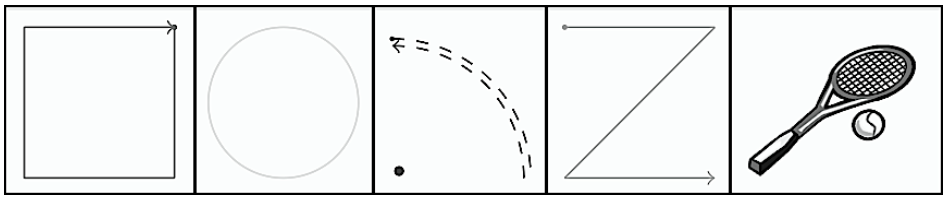


Figure 2.13. Gesture set from Thomas Schlömer's work [8].

ferent gestures with 400 samples totally captured by Sony-Ericsson W910i mobile phone which has in-built three-dimensional accelerometer accessible from Java environment.

In 2009, Jiahui Wu et al. proposed an acceleration-based gesture recognition approach called Frame-based Descriptor and multi-class SVM [1]. There were 3360 gesture samples of 12 gestures (see: Figure 2.14) captured a wearable 3-dimensional accelerometer. The experimental results showed that proposed system made 95.21% for all the 12 gestures in the user-dependent case, whereas 89.29% in the user-independent case which is higher than the compared classifiers; DTW, Naïve Bayes, C4.5 and HMM.

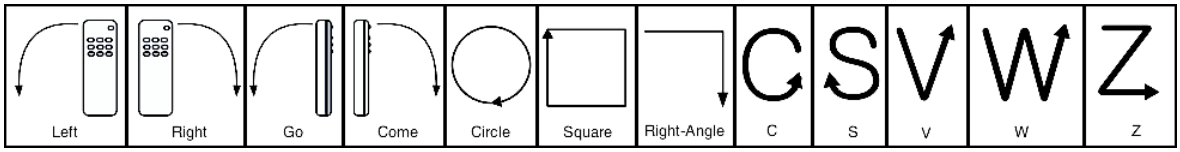


Figure 2.14. Gesture set from Jiahui Wu's work [1].

In addition to FDSVM, in 2009, uWave was proposed which is an efficient personalized gesture recognizer based on a 3-D accelerometer and it allows users to define personalized gestures with a single sample [53]. According to experimental results; there were 4480 gestures from 8 participant and uWave achieves accuracy of 98.6% in user-dependent case, whereas 75.4% in user-independent case. The gesture set was the same in Kela's work in 2006 which is given in Figure 2.12.

In the same year 2009, Tian Seng Leong et al. proposed an accelerometer-based recognition algorithm using DTW classifier [54]. The data were captured using 3-D accelerometer of the Wiimote. There were 10 different gestured, given in Figure 2.15, with 500 samples of each gestures from 50 participant. The recognition rate was 97%

using 3 or more examples in the training part.

Figure 2.15. Gesture set from Tian Seng Leong's work [54].

In 2010, Ahmad Akl, Shahrokh Valaee proposed a gesture recognition system based primarily on a single 3-D accelerometer [55]. There were 18 gestures (see: Figure 2.16) with 3,700 samples totally from 7 participants. The proposed recognizer archives 99.79% in user-dependent case, while 93.5% in user-independent case.

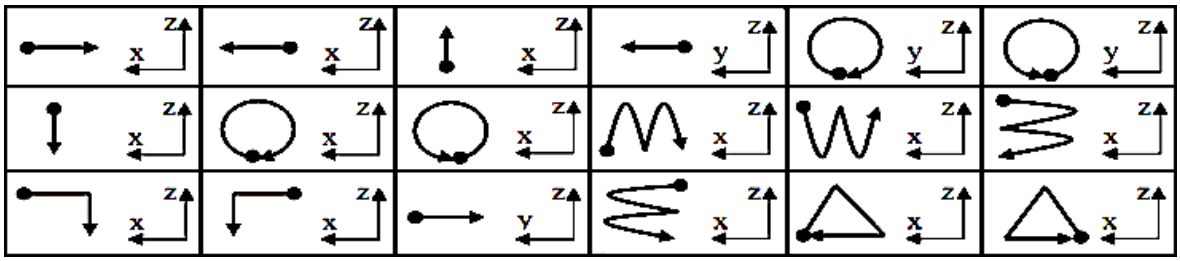


Figure 2.16. Gesture set from Ahmad Akl and Shahrokh Valaee's work [55].

In summary, the comparison table (see: Table 2.5) shows the features of the previous works and the proposed recognition system. The compared features are; the number of data in training set and in validation set, the number of symbol used in the system and user-dependent and user-independent classification results as true positive rate (recall).

Table 2.5. Comparison Table of The Previous Works and Proposed System.

Method	Data Collection	# of Symbols	User-Dependent *Rc*	User-Independent *Rc*
Jie Yang et al. [44]	100 samples for training and 50 samples for validation of each gesture	9	99%	-
Donald O. Tanguay [45]	5 samples for training and 5 samples for validation of each gesture	5	88%	-
Thad Eugene Starner [46]	395 five word sentences for training and 99 five word sentences for validation totally	40	99.4% with grammar, 93.5% without grammar	97.0% with grammar, 90.7% without grammar
Mantyla et al. [3]	4,050 samples for training and 6 samples for testing of each static gesture and 100 samples for training and 100 samples for testing of each dynamic gesture	4 static, 6 dynamic gesture	96.46% for static, 96% for dynamic gesture (50% user-independent)	-
Keskin et al. [47]	469 samples for training and 160 samples for testing totally	8	98.75%	-
Tracy Westeyn et al. [48]	72 annotated sequences of signed sentences for testing data	5	94.26%	90.48%
Ralph Niels [51]	37,683 lowercase letters, 16,451 uppercase letters and 9,195 digits for training totally	62 (totally)	90.32%: lowercase letters, 94.28%: uppercase letters, 97.16%: digits	88.20%: lowercase letters, 90.64%: uppercase letters, 95.26%: digits
Ratanamahatana et al. [9]	100 samples of each gesture from 'Gun' set, 50 samples of each gesture from 'Trace' set, 272 totally from 'Handwritten' data set for training	3, 4, 4 (respectively)	99.5%, 100%, 99.67% (respectively)	-
Juha Kela et al. [18]	240 samples for training, 28 samples for validation of each gesture	8	-	90%
Thomas Schlömer et al. [8]	90 samples for training of each gesture	5	89.7% Average	-
Zoltan Prekopcsak [52]	400 samples for training totally	10	97.4% with HMM, 96% with SVM	-
Jiahui Wu et al. [1]	3360 samples for training totally	12	95.21%	89.29%
Tian Seng Leong et al. [54]	500 samples for training of each gesture	10	97%	-
Ahmad Akl et al. [55]	3700 samples for training totally	18	99.79%	96.89% for 8 gestures, ≈93.5% for 18 gestures
Proposed System	1062 samples for training totally. Experiment 1 has 1062 test samples and Experiment 2 has 720 test samples (see: Experimental Information in Section 3.11.2)	20, 15	-	96.7%: Experiment 1, 95.5%: Experiment 2 (end-user test)

3. PROPOSED METHOD

In this chapter, our proposed method is explained. Firstly, data acquisition is given to understand the our data set, then template generation and recognition processes are stated briefly. Then variables, matrices and constant values are defined with given assumptions and initialization values. After having such an initial information, our proposed method starting from the template training part to gesture recognition part are given by following the data flow given in Figure 3.4 and 3.6.

3.1. Data Acquisition

A simple 4-operations calculator requires in total 16 gestures, since 10 gestures for digits [0-9], 4 gestures for 4-operations ('+', '−', '×', '/') , 1 gesture for 'equal' operator (to calculate and give output) and 1 gesture to clear the last digit entered.

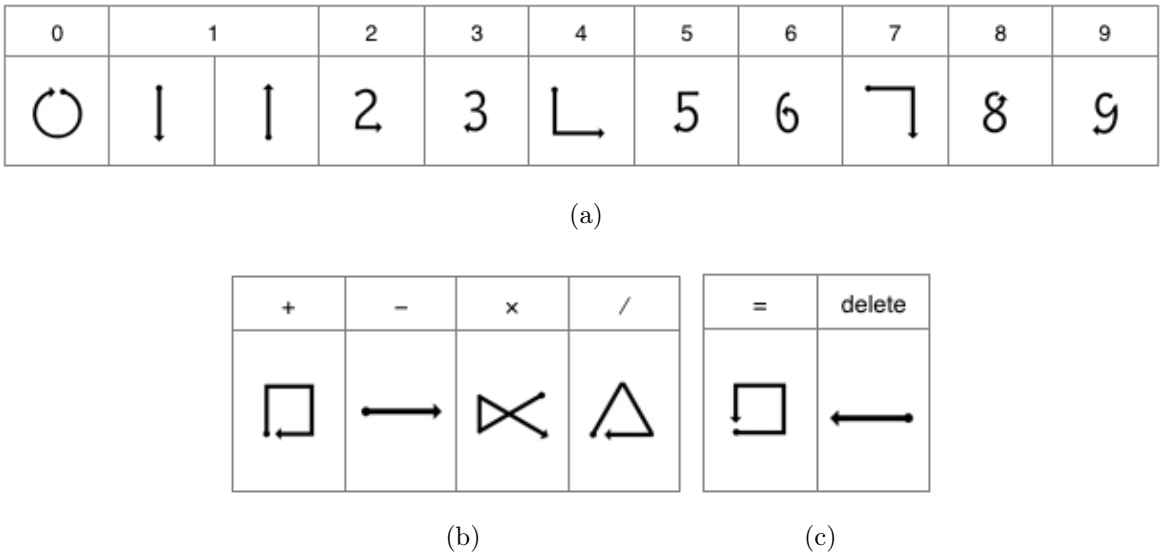


Figure 3.1. (a) shows gestures for digits, (b) shows gestures for operations and (c) shows gestures for equal and clear operations. Here the meaning of each gesture is given on top of each gesture movement (each gesture's movement is in $xy - plane$).

In this project, we gathered 22 gestures covering almost all the gestures defined in previous works (see: Section 2.9) and covering the simple 4-operations calculator ges-

tures. However, 2 of them were very close and made lower recognition accuracy than expectation (around 90% accuracy rate), so we used 20 gestures. The gesture set used in this thesis is given in Figure 3.2. In the Figure, each gesture type's; class number, the meaning and the usage in calculator are given within box shape. Gesture types are determined in order to satisfy the usage of a simple 4-operations calculator and some additional gestures using z axis.

All gesture data have gathered with iPhone 3GS at 60Hz within the implemented application. The data contains all time series of x, y and z axes with *timestamp* value for each clock time of accelerometer and additional meta information of all participant's; age, sex, profession, disability (yes or no), the hand information (right-handed or left-handed) and the current hand for gesture data. All gesture data were taken both hands from all participants in order to put various noise to our system for testing our model accuracy. In the experiments, the additional informations were not considered and every single gesture data, collected from participants, were used with the implemented application. In this study, all gesture's movement, except for the gestures [18-20], are in $xy - plane$. The gestures [18-20], has movement in $xz - plane$. In data capturing, user holds the iPhone face to him/ her and makes the given movements in Figure 3.2, while touching to the screen and holding till movement finished.

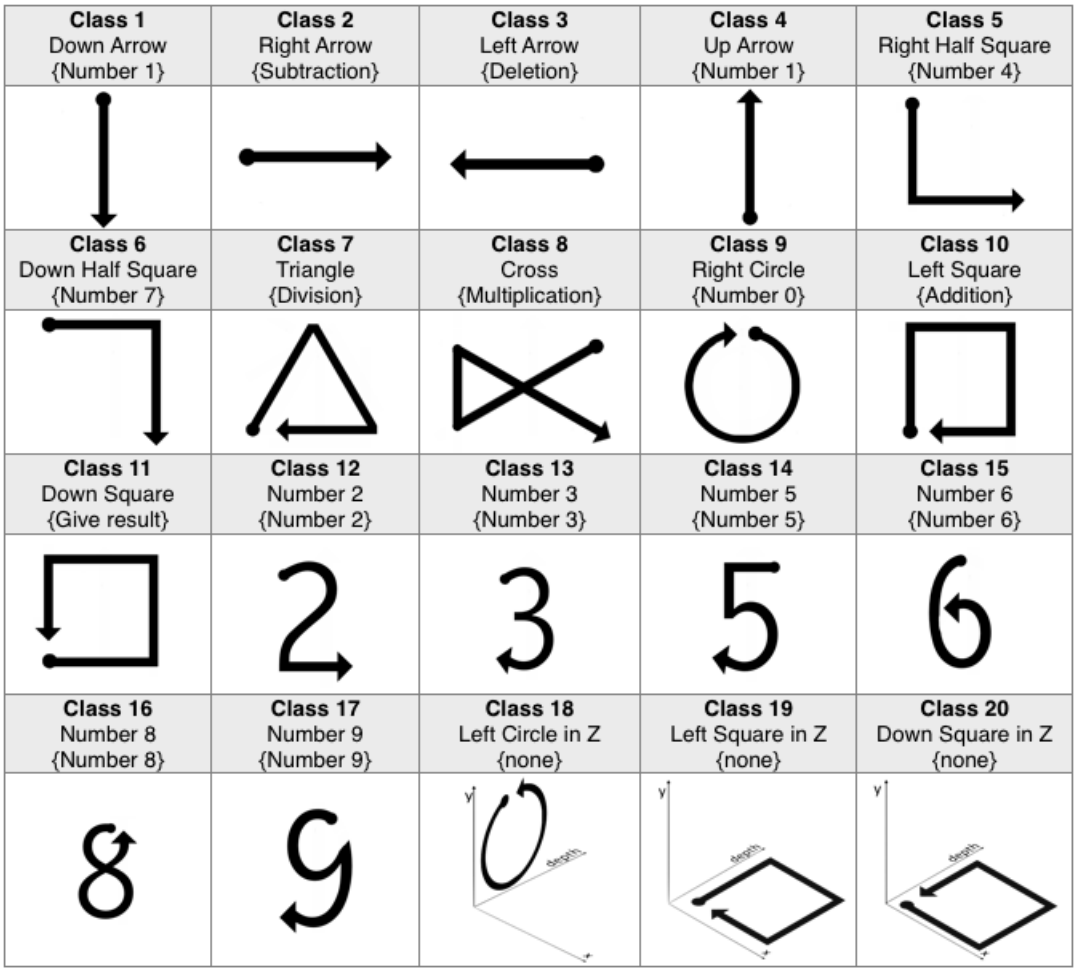


Figure 3.2. Gesture Set used in this project.

Note that; in order to access the built-in accelerometer of iPhone; iPhone operating system (iOS) has UIAccelerometer class under UIKit framework. The update frequency of the accelerometer can be configured and 3D acceleration data with timestamp can be received by using the shared UIAccelerometer object. As the iPhone moves, the built-in accelerometer reports linear acceleration changes along the primary axes in three-dimensional space.

3.2. General Model

We proposed an accelerometer based gesture recognition system, which consists of two initial parts; training (template generation) and classification (recognition process) parts. In the training part; key sequences, called templates, are generated; therefore this part is called *template generation*. Here, each gesture movement is captured by

iPhone built-in accelerometer and saved into a dataset. Then each individual gesture data in the dataset is validated, filtered, preprocessed, down-sampled and trained respectively to generate key data, called template data, of each gesture type. After templates generation operations, collected data is validated, filtered, down-sampled and trained in order to find best warping windows sizes and threshold values. The warping window sizes contain adjusted window size of each gesture type for each time t and it is calculated by considering the classification accuracy result. Threshold values determine the minimum and maximum DTW distances as a range for each template. Here, in the training part; all operations, except the data collection, are made by PC because of it's computational power. In the classification part; a candidate gesture data is tried to recognized according to trained templates; therefore this part is called *recognition process*. Here, user makes a movement with iPhone and it is captured by iPhone built-in accelerometer and saved temporary as a candidate data. Then the candidate gesture data is matched with each template using warping window sizes and threshold values and the closest template is selected as the classification result. The matching of two gesture data is based on Dynamic Time Warping (DTW) and LBKeogh algorithms, which determine the distance between two gesture data as a similarity metric [10, 11]. All operations in recognition process are done with iPhone 3GS as a new interface for the visually-impaired users. In brief, template generation is made once by PC to produce the essential data for recognition and recognition process is done by the users at their choice. As an introductory information, the legend of the template generation and recognition processes figures is given in Figure 3.3 below:

Figure 3.3. Legend used in template generation and recognition processes.

3.2.1. Template Generation

In the template generation processes, the collected data is used to generate template library, warping window sizes and threshold values of each gesture types. The templates are the key sequences for each gesture type in order to check the similarity

distance (DTW distance) with the candidate data and determine the closest template for the candidate. The Figure 3.4 is an abstraction of our model given below. Template generation is done once on a desktop machine, then the templates, warping window sizes and threshold values are stored in iPhone and used in recognition processes. There

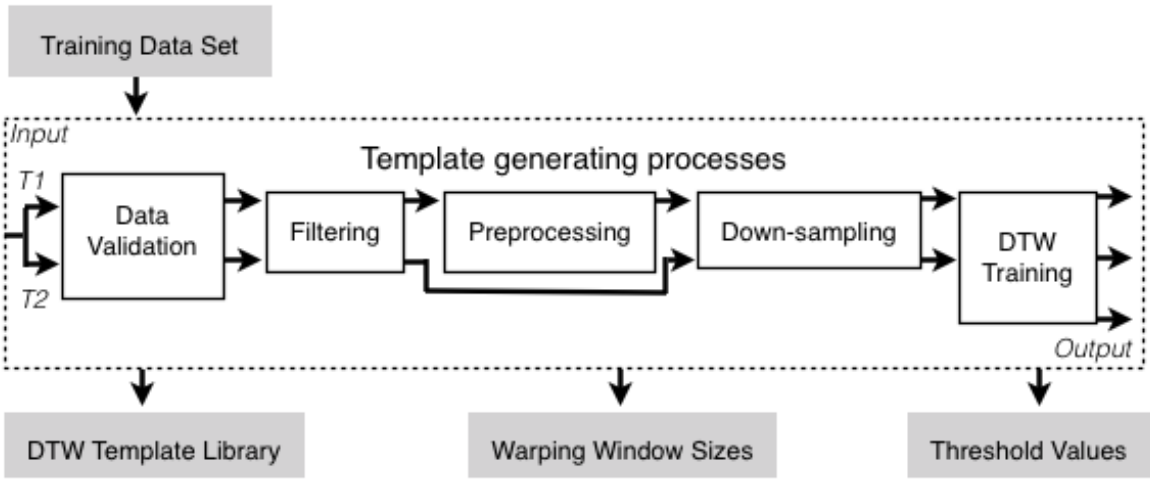


Figure 3.4. Template generating processes.

are two data flow in the system; one of them is used for generating template library ($T1$) and other one, called validation data set ($T2$), is used for generating the warping windows sizes and threshold values. The difference between training and validation set is the preprocessing step. In the preprocessing step, the gesture type information is need to process of each data (see: Preprocessing Section 3.6); therefore the validation data or any candidate data is not preprocessed. In addition, each procedure, except for preprocessing and DTW training procedure, given with boxes in Figure 3.4 gets one gesture data at one time. In the first data flow with training data set, the system generate template library of each gesture type; then in the second data flow with the validation data set, the system is used the template library to generate the warping window and threshold values. The general algorithm of template generation is given below.

```
Input: All collected data
Result: Template library, warping window sizes and threshold values are found
        and saved as library
begin
    Captured data gathering as a data set (see: Section 3.1)
    begin Training
        Validate the data (see: Section 3.4)
        Filter with low-pass filter (see: Section 3.5)
        Preprocess training data (see: Section 3.6)
        Down-sample data (see: Section 3.7)
        Train DTW; generating template library (see: Section 3.8)
    end
    begin Validation
        Validate the data (see: Section 3.4)
        Filter with low-pass filter (see: Section 3.5)
        Down-sample data (see: Section 3.7)
        begin Warping window generation
            Train DTW; finding best warping window sizes of each gesture type
            using validation data and template library (see: Section 3.8)
        end
        begin Threshold values generation
            Train DTW; finding threshold values of each gesture type using
            warping window sizes and template library (see: Section 3.8)
        end
    end
    Save values learned in algorithm as library.
end
```

Figure 3.5. Template generation algorithm.

3.2.2. Recognition Processes

In the recognition processes, single candidate gesture data is tried to classify by checking the DTW distances with template gestures of template library. Here, the distance is a metric of similarity, so the closest template gesture data means that it is the most similar gesture data with our candidate. Recognition processes are done on iPhone. The iPhone application has the templates, warping window sizes and threshold values found in template generation previously, so it is used for data collection and candidate data classification. The Figure 3.6 is an abstraction of our recognition procedures given below:

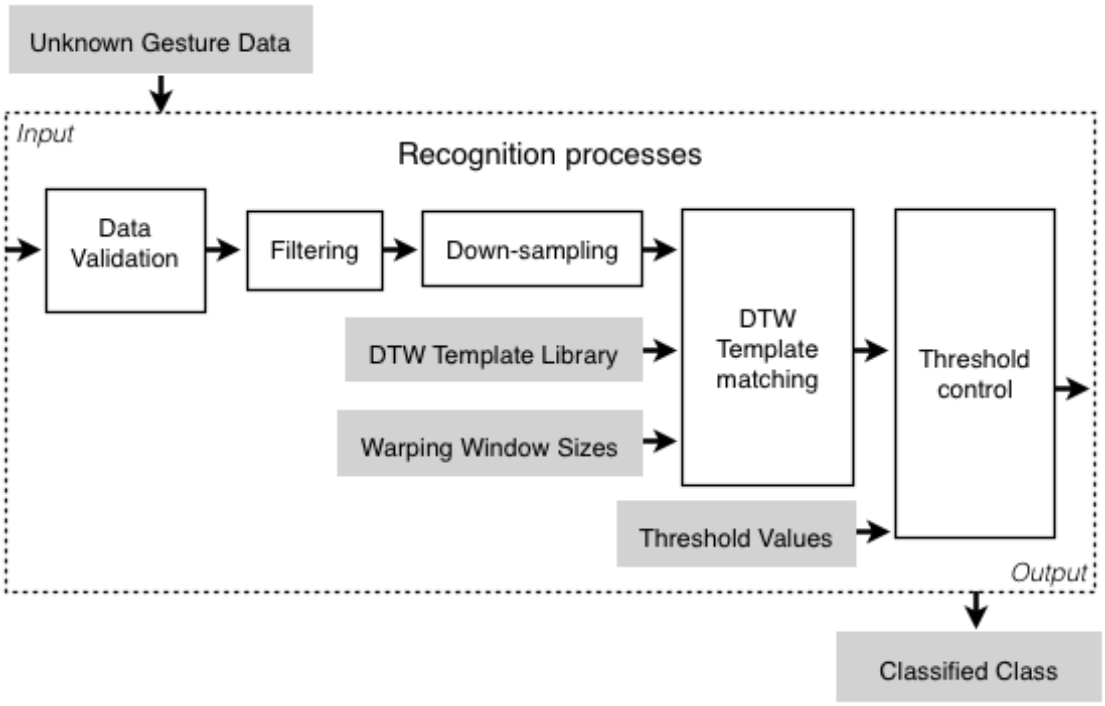


Figure 3.6. Gesture recognition processes.

Here, the captured user gesture data is used as candidate gesture data. In addition; DTW template library, warping window sizes and threshold values, which are found in template generation procedure, are used in the recognition steps. The DTW template matching step takes the down-sampled candidate data, DTW template library and warping window size values in order to find the closest template. The threshold control procedure takes the closest class information with threshold values and controls whether it is in range or not. Then gives the output as classification result.

The general algorithm is given below;

```
Input: C: Candidate Gesture Data
Output: i : index of template as classification result
begin
    Validate the data (see: Section 3.4)
    Filter with low-pass filter (see: Section 3.5)
    Downsample data (see: Section 3.7)
    Find closest template with DTW Template matching (see: Section 3.9.1)
    Control the found template with threshold values (see: Section 3.9.2)
    return Found and controlled template class value as classification result
end
```

Figure 3.7. Gesture recognition algorithm.

3.3. Theory

In this section, all definitions used in our model with their meaning and initial values are given as well as the assumptions.

3.3.1. Definitions

A captured gesture data has $x(k)$, $y(k)$ and $z(k)$ components where $k \in \{1, ..., K\}$, where K is the end time of captured user movement. These 3 time series can be represented as a $3 \times K$ matrix as $\mathbf{G}$, where the first, second and third row corresponds to $x(k)$, $y(k)$, $z(k)$ respectively. Therefore the gesture data is $\mathbf{G} \triangleq \{\overrightarrow{\mathbf{g}}(k)\}_{k=1}^{K}$ where k^{th} column of $\overrightarrow{\mathbf{g}}(k)$ is:

$$\overrightarrow{\mathbf{g}}(k) \triangleq \begin{bmatrix} x(k) \\ y(k) \\ z(k) \end{bmatrix} \tag{3.1}$$

In this study, we have $1,091$ gesture data with known gesture types. There are $M = 20$ different gesture classes given in Figure 3.2. These $1,091$ gesture data are almost evenly distributed to the classes. The class with smaller number of gesture is 48, while the one with the largest number of gesture is 55. We define $\mathcal{G} = \{\mathbf{G}_1, ..., \mathbf{G}_{1091}\}$ as the set of gestures and $\mathcal{M} = \{1, 2, ..., 20\}$ is the set of symbols. We defined a function:

$$\begin{aligned} \beta &\triangleq \mathcal{G} \rightarrow \mathcal{M} \\ \mathbf{G} &\mapsto m \end{aligned} \tag{3.2}$$

where m is the gesture type corresponding to gesture $\mathbf{G}$, so:

$$\mathcal{G}^m \triangleq \{\mathbf{G} \in \mathcal{G} | \beta(\mathbf{G}) = m\} \tag{3.3}$$

is the set of all gestures that corresponds to gesture type m. Note that, $\mathcal{G} = \bigcup_{m=1}^{20} \mathcal{G}^m$, which means that the collected gesture set is the union of the gestures of each gesture types. In addition, $\forall m \in \mathcal{M}$; $48 \leq |\mathcal{G}^m| \leq 55$ where $|\mathcal{G}| = 1091$.

3.3.2. Assumptions

- Assumption 1. All users hold the device as the same position and the device orientation is not changing dramatically while the gesture movement continues.
- Assumption 2. All users start their gesture movement by touching the screen and holding it till the movement finish. This assumption is made, because our proposed method does not contain any segmentation procedure, so the captured data should be segmented as a single gesture symbol. Here, we assumed that a visually-impaired user would like to use the recognition system on demand, so the instant continuous recognition system may lead an extra difficulty on usage of the system.
- Assumption 3. All users do not move and they stand in ground without any external force which may make noise to the accelerometer while the data is captured. Here, the frame of reference of the user is stationary, which means that

user is not an acceleration motion while the data is collected.

- Assumption 4. The built-in accelerometer of iPhone 3GS produces acceleration data with the same time interval Δt. In actual data, Δt is changing slightly because of some operating system delays.

3.4. Data Validation

The aim of data validation is eliminating some error conditions before we start to extract the DTW templates in order to increase the classification accuracy and template quality. There are three types of data validation used in this project. First one is the data validation during the gesture data collection. We noted the wrong gesture movements while participants make the data with iPhone. This step is done before the training start and the wrong movements are deleted from the data set. The other ones are validated each gesture data in the data set; according to gesture length and average amplitude respectively.

3.4.1. Gesture Length Data Validation

A gesture data $\mathbf{G}$ may be too short or too long. Any gesture data shorter than $K_{min} = 30$ or longer than $K_{max} = 205$ is rejected in the validation step. $K_{min} = 30$ is determined by the downsampling sample size N. $K_{max} = 205$ is determined by multiplying the maximum gesture data length among all gesture in training set data by 1.1 (110%), which is defined intuitively.
According to gesture length data validation;

$$LengthValidation(\mathbf{G}) \triangleq \begin{cases} valid, & \text{if } K_{min} < K_G < K_{max} \\ invalid, & \text{otherwise} \end{cases} \tag{3.4}$$

where K_G represents the length of a gesture data. Here, if a gesture data $\mathbf{G}$ is marked as invalid then it is deleted by system from the data set. This procedure is applied for

all gesture data set and each candidate gesture data entering the system.

3.4.2. Average Amplitude Data Validation

A gesture data **G** could be too weak or too strong for classification, which means that a user may make the gesture movement in slowly or fast. Such gestures are rejected. Here, amplitude is used for measuring the weakness or the strength of the gesture. The amplitude of gesture data **G** at time k is:

$$a(k) = \sqrt{x(k)^2 + y(k)^2 + z(k)^2}. \tag{3.5}$$

The average of the amplitude values of a gesture data **G** is defined as:

$$\mathbf{A}_{avg} \triangleq \frac{1}{K} \sum_{k=1}^{K} a(k) \tag{3.6}$$

where K is the length of gesture data **G**. Any gesture data, which has lower than $A_{min} = 0.95$ or greater than $A_{max} = 2.10$ is rejected in the validation step. Here, $A_{min} = 0.95$ is the minimum average amplitude value, which is determined by multiplying the minimum average amplitude among all gesture data in training set by 0.9 (90%). $A_{max} = 2.10$ is the maximum average amplitude value, which is determined by multiplying the maximum average amplitude among all gesture data in training set by 1.1 (110%), which is defined intuitively.
According to average amplitude data validation;

$$AmplituteValidation(\mathbf{G}) \triangleq \begin{cases} valid, & \text{if } A_{min} < A_G < A_{max} \\ invalid, & \text{otherwise} \end{cases} \tag{3.7}$$

where A_G represents the average amplitude value of a gesture data. Here, if a gesture data **G** is marked as invalid then it is deleted by system from the data set. This procedure is applied for all gesture data in data set and each candidate gesture data

entering the system.

3.5. Filtering Accelerometer Data

The aim of filtering is to eliminate the high frequency data, which is caused by user hand shaking or some noisy hand movements, from our gesture data and smooth the data, so we used a simple low-pass filtering (see: Low-Pass Filtering Section 2.6). There are some definitions used in low-pass filtering given below. F_{update}= 60 = The filtering update frequency which is equal to the frequency used in gesture capturing F_{cutoff}= 10 = The cutoff frequency used in filtering which is determined after some trials.

$\alpha = \frac{F_{cutoff}}{F_{cutoff}+F_{update}}$ = Filtering constant used in filtering (smoothing factor)

$\omega = (x, y, z)$ = The filtered gesture data at time t, which is used as a temporary variable in filtering.

This procedure is applied for all gesture data in data set and each candidate gesture data entering the system, and the algorithm of low-pass filter is given below;

```
Input: T : Training Data Set
begin
    foreach G ∈ T do
        G^filtered ← 0
        ω ← G(1)
        for t ← 1 to T_G do
            ω ← (G(t) × α) + (ω × (1.0 − α))
            G(t)^filtered ← ω
        end
        G ← G^filtered
    end
end
```

Figure 3.8. Low-Pass Filter algorithm.

3.6. Preprocess

The all gesture data, which are belong to same gesture type, may have three different properties under the certain assumptions given at the beginning. i) user may make a gesture movement slower or faster than other gesture data. ii) user hold the iPhone with a certain angle in x, y and z axes even if he/she tries to hold it face to him/her. iii) user may change the iPhone position while gesture is captured. All three properties changes the shape of gesture data sequences, which affects the quality of templates generated by using all same type gesture data; therefore, the aim of preprocessing is close up each gesture data in same gesture type. Here, we can eliminate the first and the second situations by making the preprocessing. However, the position information at time t is not known, so we can't eliminate the third problem. Firstly, the average mean value is found among the gesture data in same type. Then each gesture data is shifted in y to have the same mean value with the average one. After shifting in y; all the same type gesture data are aligned in terms of average mean value. This procedure eliminates the differences among same type gesture data caused by initial phone position (see: Figure 3.9b).

Secondly, the oscillation rate around mean is tuned with average variance value to get same oscillation rate of each same type gesture data. This procedure eliminates the differences among same type gesture data caused by user hand movement sharpness, which affects the amplitude of gesture sequences (see: Figure 3.9c).

In brief, there are two types of preprocessing made for this project; one of them is mean adjustment and other one is variance adjustment of gesture data of same class in order to increase the quality of template (and also increase the accuracy rate). The Figure 3.9 shows the effects the preprocessing procedures.

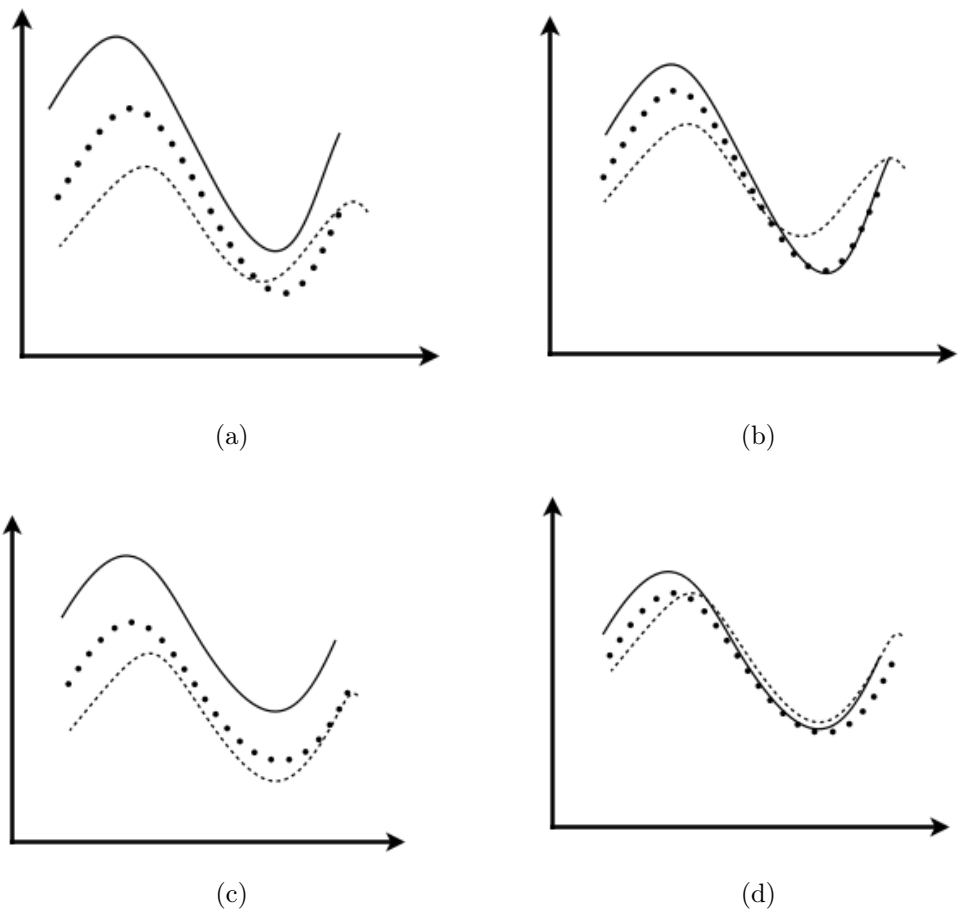


Figure 3.9. Illustration of preprocess effects. (a) Three raw gesture data in x axis. (b) After aligned in terms of average mean value. (c) After aligned in terms of average variance value. (d) After aligned in terms of mean and average variance value.

3.6.1. Mean Adjustment

In mean adjustment, we make all gesture data having same mean value of the same gesture type in training set. In order to equalize the mean values of gestures of the same gesture type, the average mean value of the given gesture type is calculated. Then the difference value between average mean value and the current mean value of given gesture is added to the given gesture data.

The mean value of a gesture data is defined as:

$$\mu \triangleq \begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix} \tag{3.8}$$

where mean value of each dimension is calculated as:

$$\mu_x \triangleq \frac{1}{K}\sum_{i=1}^{K} x(i), \qquad \mu_y \triangleq \frac{1}{K}\sum_{i=1}^{K} y(i), \qquad \mu_z \triangleq \frac{1}{K}\sum_{i=1}^{K} z(i) \tag{3.9}$$

where K is the length of each time series. The average mean of all gesture data of the same gesture type m in training set is defined as:

$$\overline{\mu}^m \triangleq \begin{bmatrix} \overline{\mu}_x^m \\ \overline{\mu}_y^m \\ \overline{\mu}_z^m \end{bmatrix} \tag{3.10}$$

where average mean value of each dimension is defined as:

$$\overline{\mu}_x^m \triangleq \frac{1}{|\mathcal{G}^m|}\sum_{\mathbf{G}\in\mathcal{G}^m} \mu_{\mathbf{G},x}, \quad \overline{\mu}_y^m \triangleq \frac{1}{|\mathcal{G}^m|}\sum_{\mathbf{G}\in\mathcal{G}^m} \mu_{\mathbf{G},y}, \quad \overline{\mu}_z^m \triangleq \frac{1}{|\mathcal{G}^m|}\sum_{\mathbf{G}\in\mathcal{G}^m} \mu_{\mathbf{G},z} \tag{3.11}$$

where $|\mathcal{G}^m|$ is the number of gesture data of gesture type m in the training set. After defining mean of a gesture and average mean of all gestures in same gesture type; the mean adjustment operation for any gesture data $\mathbf{G}$ of gesture type m is defined as:

$$\mathbf{G}^* \triangleq \mathbf{G} - \mu_{\mathbf{G}} + \overline{\mu}^m \tag{3.12}$$

where $\mathbf{G}^*$ is the mean adjusted gesture data.

3.6.2. Variance Adjustment

In variance adjustment, we make all gesture data having the same variance value of the same gesture type in training set. In order to equalize the variance values of gestures of the same gesture type, the average variance value of the given gesture type is calculated. Then the average variance value is divided by the current variance value of given gesture and multiplied by the difference between gesture data point and mean value and added the mean value to the result in order to have the same variance value among gestures of the same gesture type.

The variance value of a gesture data is defined as:

$$v \triangleq \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{3.13}$$

where variance value of each dimension is calculated as:

$$v_x \triangleq \frac{1}{K}\sum_{i=1}^{K}(\mathbf{G}_x - \mu_x)^2, \quad v_y \triangleq \frac{1}{K}\sum_{i=1}^{K}(\mathbf{G}_y - \mu_y)^2, \quad v_z \triangleq \frac{1}{K}\sum_{i=1}^{K}(\mathbf{G}_z - \mu_z)^2, \tag{3.14}$$

where K is the length of each time series and μ is the mean value of the given gesture with given axis. The average variance of all gesture data of the same gesture type m in training set is defined as:

$$\overline{v}^m \triangleq \begin{bmatrix} \overline{v}_x^m \\ \overline{v}_y^m \\ \overline{v}_z^m \end{bmatrix} \tag{3.15}$$

where average variance value of each dimension is defined as:

$$\overline{v}_x^m \triangleq \frac{1}{|\mathcal{G}^m|}\sum_{\mathbf{G}\in\mathcal{G}^m} v_{\mathbf{G},x}, \quad \overline{v}_y^m \triangleq \frac{1}{|\mathcal{G}^m|}\sum_{\mathbf{G}\in\mathcal{G}^m} v_{\mathbf{G},y}, \quad \overline{v}_z^m \triangleq \frac{1}{|\mathcal{G}^m|}\sum_{\mathbf{G}\in\mathcal{G}^m} v_{\mathbf{G},z}, \tag{3.16}$$

where $|\mathcal{G}^m|$ is the number of gesture data of gesture type m in the training set. After defining variance of a gesture and average variance of all gestures in same gesture type; the variance adjustment operation for any gesture data $\mathbf{G}$ of gesture type m is defined as:

$$\mathbf{G}^{**} \triangleq \mu_{\mathbf{G}} + \left(\sqrt{\frac{\overline{v}^m}{v_{\mathbf{G}}}} \times (\mathbf{G} - \mu_{\mathbf{G}}) \right) \tag{3.17}$$

where $\mathbf{G}^{**}$ is the variance adjusted gesture data.

3.7. Downsampling

The aim of downsampling is getting gesture data with same length without having dependency on time, which means that if a user makes a gesture slowly where as another user makes the same gesture in a fast way, we can not overlap two data and can't generate a quality templates making increase in accuracy. In this study, we use average downsampling (see: Section 2.6.2.1). In data collection part, the gesture data is sampled to 60 Hz, which is the running frequency of iPhone built-in accelerometer. Here, the downsampling rate is 30 Hz, which is a popular frame rate used in high definition video and determined intuitively, but the segmentation is assumed to 1 second, so sample size becomes $N = 30$. The selection group length is defined as $D = K/N$ where K is the length of a gesture data $\mathbf{G}$ on any axis. After defining selection group length, we divide each sequence of gesture data in x, y and z axes into groups. Then, we take the average of each group and put them into new sequence, which is down-sampled sequence. Let's define average (mean) value of a gesture data with given *start* and *end* time, where $(start < end)$ and $(start > 0)$ and $(end \leq T)$;

$$\mu(start, end) \triangleq \begin{bmatrix} \mu_x(start, end) \\ \mu_y(start, end) \\ \mu_z(start, end) \end{bmatrix} \tag{3.18}$$

where segmented mean value of each dimension is defined as:

$$\begin{aligned}
\mu_x(start, end) &= \frac{1}{(end - start)} \sum_{i=start}^{end} x(i), \\
\mu_y(start, end) &= \frac{1}{(end - start)} \sum_{i=start}^{end} y(i), \\
\mu_z(start, end) &= \frac{1}{(end - start)} \sum_{i=start}^{end} z(i)
\end{aligned} \tag{3.19}$$

After defining mean value of a gesture data with given *start* and *end* time, the downsampling algorithm is defined as given below:

```
Input: T : Training Data Set
begin
    G^downsampled ← 0
    foreach G ∈ T do
        for time t ← 1 to N − 1 do
            start ← t × D
            end ← (t + 1) × D
            G(t)^downsampled ← μ(start, end)
        end
        G ← G^downsampled
    end
end
```

Figure 3.10. Downsampling algorithm.

3.8. DTW Training Processes

3.8.1. DTW Training

The aim of DTW training is learning the template sequences, threshold and warping window size values from training data set which make the highest accuracy on the validation data. DTW training takes two data set; one of them is the preprocessed training data set and other one is the validation data set (see: Figure 3.6). The general algorithm and detailed definitions are given below. Note that, after downsampling our time limit is now N, which is the sample size.

```
Input: T1 : Training Data Set, T2 : Validation Data Set
begin
    Generate DTW Template Library TL from training data set T1.
    Finding best warping window sizes of validation data set T2 using DTW
    Template Library
    Finding threshold values DTW distances of validation dataset T2 with
    DTW Template Library TL and warping window sizes.
end
```

Figure 3.11. General DTW Training algorithm.

3.8.1.1. Template Library Generation. In template library generation, we calculate the mean value of gestures, which are the same gesture type, in training set **T1** for each time index. Then for each gesture type a gesture data is generated, called query or template, within the mean value for each time index. Template is represented by **Q** and defined as:

$$\mathbf{Q} \triangleq \{\overrightarrow{\mathbf{q}}(n)\}_{n=1}^{N} \tag{3.20}$$

where template data $\mathbf{Q}$ of any time index n is:

$$\overrightarrow{\mathbf{q}}(n) \triangleq \begin{bmatrix} x(n) \\ y(n) \\ z(n) \end{bmatrix} \tag{3.21}$$

The value of template $\mathbf{Q}$ of gesture type m for each axis on any time n is defined as:

$$\mathbf{q}_x(n) \triangleq \frac{1}{|\mathcal{G}^m|} \sum_{\mathbf{G} \in \mathcal{G}^m} \mathbf{G}_x(n), \quad \mathbf{q}_y(n) \triangleq \frac{1}{|\mathcal{G}^m|} \sum_{\mathbf{G} \in \mathcal{G}^m} \mathbf{G}_y(n), \quad \mathbf{q}_z(n) \triangleq \frac{1}{|\mathcal{G}^m|} \sum_{\mathbf{G} \in \mathcal{G}^m} \mathbf{G}_z(n), \tag{3.22}$$

where $|\mathcal{G}^m|$ is the number of gesture data of gesture type m in the training set. According to template $\mathbf{Q}$ definition on any time index n, we can generate our DTW template library $\mathbf{TL}$ by calculating each template values for each gesture type m, which is given below:

$$\mathbf{TL} \triangleq \begin{bmatrix} \mathbf{Q}^1 \\ . \\ . \\ . \\ \mathbf{Q}^M \end{bmatrix} \tag{3.23}$$

where total gesture type value $M = 20$. After defining DTW template library, now we can define the lower limit $\mathbf{L}$ and upper limit $\mathbf{U}$ matrices for each model m which are used in LBKeogh distance calculations (see: LBKeogh distance in Section 2.7.3).

$$\mathbf{L} \triangleq \{\overrightarrow{\mathbf{l}}(n)\}_{n=1}^{N}, \qquad \mathbf{U} \triangleq \{\overrightarrow{\mathbf{u}}(n)\}_{n=1}^{N} \tag{3.24}$$

where

$$\overrightarrow{\mathbf{l}}(n) \triangleq \begin{bmatrix} l^1(n) \\ . \\ . \\ . \\ l^M(n) \end{bmatrix}, \qquad \overrightarrow{\mathbf{u}}(n) \triangleq \begin{bmatrix} u^1(n) \\ . \\ . \\ . \\ u^M(n) \end{bmatrix} \tag{3.25}$$

and all lower and upper bound has three time series $\mathbf{l}^m(i) = (x(i), y(i), z(i))$ and $\mathbf{u}^m(i) = (x(i), y(i), z(i))$; so our lower and upper bounds are given for the gesture type m on x axis by using template q;

$$\begin{aligned} l(i)_x &\triangleq \min\{q_x(j)|j \in [\max\{1, i-R\}, min\{N, i+R\}]\} \\ u(i)_x &\triangleq \max\{q_x(j)|j \in [\max\{1, i-R\}, min\{N, i+R\}]\} \end{aligned} \tag{3.26}$$

where the range is $R = 3$, which is determined after some trials.

<u>3.8.1.2. Best Warping Window Size.</u> In this project, we used dynamic warping window sizes defined in Ratanamahatana and Keogh's work [11]. Validation data set **T2** is used to find warping window sizes, which produce highest accuracy rate on classification. Firstly, we define warping window array as:

$$\mathbf{W} \triangleq \{\mathbf{w}(n)\}_{n=1}^{N} \tag{3.27}$$

and initialize the all values to 1, then we calculate *distance metric* and save it. The distance metric is given below;

$$DistanceMetric \triangleq (TPD/FND) \times (FN/TP) \tag{3.28}$$

where TPD is the total DTW distance of true positive classification, FND is the total DTW distance of false negative classification, FN is false negative classification number and TP is true positive classification number among the training set by checking each gesture data in training set and finding the closest template.

Secondly, we increment the values on warping window array **W** by 1 and calculate the distance metric and compare with the previous one. If distance metric is lower than the previous one, it means that our system is improved and we can continue to increase the warping size until reaching the maximum limit N, which is the same as sample size. However, if the distance metric is greater than the previous one, then we divide the warping window array into two part and follow the given procedure for each part. This is a recursive procedure called *forward search* and the dividing warping size array continues till the length of part becomes 1 (lower bound) or there is no more increment value (upper bound). However, defining the minimum limit as 1 produce a danger of over-fitting, so we define a lower bound threshold limit for warping array part length as:

$$\left\lceil \sqrt{(}N)/2 \right\rceil = 3 \tag{3.29}$$

In addition, after finishing one forward search we can save the warping window size array and restart the algorithm from the saved array. In our study, the maximum iteration for each forward search algorithm is defined as $I = 3$, which is found empirically. After 3 iteration, the best warping window array did not change in our experiments. The Figure 3.12 gives the illustration of the forward search for finding best warping window sizes. Here, $h(1)$ and $h(2)$ represent the distance metrics of each warping array condition. In addition, the general algorithm is given in Appendix in Figure C.2 with other used algorithms.

Here, in order to decrease the computational time for finding the best warping window size array; LBKeogh distance, defined in Section 2.7.3, is used while comparing

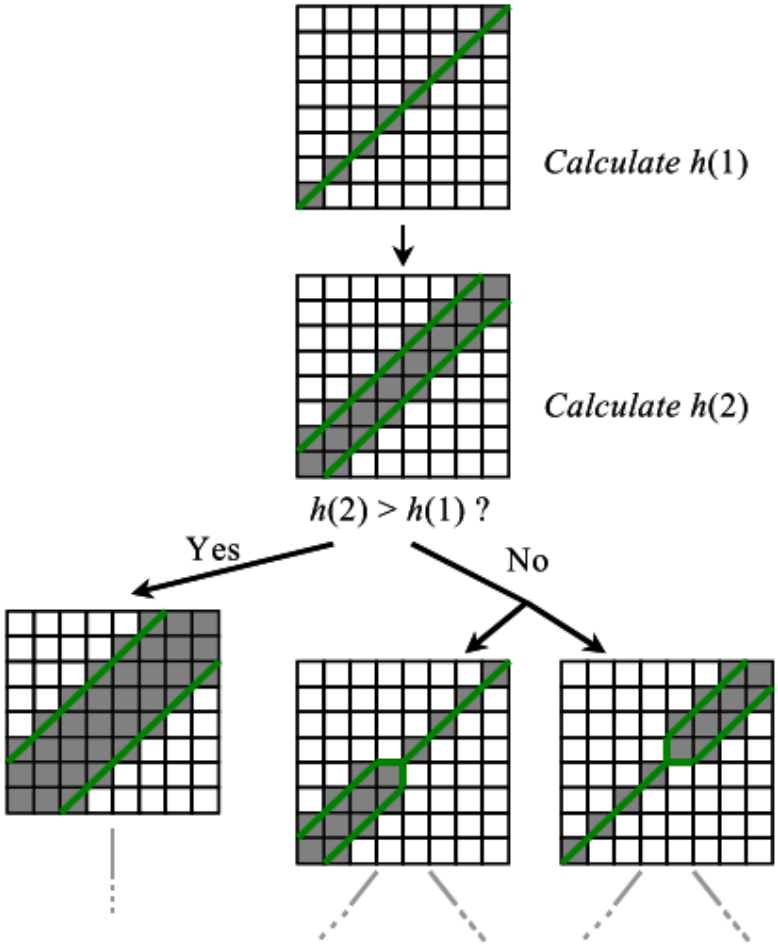


Figure 3.12. Forward search algorithm from Ratanamahatana and Keogh's work [11].

two gesture data. LBKeogh distance is calculated by adding each time series (each dimension) distances (see: Appendix C Figure C.5).

$$LBKeogh(G1, G2) \triangleq \sum_{i=1}^{3} LBKeogh(G1_i, G2_i) \tag{3.30}$$

where i is the index of each dimension x, y and z respectively. In addition, the DTW distance, defined in Section 2.7.2, is used while calculating the distance between two gesture data. DTW distance is calculated by adding each time sequence (each dimension) distances (see: Appendix C Figure C.6).

$$DTWDistance(G1, G2) \triangleq \sum_{i=1}^{3} DTWDistance(G1_i, G2_i) \tag{3.31}$$

where i is the index of each dimension x, y and z respectively.

3.9. Recognition Processes

In order to classify an unknown gesture data, recognition processes (see: Figure 3.6) are followed, given in Figure 3.7.

3.9.1. Finding Closest Class with Threshold Control

The closest gesture type is found using DTW template library, warping window sizes and threshold values found in the template generation. Firstly, an unknown gesture as candidate gesture data $\mathbf{C}$ is taken and calculated all DTW distances between each template in DTW template library. Then the closest template in the given threshold range for each dimension x, y and z becomes the prediction (classification) result, if there isn't any template in the range the unknown gesture data is marked as unclassified gesture. The algorithm is given below:

3.9.2. Threshold Control

An unknown gesture data $\mathbf{C}$ may be misclassified if there are some other closer templates than the actual one, which is not a general condition. This problem, called misclassification problem, can be caused because of various different reasons such as; low quality template set, noisy candidate data etc. It can not be eliminated directly but reduced by defining a range of DTW distance values for each gesture type m, which are called threshold values. The minimum threshold values $\Phi_{min} \triangleq \{\phi_{min}(m)\}_{m=1}^{M}$ and maximum threshold values $\Phi_{max} \triangleq \{\phi_{max}(m)\}_{m=1}^{M}$ are defined according to DTW distances between each validation data and the corresponding template. Here, $\phi_{min}(m)$ is the minimum DTW distance among all training data in gesture type m with template $\mathbf{Q}^m$ and $\phi_{max}(m)$ is the maximum DTW distance among all training data in gesture type m with template $\mathbf{Q}^m$.

```
Input: C: An unknown gesture data, TL : Template Library
Output: PredictedIndex : The closest gesture type index value according to
        DTW template library
begin
    PredictedIndex ← -1
    Distance ← ∞
    for m ← 1 to M = 20 do
        if DTW(Q^m,C)< Distance ∧ ThresholdControl(Q^m,C) then
            Distance ← DTW(Q^m,C)
            PredictedIndex ← m
        end
    end
    return PredictedIndex
end
```

Figure 3.13. Finding Closest Class Index algorithm.

According to threshold control;

$$ThresholdControl(\mathbf{Q}^m, \mathbf{C}) \triangleq \begin{cases} Invalid, & \text{if } DTW(\mathbf{Q}^m, \mathbf{C}) < V_{tolerance} \times \phi_{min}(m) \\ Invalid, & \text{if } DTW(\mathbf{Q}^m, \mathbf{C}) > (1 + V_{tolerance}) \times \phi_{max}(m) \\ Valid, & \text{otherwise} \end{cases} \tag{3.32}$$

where $V_{tolerance} = 0.5$ is tolerance value, which is determined after some trials. Here, if DTW distance between unknown gesture data $\mathbf{C}$ and template $\mathbf{Q}^m$ is not in range then it is not used in classification. This procedure is applied for all gesture data given by the system for classification.

3.10. Paramater Tuning

In this study, we made a brute-force search to find best parameters giving the highest accuracy. Firstly, minimum and maximum range values are defined for each parameters used in this project as well as the increment value intuitively. Secondly, all parameters are set the minimum range value. Then, for each increment value of each parameter; the classification accuracy is checked to find the maximum classification accuracy. This operation is very costly according to CPU time, so some parameters are defined in a very low range or high increment level.

3.11. Experimental Results

3.11.1. Implementation Information

All algorithms given in this thesis were implemented and packed as framework. In the framework, all the parameters can be configured dynamically as parameter tuning. In addition to the algorithms given with this study, there are numerous algorithms in the framework, which are not mentioned in thesis. In thesis, the algorithms caused the best classification accuracy is given with optimized parameter values. All implemented algorithms are given below:

- Filtering Algorithms. Low-pass, high-pass and band-pass filtering algorithms [56].
- Dimensional Reduction. Fast Fourier Transformation (FFT) [28].
- Clustering Algorihtms. K-Mean, K-Medoid, fixed amplitude, dynamic amplitude and dynamic split clustering, dynamic trajectory angle quantization [57–59].
- Sampling Algorithms. Simple, average, median and middle down-sampler.
- Classifier Model. Markov model (observable HMM as MM), HMM, 3D HMM and DTW models [11, 29, 60].
- Cross Validation. K-Fold, Random Subsampling and leave-one-out algorithms [61].

The implementation of our system consist of two parts. One of them is the computer based part and other one is the iPhone application. Here, our proposed

system has been implemented as a framework called 'GestureRecognition' in order to reuse the code in the iPhone application[2] . Gestures can be freely collected by iPhone application, then they should be saved in computer. The application for computer, which uses the 'GestureRecognition' framework, finds the optimal template library, warping window sizes and threshold values. The optimal condition satisfaction may last longer than expectation, in our case; it took 2 days with 2.53Gz Core 2 Duo, 4GB Ram Mac Pro for 1062 gesture data with 20 different gestures. Then the application produces an XML file containing optimum values for gesture recognition system which is used by our iPhone application. After this point, we can put our XML file to our iPhone application and test the system by making a gesture.

3.11.2. Experimental Information

In this study; two experiments were made for validating our recognition system. In the first experiment; training set were collected for 20 different gesture from 15 participant, who do not have any disability, with 55 samples for each gesture types (totally 1100 gesture samples). In the collection stage, we taught the gesture set to each participant before data capturing and did not interrupt on data capturing. However, some participants made wrong gesture movement involuntary, which was noted at that time and deleted from the gesture set. After this simple validation operation, 1091 samples were in the data set. However, in the data validation step, the system eliminated 29 gesture data, so 1062 gesture data were used totally. After data collection, our system was trained and validated with leave-one-out cross validation.

In the second experiment, we made a gesture based talking calculator with 15 different gestures by using proposed system (implemented framework) and first experiment's training results. Here, Flite text-to-speech library and its wrapper by Sam Foster was used [62, 63]. In order to test our system with the calculator; 40 calculations, containing totally 180 characters, were defined given in Table 3.1. Then 4 visually-impaired people tested the system by making defined calculations using our

[2]Thesis web page (`http://soslab.cmpe.boun.edu.tr/doku.php/wiki/theses/abc`) contains framework and the details of this thesis.

calculator for an adaptation period of 7 days, here the number of days is determined intuitively. In each day, all participants calculation data were collected in order to check the learning curve and the accuracy of the classifications.

Table 3.1. Calculations used in second experiment.

6+9=	72−4=	7+9=	1−24=	5+1=	8−9=	90+7=	3−6=	8+3=	56−2=
1/50=	3×8=	30/5=	8×6=	40/2=	76×3=	8/24=	1×5=	10/4=	7×9=
2+0=	6−7=	4+5=	8−7=	67+4=	1−9=	30+4=	1−5=	81+9=	30−5=
58/9=	31×4=	6/13=	90×2=	5/2=	3×80=	2/8=	67×9=	2/6=	4×7=

3.11.3. Results

In the first experiment, 20 different gestures were collected and classified with average of 96.7% recognition accuracy, which is user-independent classification result. The data set contains 1062 data totally and leave-one-out cross validation is used on validation. The confusion matrix of the first experiment is given in Table 3.2. Moreover, minimum and maximum DTW distances (threshold values) and warping windows size of each gesture type are given in the Appendix part Chapter B.

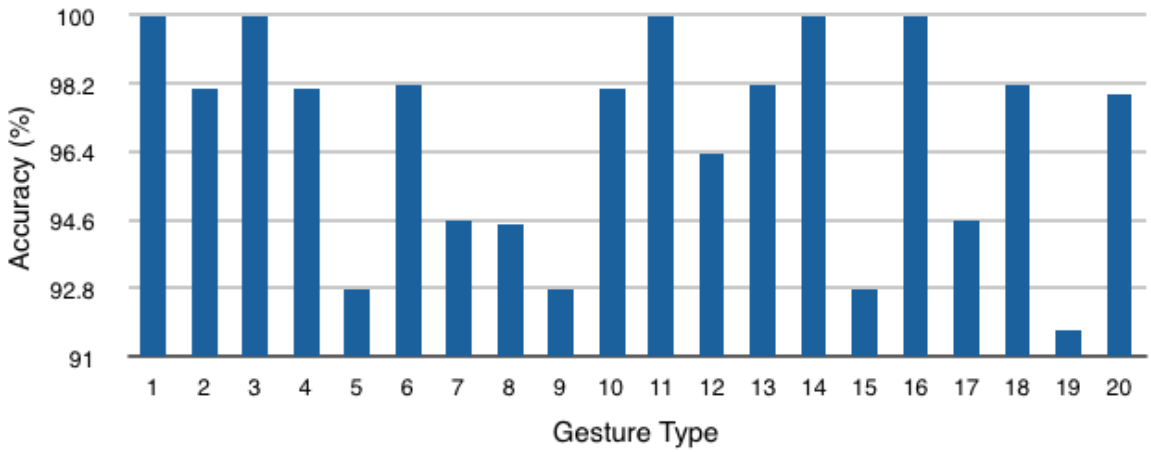


Figure 3.14. Accuracy rate per symbol from experiment 1.

In the second experiment, we got 95.5% recognition accuracy in the last day, which is user-independent classification result. The data set contains 15 different gestures with

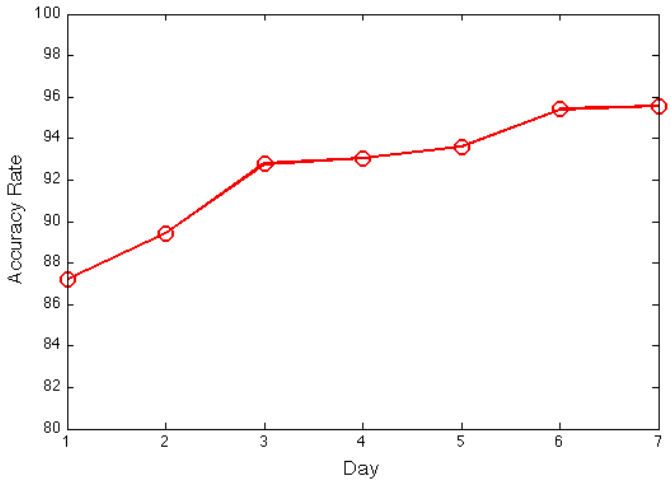
720 data totally (180 samples from 4 visually-impaired people). The experimental data was recorded day by day and the daily average recognition accuracy (learning curve), daily recognition accuracy per person and per letters are given in Figure 3.15.

Lastly, the effects of each procedure in the system on the classification result is calculated and given in the Appendix A with Table A.1 and Figure A.1.

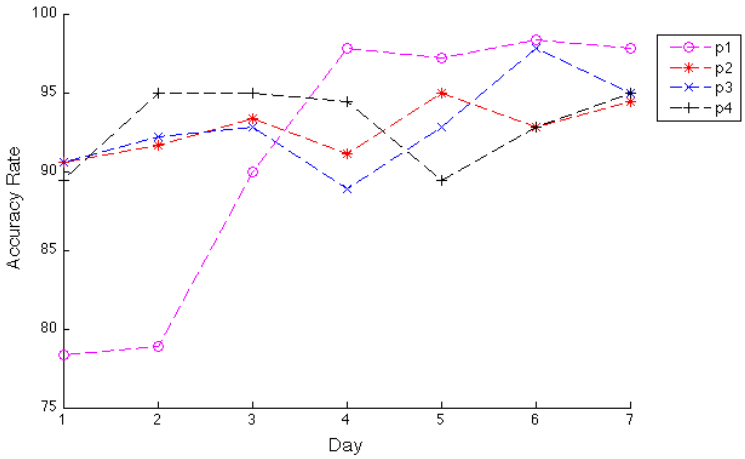
In addition to all findings in experiments, interestingly, all participants of experiment 2 were color-graphemic synesthesia, which is a neurologically-based condition in which letters or numbers are perceived as inherently colored and they were unaware about it during the experiment [64].

Table 3.2. Confusion matrix of experimental result.

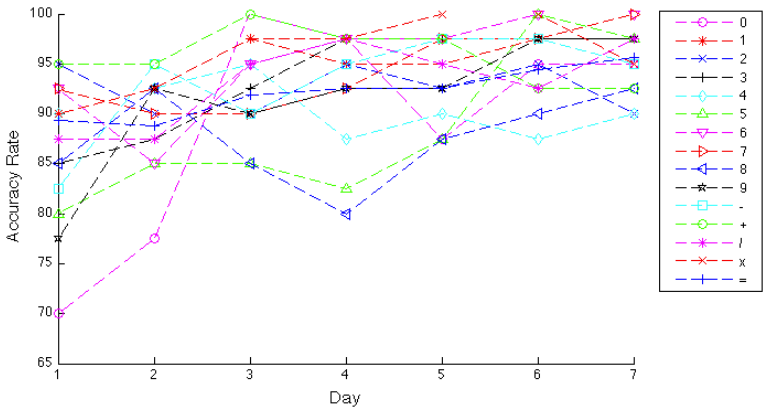
Actual Class Numbers \ Predicted Class Numbers	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	sum	TP	FP	FN	Pr %	Rc %
1	53	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	53	0	0	100	100
2	0	49	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	51	49	0	2	100	96.08
3	0	0	54	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	54	0	0	100	100
4	0	0	0	51	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	52	51	0	1	100	98.08
5	0	0	0	0	51	0	0	0	3	0	0	0	0	0	0	0	0	1	0	0	55	51	3	4	94.44	92.73
6	0	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	1	0	0	0	55	54	1	1	98.18	98.18
7	0	0	0	0	0	0	52	0	0	1	0	0	0	0	0	0	0	0	1	1	55	52	3	3	94.55	94.55
8	0	0	0	0	1	0	0	51	1	0	0	0	0	0	0	0	1	0	0	0	54	51	1	3	98.08	94.44
9	0	0	0	0	0	0	0	1	51	0	0	0	0	0	0	1	0	0	0	2	55	51	4	4	92.73	92.73
10	0	0	0	0	0	0	1	0	0	51	0	0	0	0	0	0	0	0	0	0	52	51	1	1	98.08	98.08
11	0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	53	53	0	0	100	100
12	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	1	0	0	0	1	55	53	0	2	100	96.36
13	0	0	0	0	1	0	0	0	0	0	0	0	54	0	0	0	0	0	0	0	55	54	0	1	100	98.18
14	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	0	0	0	0	0	53	53	3	0	94.64	100
15	0	0	0	0	1	0	0	0	0	0	0	0	0	0	51	0	0	0	2	1	55	51	0	4	100	92.73
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	51	0	0	0	0	51	51	3	0	94.44	100
17	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	52	0	0	0	55	52	4	3	92.86	94.55
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	53	0	1	54	53	2	1	96.36	98.15
19	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	1	0	44	0	48	44	4	4	91.67	91.67
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	46	47	46	6	1	88.46	97.87
sum	53	49	54	51	54	55	55	52	55	52	53	53	54	56	51	54	56	55	48	52	1062	1027	35	35	96.72	96.72



(a)



(b)



(c)

Figure 3.15. Second Experiment's Result. (a) Daily Average Accuracy. (b) Daily Average Accuracy Per Person. (c) Daily Average Accuracy Per Symbol.

4. CONCLUSION AND FUTURE WORK

We have introduced an optimized accelerometer-based gesture recognition on touch-screen smartphones with accelerometer for visually-impaired people as well as its implementation as gesture based talking calculator.

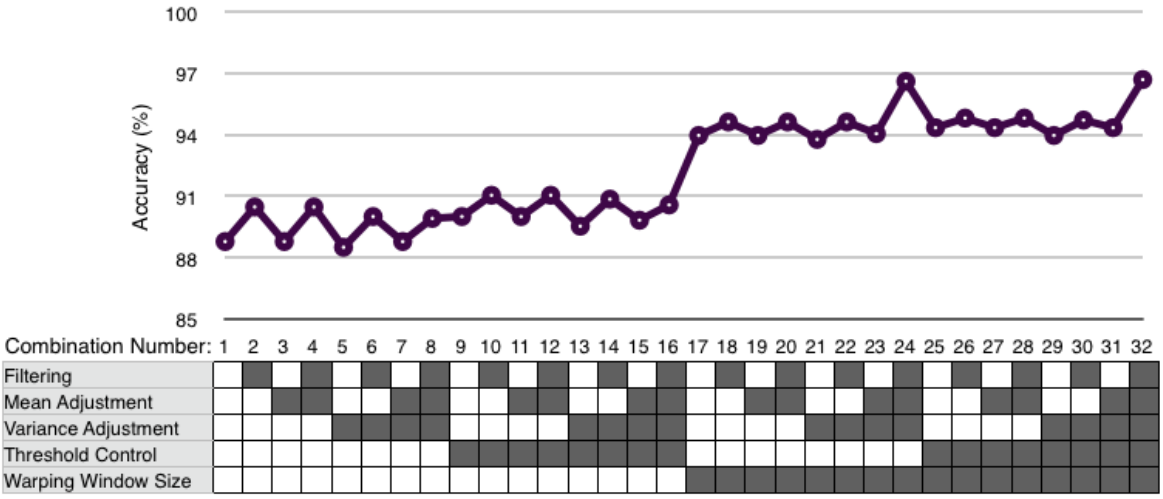
This kind of gesture recognition systems can be used in anywhere between human-computer interaction occurs. They may be used in games or in simulations in order to get and analyze the information of interactions of human with human and with computer; for example a designed war game using the hand gestures can give better understanding the work of gestures under-stress [65].

In our experiments, some assumptions were made on gesture collection in order to increase our accuracy rate such as holding phone as face to user. However, the system is totally user-independent and it doesn't have limitations on gesture collection. The only and main limitation is that; a gesture data should be captured first, then classified. The system doesn't check the data while user making gesture movement. Here, a participant should press the screen to save the accelerometer data till touching up from the screen. Making our proposed system to an continuous-gesture recognition system, which segments the data instantly and tries to classify each segment, may be a good goal for future works.

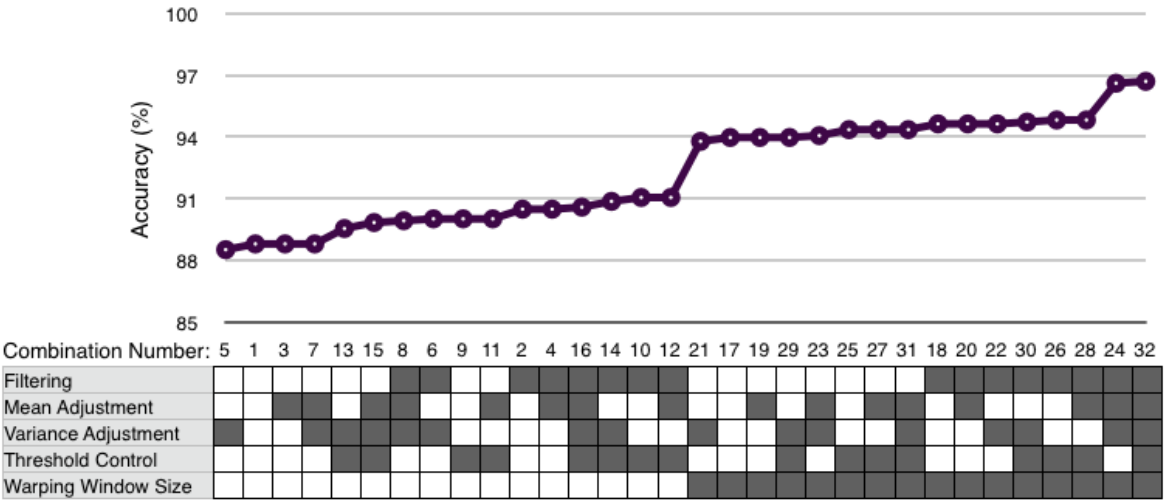
In order to increase the accuracy rate, a gesture set can be determined by increasing the dissimilarity of gestures from each other while considering user ability and comfort during writing. In addition, some additional sensor's data (e.g. gyroscope) can be added to the system while preprocessing the data in order to get more accurate result. In this study, templates are generated directly from simple preprocessed training data set and the preprocessing steps are defined intuitively; therefore a template quality metric may be defined considering gesture data features and possible noises in the data.

In summary, our proposed gesture recognition system provides an outstanding performance in terms of user-independent recognition accuracy, experimental results of end-users and variety of gesture set when compared to other systems in literature.

APPENDIX A: EFFECTS OF PROCEDURES



(a) Effects of procedures on classification accuracy ordered by combination number



(b) Effects of procedures on classification accuracy ordered by accuracy rate

Figure A.1. Effects of procedures on classification accuracy. Here, the grey boxes show the procedures that takes place (on), while white boxes show the procedures that does not take place (off).

Table A.1. Effects of procedures on classification accuracy.

Combination No \ Procedure Name	Filtering	Preprocess Mean Adjustment	Preprocess Variance Adjustment	Threshold Control	Warping Window Sizes	Accuracy
1	OFF	OFF	OFF	OFF	Constant : 3	88.79%
2	ON	OFF	OFF	OFF	Constant : 3	90.48%
3	OFF	ON	OFF	OFF	Constant : 3	88.79%
4	ON	ON	OFF	OFF	Constant : 3	90.48%
5	OFF	OFF	ON	OFF	Constant : 3	88.51%
6	ON	OFF	ON	OFF	Constant : 3	90.01%
7	OFF	ON	ON	OFF	Constant : 3	88.79%
8	ON	ON	ON	OFF	Constant : 3	89.92%
9	OFF	OFF	OFF	ON	Constant : 3	90.01%
10	ON	OFF	OFF	ON	Constant : 3	91.05%
11	OFF	ON	OFF	ON	Constant : 3	90.01%
12	ON	ON	OFF	ON	Constant : 3	91.05%
13	OFF	OFF	ON	ON	Constant : 3	89.54%
14	ON	OFF	ON	ON	Constant : 3	90.86%
15	OFF	ON	ON	ON	Constant : 3	89.83%
16	ON	ON	ON	ON	Constant : 3	90.58%
17	OFF	OFF	OFF	OFF	Dynamic	93.97%
18	ON	OFF	OFF	OFF	Dynamic	94.63%
19	OFF	ON	OFF	OFF	Dynamic	93.97%
20	ON	ON	OFF	OFF	Dynamic	94.63%
21	OFF	OFF	ON	OFF	Dynamic	93.78%
22	ON	OFF	ON	OFF	Dynamic	94.63%
23	OFF	ON	ON	OFF	Dynamic	94.06%
24	ON	ON	ON	OFF	Dynamic	96.61%
25	OFF	OFF	OFF	ON	Dynamic	94.35%
26	ON	OFF	OFF	ON	Dynamic	94.82%
27	OFF	ON	OFF	ON	Dynamic	94.35%
28	ON	ON	OFF	ON	Dynamic	94.82%
29	OFF	OFF	ON	ON	Dynamic	93.97%
30	ON	OFF	ON	ON	Dynamic	94.72%
31	OFF	ON	ON	ON	Dynamic	94.35%
32	ON	ON	ON	ON	Dynamic	**96.70%**

APPENDIX B: FINDINGS FROM TRAINING PART

Table B.1. DTW distance threshold values of each template gesture type on each axis.

Gesture Class \ Threshold	Min DTW Distance on x axis	Max DTW Distance on x axis	Min DTW Distance on y axis	Max DTW Distance on y axis	Min DTW Distance on z axis	Max DTW Distance on z axis
1	0.366963	4.96132	0.660745	15.759	0.488112	18.1234
2	0.628353	10.4934	0.140728	5.46075	0.36975	8.84511
3	0.523203	6.61127	0.249236	4.19809	0.306786	6.76258
4	0.2111	2.49621	0.543717	8.4402	0.253275	9.05376
5	0.55948	7.30249	0.44584	6.20961	0.357632	10.5646
6	0.704822	7.30562	0.621314	9.09508	0.567924	13.1071
7	1.00674	7.48257	0.892143	8.36223	0.580739	9.21046
8	1.02519	7.33565	1.26738	10.6344	0.627831	8.45951
9	1.1798	12.0865	0.652959	13.6204	0.70625	17.3523
10	0.977618	9.8662	0.880651	9.2703	0.542985	9.8274
11	1.02641	6.79466	0.870725	6.2732	0.610638	6.76474
12	1.30977	11.5253	0.614656	8.7932	0.552789	11.5179
13	1.29786	10.12	0.943864	8.58284	0.584787	9.82795
14	1.09326	7.52301	0.781868	7.12837	0.684935	9.57316
15	0.982116	9.43468	0.554388	9.21406	0.513074	8.40113
16	1.08159	8.51399	0.788986	6.0276	0.726587	7.75498
17	1.08345	6.7967	1.3236	11.5169	0.975159	11.9138
18	0.520953	7.15434	0.577131	8.44041	0.86949	11.5803
19	0.787927	8.74777	0.650224	5.02938	0.968705	9.47047
20	0.766398	6.36951	0.595201	5.87538	1.24038	7.88993

Table B.2. Warping window sizes of all gesture types for x axis (m is gesture type, n is time).

m\n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
2	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	6	6	4	4	29	29	29	29	29	29	29	29	29
3	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	4	4	3	3	29	29	29
4	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
5	29	29	29	29	29	29	29	29	29	29	29	29	6	6	6	29	29	29	29	29	29	5	5	29	29	29	29	29	29	29
6	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	5	5	29	29	29	29	29	29	5	5	5
7	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	4	4	29	29	4	4	29	29	5	5	29	29	29
8	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	7	7	29	29	29
9	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	7	7	29	29	29	29	29	29	29	29	29
10	29	29	29	29	29	29	29	29	29	29	29	29	4	4	4	29	29	29	29	4	4	29	29	29	29	29	29	7	7	7
11	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	5	5	5	5	29	29	29	29	29	29	29	29	29
12	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
13	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29	29	29	29	29	5	5	29	29	29
14	29	29	29	29	29	29	29	29	29	29	29	29	5	5	5	4	4	29	29	4	4	29	29	29	29	29	29	29	29	29
15	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
16	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
17	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	4	4	29	29	29	29	29	29	29	29	29	29	29
18	29	29	29	29	29	29	29	29	29	29	29	29	4	4	4	1	1	29	29	29	29	29	29	29	29	29	29	29	29	29
19	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29
20	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29

Table B.3. Warping window sizes of all gesture types for y axis (m is gesture type, n is time).

m\n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
2	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	8	8	3	3	29	29	29
3	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
4	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
5	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
6	29	29	29	29	29	29	29	29	3	3	29	29	3	3	3	29	29	5	5	3	3	3	3	3	3	3	3	29	29	29
7	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	4	4	29	29	29	29	29	29	29	29	3	3	29	29	29
8	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	5	5	29	29	29	29	29	29	29	29	29	29	29	29	29
9	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	3	3	29	29	29	29	29	29	29	29	29	29	29
10	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	29	29	3	3	29	29	3	3	29	29	29	29	29
11	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29	29	29
12	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
13	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	3	3	29	29	3	3	3
14	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	8	8	29	29	29	29	29	29	29	29	29	29	29
15	29	29	29	29	29	29	29	29	29	29	29	29	4	4	4	29	29	4	4	3	3	29	29	29	29	29	29	3	3	3
16	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	6	6	29	29	29	29	29
17	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29	29	29	29	29	4	4	29	29	3	3	3
18	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	3	3	29	29	29	29	29	29	29	29	29	29	29	29	29
19	29	29	29	29	1	1	29	29	29	29	1	1	0	0	0	29	29	29	29	0	0	29	29	29	29	29	29	2	2	2
20	29	29	29	29	29	29	29	29	29	29	29	29	3	3	3	29	29	29	29	29	29	29	29	4	4	29	29	29	29	29

Table B.4. Warping window sizes of all gesture types for z axis (m is gesture type, n is time).

m\n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
2	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	3	3	29	29	29	29	29	29	29
3	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
4	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
5	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	10	10	3	3	29	29	29
6	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
7	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	8	8	29	29	29	29	29	29	29	29	29	29	29
8	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	5	5	29	29	29	29	29	29	29	29	29
9	29	29	29	29	29	29	29	29	29	29	29	29	3	3	3	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29
10	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	3	3	29	29	29	29	29	29	29	29	29	29	29
11	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29
12	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	6	6	5	5	29	29	29	29	29
13	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	4	4	29	29	29	29	3	3	29	29	29	29	29	29	29
14	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	7	7	29	29	4	4	4
15	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	5	5	29	29	29	29	4	4	29	29	29	29	4	4	4
16	29	29	29	29	29	29	29	29	29	29	29	29	3	3	3	29	29	29	29	3	3	29	29	29	29	4	4	29	29	29
17	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	3	3	29	29	3	3	29	29	29	29	29	29	29
18	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
19	29	29	29	29	29	29	29	29	29	29	29	29	3	3	3	29	29	29	29	4	4	29	29	29	29	3	3	29	29	29
20	29	29	29	29	29	29	29	29	3	3	29	29	29	29	29	29	29	29	29	5	5	29	29	3	3	29	29	29	29	29

APPENDIX C: ALGORITHMS

Input: Two character array with different lengths and a warping window array;

$\{q(i)\}_{i=1}^{M}$: as template sequence with length of M

$\{c(i)\}_{i=1}^{N}$: as candidate sequence with length of N

$\{w(i)\}_{i=1}^{\max(M,N)}$: as warping window array sequence with length of maximum value of M and N

Output: The value of DTW distance between template and candidate.

```
begin
    DTW(0,0) ← 0
    DTW(0,1...N) ← ∞
    DTW(1...M,0) ← ∞
    for i ← 1 to M do
        for j ← 1 to N do
            cost ← |q(i) − c(j)|
            if |i − j| ≤ w(max(i,j)) then
                DTW(i,j) ← cost + min(DTW(i−1,j), DTW(i,j−1), DTW(i−1,j−1))
            end
            else
                DTW(i,j) ← ∞
            end
        end
    end
    return DTW(M,N)
end
```

Figure C.1. Simple DTW Distance with Dynamic Warping Window Size algorithm.

```
Input: T2 : Validation Data Set
Data: I: Maximum iteration number to learn best warping window sizes, M: Number of total
      gesture types, D: total number of dimensions (x, y and z), N: Number of elements in a
      sequence
Output: W = {w⃗(n)}_{n=1}^{20} : Warping window size matrix
begin
    W ← 1
    for iteration ← 1 to I=3 do
        for m ← 1 to M=20 do
            for d ← 1 to D=3 do
                DistanceMetric ← ∞
                TPNumber ← 0
                Save W
                EvaluateDTW(T2, DistanceMetric, TPNumber)
                FindBestWarpingSize(T2, m, d, 0, N, 1, DistanceMetric, TPNumber)
                Improvement ← EvaluateDTW(T2, DistanceMetric, TPNumber)
                if Improvement then
                    continue
                end
                else
                    Undo W from saved
                end
            end
        end
    end
    return W
end
```

Figure C.2. Best Warping Window Size algorithm.

```
Input: T2 : Validation Data Set, DistanceMetric: Current Distance metric value,
       TPNumber: Current true positive number coming from validation data set
       classification
Data: M: Number of total gesture types
Result: DistanceMetric, TPNumber are overridden and saved if there is an improvement
Output: Improvement: Improvement result (yes or no) according to last distance metric and
        true classification number.
begin
    Improvement ← False, TPD ← 0, FND ← 0, TP ← 0, FN ← 0
    foreach G ∈ T2 do
        BestDistance ← ∞, PredictedClass ← −1
        for m ← 1 to M do
            if LBKeoghDistance(G,m) < BestDistance ∧ DTWDistance(G,m)
            < BestDistance then
                BestDistance ← DTWDistance(G,m)
                PredictedClass ← m
            end
        end
        if PredictedClass = class of G then
            TPD ← TPD + DTWDistance(G,m)
            TP ← TP + 1
        end
        else
            FND ← FND + DTWDistance(G,m)
            FN ← FN + 1
        end
    end
    TempDistanceMetric ← (TPD/FND) × (FN/TP)
    if TP > TPNumber ∧ (TP ≤ TPNumber ∨ TempDistanceMetric < DistanceMetric)
    then
        Improvement ← True
        TPNumber ← TP
        DistanceMetric ← TempDistanceMetric
    end
    return Improvement
end
```

Figure C.3. DTW evaluation algorithm.

Input: **T2**: Validation Data Set, m: Gesture type index (model number), d: dimension , $startTime$: warping band start, $endTime$: warping band end, WS: current warping size, $DistanceMetric$: Current Distance metric value, $TPNumber$: Current true positive number coming from validation data set classification

Data: **W**: Warping windows size matrix (see: Equation 3.27), $V_{threshold}$: Threshold variable used for eliminating overfitting (see: Equation 3.29)

Result: Best warping window size is saved to be used in classification

```
begin
    if (WS < S) ∧ (endTime − startTime) ≥ 1 then
        Save W_d^m
        W_d^m(startTime, ..., endTime) ← WS
        Improvement ← EvaluateDTW(T2, DistanceMetric, TPNumber)
        if Improvement then
            FindBestWarpingSize(T2, m, d, startTime, endTime, WS + 1, DistanceMetric,
            TPNumber)
        end
        else
            Load saved W_d^m
            if (endTime − startTime) > V_threshold + 1 then
                Middle ← (starttime + endtime)/2
                FindBestWarpingSize(T2, m, d, startTime, Middle, WS, DistanceMetric,
                TPNumber)
                FindBestWarpingSize(T2, m, d, Middle + 1, endTime, WS,
                DistanceMetric, TPNumber)
            end
        end
    end
end
```

Figure C.4. Finding Best Warping Window Sizes algorithm.

Input: **G**: Gesture data, m: Gesture type index (model number)

Data: $\overrightarrow{U}(n)$: upper bound sequences (see: Equation 3.25), $\overrightarrow{L}(n)$: lower bound sequences (see: Equation 3.25), N: Number of elements in a sequence

Output: *Distance* : LBKeogh Distance of given gesture data from corresponding template

```
begin
    Distance ← 0
    for d ← 1 to 3 do
        for n ← 1 to N=30 do
            if G_d(n) > U_d^m(n) then
                Distance ← Distance + (G(n)_d − U_d^m(n))^2
            end
            else if G_d(n) < L_d^m(n) then
                Distance ← Distance + (G_d(n) − L_d^m(n))^2
            end
            else ,data point is in range
                continue
            end
        end
    end
    return Distance
end
```

Figure C.5. LBKeogh Distance calculation algorithm.

Input: $\mathbf{G}$: Gesture data, m: Gesture type index (model number)

Data: $\mathbf{Q}$: Template library, $\mathbf{W}$: Warping windows size matrix (see: Equation 3.27), N: Number of elements in a sequence

Output: $Distance$: DTW Distance of given gesture data from corresponding template

```
1  begin
2      Distance ← 0
3      for d ← 1 to 3 do
4          Cost ← 0
5          DistanceMatrix(0,0) ← 0
6          DistanceMatrix(0,1...N) ← ∞
7          DistanceMatrix(1...N,0) ← ∞
8          for i ← 1 to N do
9              for j ← 1 to N do
10                 Cost ← |G_d(i) − Q_d^m(j)|
11                 if |i − j| ≤ W_d^m(max(i,j)) then
12                     DistanceMatrix(i,j) ← Cost + min( DistanceMatrix(i−1,j),
                                                         DistanceMatrix(i,j−1),
                                                         DistanceMatrix(i−1,j−1) )
13                 end
14                 else ,data point is out of warping window range
15                     DistanceMatrix(i,j) ← ∞
16                 end
17             end
18         end
19         Distance ← Distance + DistanceMatrix(N,N)
20     end
21     return Distance
22 end
```

Figure C.6. DTW Distance calculation algorithm.

APPENDIX D: CASE STUDY

Let's assume; there are 2 gesture class and 1 participant and 4 gesture data for training and 5 gesture data for validation of each gesture classes were collected with the same conditions defined in Experimental Information section (see: Section 3.11.2). The first gesture type called 'Class A' and the second gesture type called 'Class B' are shown in Figure D.1, which were chosen among gesture set used in this theses (see: Figure 3.2). All main procedures and results are explained below.

- Template Generation. In order to generate templates, warping window sizes and threshold values; the procedures described in Template Generation section are followed by using the training data set of case study. The raw, filtered, preprocessed, sampled training data and generated template sequences with upper and lower bounds are given in Figures from D.2 to D.6, respectively.

- Recognition Processes. After DTW training step, now the system is ready to classify each gesture data of validation data set by following the general recognition algorithm explained in Recognition Processes section (see: Section 3.2.2). Raw, validated and sampled validation data are given in Figure D.7. Template matching result and generated confusion matrix are shown in Table D.1.

- Result of case study. After getting the results from template matching with threshold control, a confusion matrix is generated in order to see all classification result. Here, in the validation part; 2 data sequences are eliminated in the data validation, so there are 2 sequence of non-classified. However, the classification accuracy is 100% for 2 class with 8 validation data.

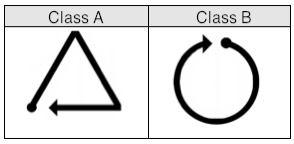
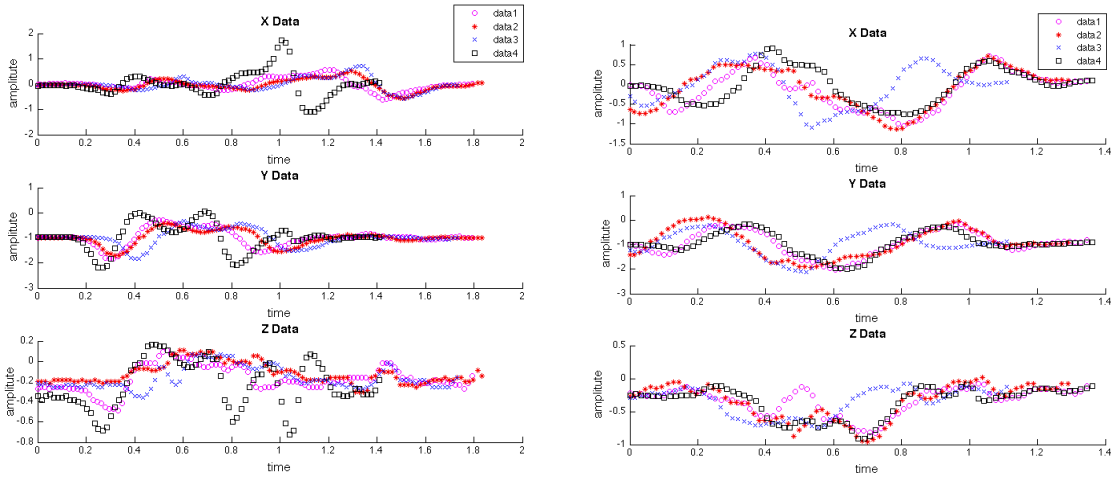
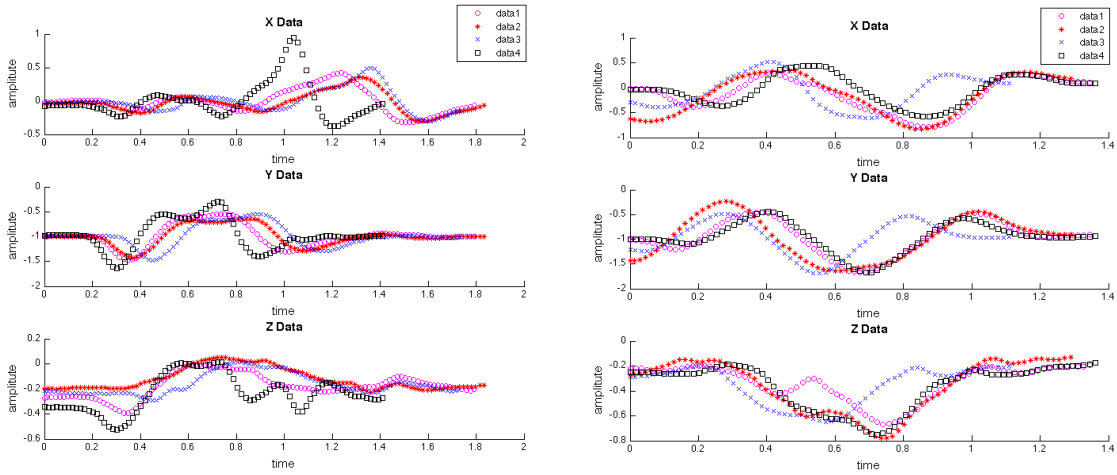


Figure D.1. Gesture types used in case study.



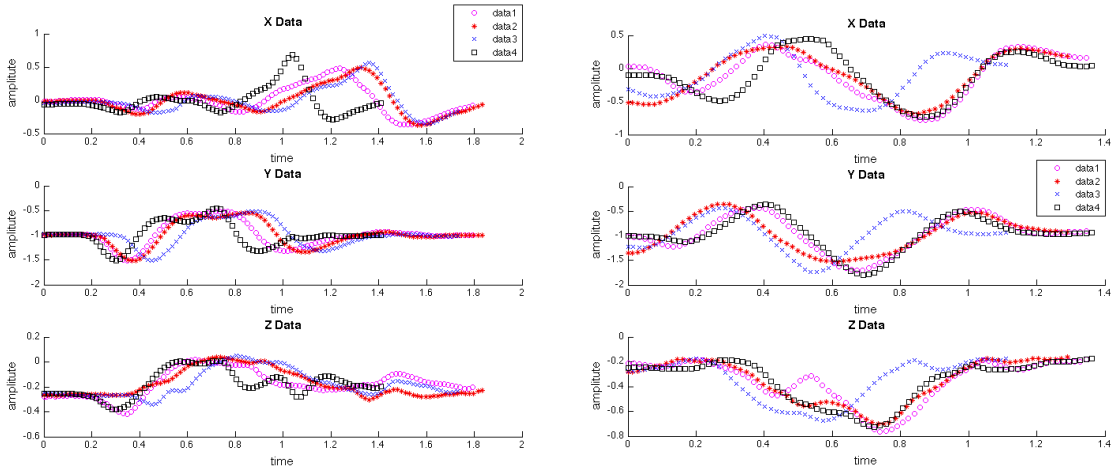
(a) (b)

Figure D.2. Raw gesture data sequences. (a) Raw Class A. (b) Raw Class B.



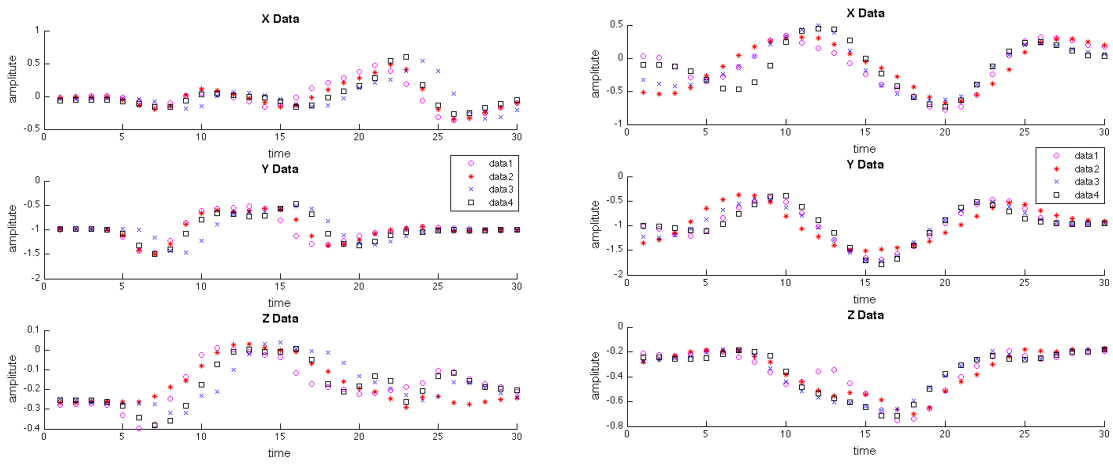
(a) (b)

Figure D.3. Filtered gesture data sequences. (a) Filtered Class A. (b) Filtered Class B.



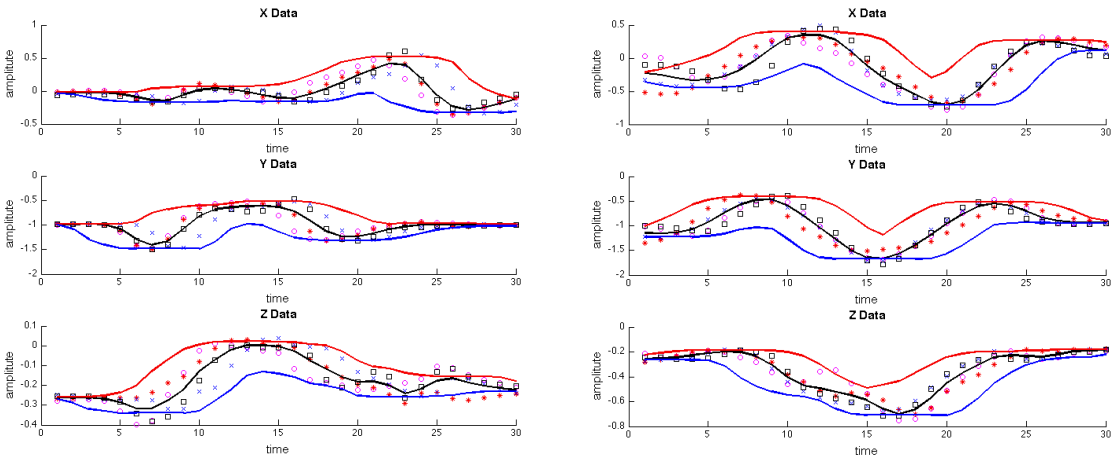
(a) (b)

Figure D.4. Preprocessed gesture data sequences. (a) Preprocessed Class A. (b) Preprocessed Class B.



(a) (b)

Figure D.5. Sampled gesture data sequences. (a) Sampled Class A. (b) Sampled Class B.



(a) (b)

Figure D.6. Template data sequences. (a) Template sequences of Class A with upper and lower bounds. (b) Template sequences of Class B with upper and lower bounds.

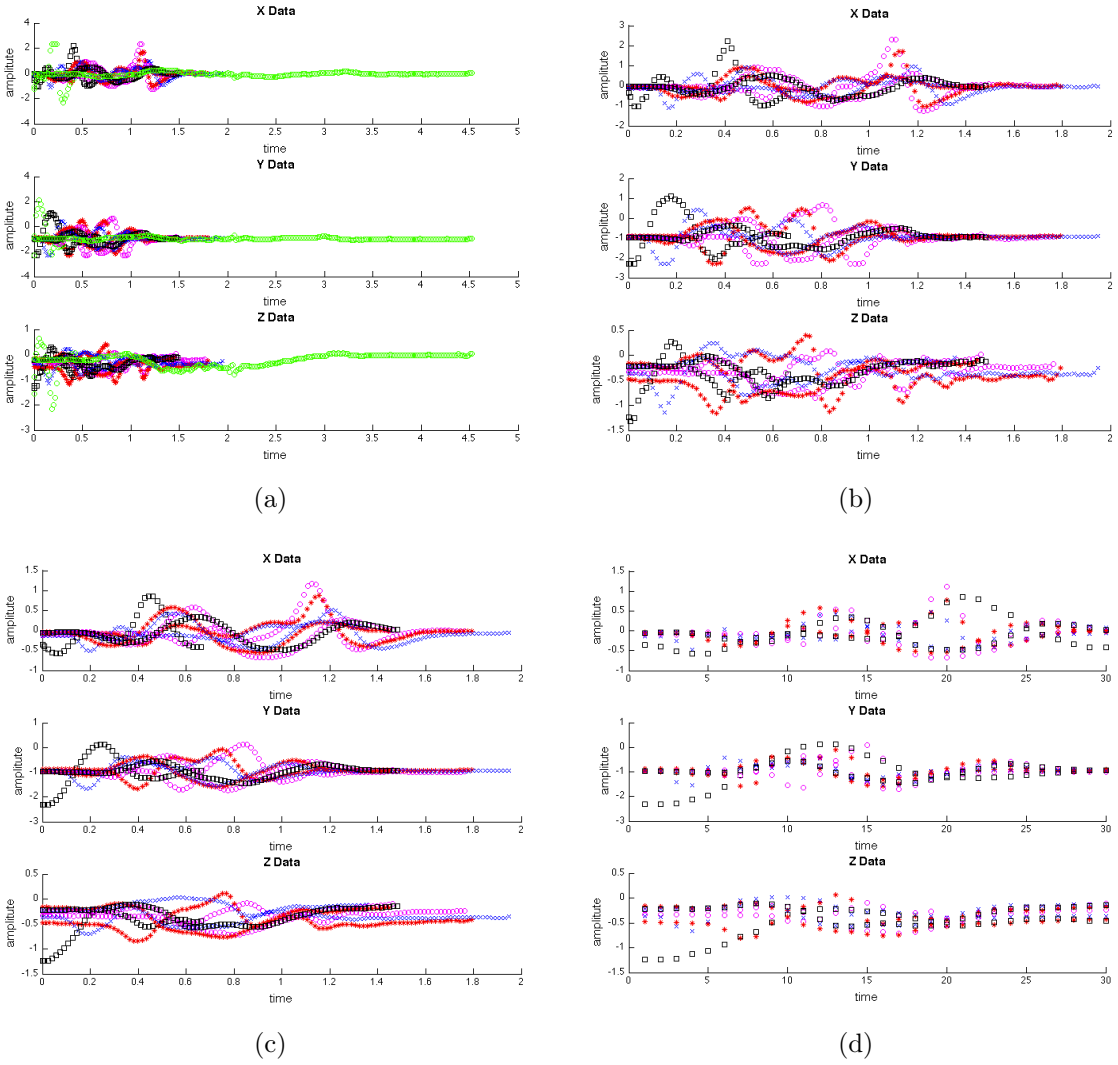


Figure D.7. Raw, Filtered, Sampled validation data for Class A and Class B. (a) Raw validation data. (b) Validation data after Data Validation. (c) Filtered validation data. (d) Down-sampled validation data.

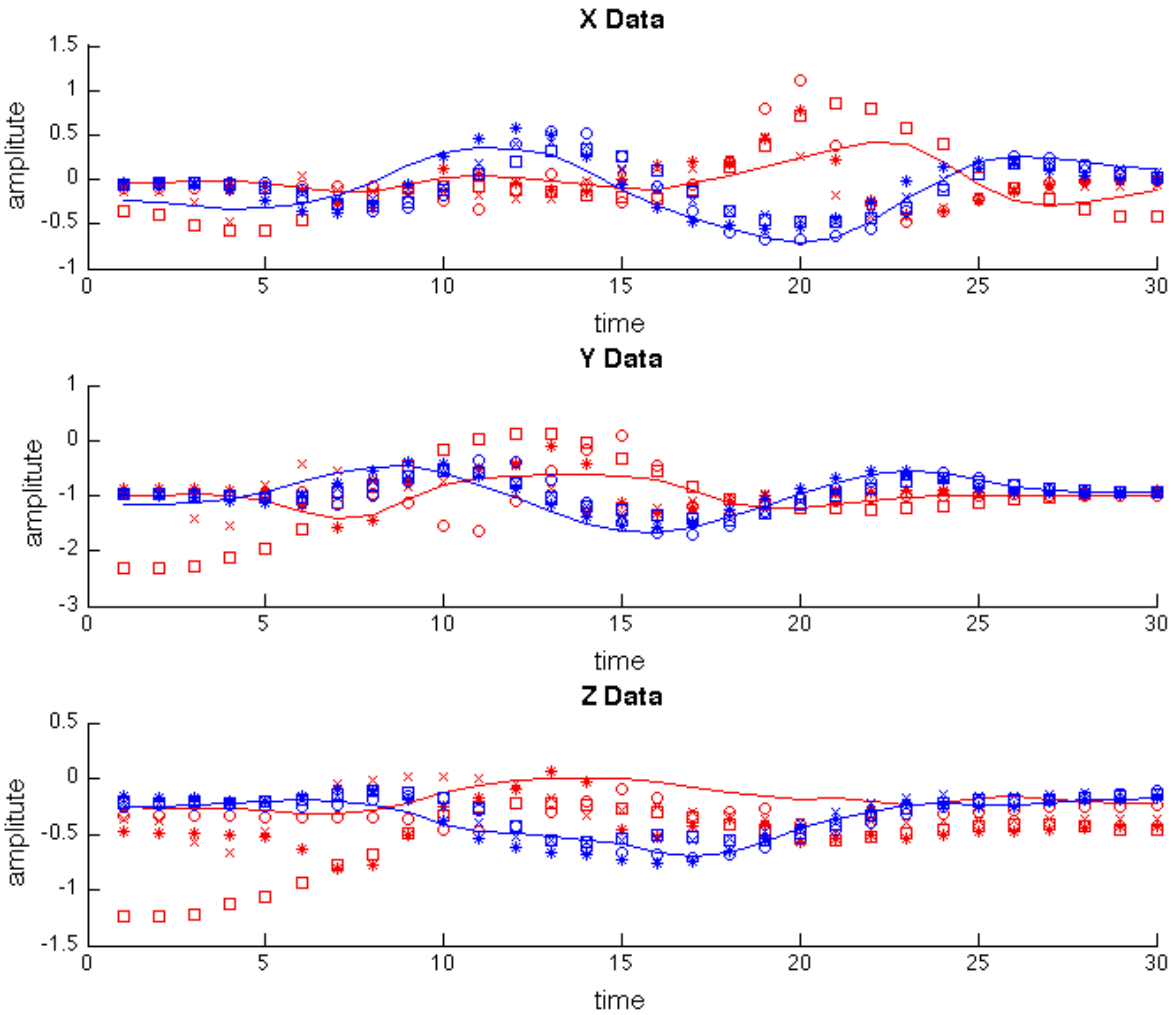


Figure D.8. Template Matched data.

Table D.1. Template Matching result with confusion matrix.

		Predicted Class Numbers							
		A	B	sum	TP	FP	FN	Pr %	Rc %
Actual Class Numbers	A	4	0	4	4	0	0	100	100
	B	0	4	4	4	0	0	100	100
	sum	4	4	8	8	0	0	100	100

REFERENCES

1. Wu, J., G. Pan, D. Zhang, G. Qi, and S. Li, "Gesture Recognition with a 3-D Accelerometer", *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing*, UIC '09, pp. 25–38, Springer-Verlag, Berlin, Heidelberg, 2009.

2. Klingmann, M., *Accelerometer-Based Gesture Recognition with the iPhone*, Ph.D. thesis, Goldsmiths University of London, 2009.

3. Mantyla, V.-M., J. Mantyjarvi, T. Seppanen, and E. Tuulari, "Hand Gesture Recognition of a Mobile Device User", *IEEE International Conference on Multimedia and Expo, 2000. ICME 2000.*, Vol. 1, pp. 281 –284 vol.1, 2000.

4. Mäntyjärvi, J., J. Kela, P. Korpipää, and S. Kallio, "Enabling Fast and Effortless Customization in Accelerometer Based Gesture Interaction", *Proceedings of the 3rd International Conference on Mobile and Ubiquitous Multimedia*, MUM '04, pp. 25–31, ACM, New York, NY, USA, 2004.

5. Pylvänäinen, T., "Accelerometer Based Gesture Recognition Using Continuous HMMs", Marques, J., N. Pérez de la Blanca, and P. Pina (editors), *Pattern Recognition and Image Analysis*, Vol. 3522 of *Lecture Notes in Computer Science*, pp. 413–430, Springer, Berlin, Heidelberg, 2005.

6. Deutsch, J. E., M. Borbely, J. Filler, K. Huhn, and P. Guarrera-Bowlby, "Use of a Low-Cost, Commercially Available Gaming Console (Wii) for Rehabilitation of an Adolescent With Cerebral Palsy", *Physical Therapy*, Vol. 88, No. 10, pp. 1196–1207, October 2008.

7. Kratz, L., M. Smith, and F. J. Lee, "Wiizards: 3D Gesture Recognition for Game Play Input", *Proceedings of the 2007 Conference on Future Play*, Future Play '07, pp. 209–212, ACM, New York, NY, USA, 2007.

8. Schlömer, T., B. Poppinga, N. Henze, and S. Boll, "Gesture Recognition with a Wii Controller", *Proceedings of the 2nd International Conference on Tangible and Embedded Interaction*, TEI '08, pp. 11–14, ACM, New York, NY, USA, 2008.

9. Ratanamahatana, C. A. and E. J. Keogh, "Making Time-Series Classification More Accurate Using Learned Constraints", Berry, M. W., U. Dayal, C. Kamath, and D. B. Skillicorn (editors), *Proceedings of the Fourth SIAM International Conference on Data Mining*, SIAM, Lake Buena Vista, Florida, USA, April 2004.

10. Sakoe, H. and S. Chiba, "Dynamic Programming Algorithm Optimization for Spoken Word Recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 26, No. 1, pp. 43–49, February 1978.

11. Keogh, E., "Exact Indexing of Dynamic Time Warping", *Proceedings of the 28th International Conference on Very Large Databases*, VLDB '02, pp. 406–417, VLDB Endowment, Hong Kong, China, 2002.

12. Halliday, D., R. Resnick, and J. Walker, *Fundamentals of Physics*, Chapter 2: Motion Along a Straight Line, pp. 17–26, Wiley, 6 edition edition, 2000.

13. Apple Inc., *iPhone User Guide for iPhone OS 3.1*, Apple Inc., Cupertino, CA, USA, 2009.

14. STMicroelectronics Inc., *LIS302DL, MEMS Motion Sensor 3-Axis - ± 2g/± 8g Smart Digital Output "Piccolo" Accelerometer*, 2008.

15. Sensr Inc., *Practical Guide to Accelerometers*, Sensr Inc., Saline, MI, USA, 2011.

16. Agrawal, S., I. Constandache, S. Gaonkar, and R. R. Choudhury, "Phonepoint Pen: Using Mobile Phones to Write in Air", *Proceedings of the 1St Acm Workshop on Networking, Systems, and Applications for Mobile Handhelds*, MobiHeld '09, pp. 1–6, ACM, New York, NY, USA, 2009.

17. Centers for Disease Control and Prevention, "Cerebral Palsy", 2011, `http://www.`

`cdc.gov/ncbddd/dd/ddcp.htm`, June 14, 2011.

18. Kela, J., P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and S. Marca, "Accelerometer-Based Gesture Control for a Design Environment", *Personal and Ubiquitous Computing*, Vol. 10, pp. 285–299, 2006.

19. Mitra, S. and T. Acharya, "Gesture Recognition: A Survey", *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 37, No. 3, pp. 311–324, May 2007.

20. Teitelman, W., "Real Time Recognition of Hand-Drawn Characters", *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, AFIPS '64 (Fall, part I), pp. 559–575, ACM, New York, NY, USA, 1964.

21. Davis, M. R. and T. O. Ellis, "The RAND Tablet: a Man-Machine Graphical Communication Device", *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, AFIPS '64 (Fall, part I), pp. 325–331, ACM, New York, NY, USA, 1964.

22. Christensen, C., "On the Implementation of Ambit, a Language for Symbol Manipulation", *Communications of the ACM*, Vol. 9, pp. 570–573, August 1966.

23. Rovner, P. D. and D. A. Henderson, "On the Implementation of Ambit/G: a Graphical Programming Language", *Proceedings of the 1St International Joint Conference on Artificial Intelligence*, pp. 9–20, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1969.

24. Coleman, M. L., "Text Editing on a Graphic Display Device Using Hand-Drawn Proofreader's Symbols", Faiman, M. and J. Nievergelt (editors), *Pertinent Concepts in Computer Graphics, Proceedings of the 2nd University of Illinois Conference on Computer Graphics*, pp. 282–290, University of Illinois Press, Champaign, Ill., 1969.

25. Myers, B. A., "A Brief History of Human-Computer Interaction Technology", *interactions*, Vol. 5, No. 2, pp. 44–54, March 1998.

26. Welch, G. and G. Bishop, "An Introduction to the Kalman Filter", March 2007, `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`, February 4, 2011.

27. Jolliffe, I. T., *Principal Component Analysis*, Springer-Verlag, New York, NY, USA, second edition, October 2002.

28. Brigham, E. O. and R. E. Morrow, "The Fast Fourier transform", *IEEE Spectrum*, Vol. 4, No. 12, pp. 63–70, December 1967.

29. Rabiner, L., "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 77, No. 2, pp. 257–286, February 1989.

30. Schölkopf, B., C. J. C. Burges, and A. J. Smola (editors), *Advances in Kernel Methods: Support Vector Learning*, MIT Press, Cambridge, MA, USA, 1999.

31. Kwok, C., D. Fox, and M. Meila, "Real-Time Particle Filters", *Proceedings of the IEEE*, Vol. 92, No. 3, pp. 469–484, March 2004.

32. Isard, M. and A. Blake, "Contour Tracking by Stochastic Propagation of Conditional Density", *Proceedings of the 4th European Conference on Computer Vision*, Vol. 1 of *ECCV '96*, pp. 343–356, Springer-Verlag, London, UK, 1996.

33. Nintendo Inc., "Wii Technical Details", 2011, `http://www.nintendo.co.uk/NOE/en_GB/systems/technical_details_1072.html`, May 18, 2011.

34. Analog Devices, "ADXL330", March 2007, `http://www.analog.com/en/mems-sensors/inertial-sensors/adxl330/products/product.html`, January 22, 2011.

35. Sony Computer Entertainment Inc., "PlayStation Move Motion

Controller", 2011, http://us.playstation.com/ps3/accessories/playstation-move-motion-controller-ps3.html, June 4, 2011.

36. Smith, S. W., *The Scientist and Engineer's Guide to Digital Signal Processing*, Chapter 28: Digital Signal Processors, California Technical Publishing, 1997.

37. Wikipedia, "Digital Filter", 2011, http://en.wikipedia.org/wiki/Digital_filter, May 3, 2011.

38. Wikipedia, "Low-Pass Filter", 2011, http://en.wikipedia.org/wiki/Low-pass_filter, March 14, 2011.

39. Myers, C. and L. Rabiner, "A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition", *The Bell System Technical Journal*, Vol. 60, No. 7, pp. 1389–1409, September 1981.

40. Itakura, F., "Minimum Prediction Residual Principle Applied to Speech Recognition", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 23, No. 1, pp. 67–72, February 1975.

41. Arlot, S. and A. Celisse, "A Survey of Cross-Validation Procedures for Model Selection", *Statistics Surveys*, Vol. 4, pp. 40–79, 2010.

42. Refaeilzadeh, P., L. Tang, and H. Liu, *Encyclopedia of Database Systems*, Chapter Cross-Validation, pp. 532–538, Springer US, 2009.

43. Kohavi, R. and F. Provost, "Glossary of Terms", 1998, http://robotics.stanford.edu/~ronnyk/glossary.html, January 13, 2011.

44. Yang, J. and Y. Xu, "Hidden Markov Model for Gesture Recognition", Technical Report CMU-RI-TR-94-10, Robotics Institute, Cargnegie Mellon University, Pittshburgh, PA, May 1994.

45. Tanguay Jr., D., *Hidden Markov Models for Gesture Recognition*, Master's thesis,

Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, Cambridge, MA, USA, August 1995.

46. Starner, T. E., *Visual Recognition of American Sign Language Using Hidden Markov Models*, Master's thesis, Massachusetts Institute of Technology, Program in Media Arts and Sciences, Cambridge, MA, USA, February 1995.

47. Keskin, C., A. Erkan, and L. Akarun, "Real Time Hand Tracking and 3D Gesture Recognition for Interactive Interfaces Using Hmm", *Proceedings of the Joint International Conference*, ICANN/ICONIP, Springer, 2003.

48. Westeyn, T., H. Brashear, A. Atrash, and T. Starner, "Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition", *Proceedings of the 5th International Conference on Multimodal Interfaces - ICMI '03*, p. 85, ACM Press, New York, New York, USA, 2003.

49. Brashear, H., T. Starner, P. Lukowicz, and H. Junker, "Using Multiple Sensors for Mobile Sign Language Recognition", *Proceedings of Seventh IEEE International Symposium on Wearable Computers*, pp. 45–52, October 2003.

50. Mizell, D., "Using Gravity to Estimate Accelerometer Orientation", *Seventh IEEE International Symposium on Wearable Computers*, pp. 252–253, October 2003.

51. Niels, R., *Dynamic Time Warping An Intuitive Way of Handwriting Recognition ?*, Master's thesis, Radboud University, Nijmegen, Netherlands, December 2004.

52. Prekopcsák, Z., "Accelerometer Based Real-Time Gesture Recognition", *Proceedings of the 12th International Student Conference on Electrical Engineering*, pp. 1–5, Prague, Czech Republic, May 2008.

53. Liu, J., Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-Based Personalized Gesture Recognition and its Applications", *IEEE International Conference on Pervasive Computing and Communications,*

2009. PerCom 2009., pp. 1–9, 2009.

54. Leong, T., J. Lai, J. Panza, P. Pong, and J. Hong, "Wii Want to Write: An Accelerometer Based Gesture Recognition System", *International Conference on Recent and Emerging Advanced Technologies in Engineering*, pp. 4–7, Kuala Lumpur, Malaysia, 2009.

55. Akl, A. and S. Valaee, "Accelerometer-Based Gesture Recognition via Dynamic-Time Warping, Affinity Propagation, and Compressive Sensing", *IEEE International Conference on Speech and Signal Processing*, pp. 2270–2273, Dallas, Texas, USA, March 2010.

56. Smith, S. W., *The Scientist and Engineer's Guide to Digital Signal Processing*, Chapter 14: Introduction to Digital Filters, California Technical Publishing, 1997.

57. Park, H.-s., J.-s. Lee, and C.-h. Jun, "A K-Means-Like Algorithm for K-Medoids Clustering and Its Performance", *Proceedings of the 36th CIE Conference on Computers and Industrial Engineering*, pp. 20–23, Taipei, Taiwan, June 2006.

58. Tan, P., M. Steinbach, and V. Kumar, *Introduction to Data Mining*, Chapter 8. Cluster Analysis: Basic Concepts and Algorithms, Addison-Wensley, 2006.

59. Elmezain, M., A. Al-Hamadi, J. Appenrodt, and B. Michaelis, "A Hidden Markov Model-Based Continuous Gesture Recognition System for Hand Motion Trajectory", *19th International Conference on Pattern Recognition*, pp. 1–4, December 2008.

60. Alexandrov, V. and M. Gerstein, "Using 3D Hidden Markov Models that Explicitly Represent Spatial Coordinates to Model and Compare Protein Structures.", *BMC Bioinformatics*, Vol. 5, p. 2, January 2004.

61. Alpaydın, E., *Introduction to Machine Learning*, Chapter 7. Clustering, MIT Press, Cambridge, MA, USA, 2004.

62. Black, A. W. and K. A. Lenzo, "Flite: a Small Fast Run-Time Synthesis Engine", *4th International Speech Communication Association Tutorial and Research Workshop on Speech Synthesis*, Perthshire, UK, 2001.

63. Foster, S., "iPhone Text-To-Speech", 2010, `https://bitbucket.org/sfoster/iphone-tts/`, June 1, 2011.

64. Weiss, P. H., K. Zilles, and G. R. Fink, "When Visual Perception Causes Feeling: Enhanced Cross-Modal Processing in Grapheme-Color Synesthesia", *NeuroImage*, Vol. 28, No. 4, pp. 859 – 868, 2005.

65. Von Ahn, L. and L. Dabbish, "Designing Games with a Purpose", *Communications of the ACM*, Vol. 51, pp. 58–67, August 2008.