APPLYING TEELINE SHORTHAND USING LEAP MOTION CONTROLLER

by

Weikai Zang

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science (MSc) in
Mathematics and Computational Science

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

Abstract

A hand gesture recognition program was developed to recognize users' Teeline shorthand gestures as English letters, words and sentences using Leap Motion Controller. The program is intended to provide a novel way for the users to interact with electronics by waving gestures in the air to input texts instead of using keyboards. In the recognition mode, the dynamic time warping algorithm is used to compare the similarities between different templates and gesture inputs and summarize the recognition results; in the edit process, users are able to build their own gestures to customize the commands. A series of experiment results show that the program can achieve a considerable recognition accuracy, and it has consistent performance in face of different user groups.

Keywords

Human-Computer Interaction, Teeline Shorthand, Leap Motion Controller, Dynamic Time Warping, Hand Gesture Recognition

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Prof. Ratvinder Singh Grewal for the continuous support of my master study and research. His guidance helped me during the time of research and writing of this thesis.

Besides my advisor, I would like to thank my committee members, Prof. Julia Johnson and Prof. Kalpdrum Passi for their insightful comments and hard questions; and all Computer-Human Interaction Lab (CHIL) members for their help and kindly suggestions.

Last but not the least, I owe sincere and earnest thankfulness to my family: my parents, for giving birth to me at the first place and supporting me spiritually throughout my life.

Table of Contents

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables viii

List of Figures ix

List of Appendices xi

Chapter 1 1

Introduction 1

1.1 Background 1

1.2 Current situation 3

1.3 Thesis objective 4

1.4 Thesis outline 6

Chapter 2 7

Literature Review 7

2.1 Leap Motion Controller 7

2.1.1 Overview 7

2.1.2 Motion tracking data 9

2.1.3 System architecture 11

2.2 Shorthand 13

2.2.1 Introduction to shorthand 13

2.2.2 Classification 14

2.2.3 Teeline shorthand 15

2.3 Programming language and analysis software 17

Chapter 3 19

Design 19

3.1 Program algorithm-Dynamic Time Warping 19

3.1.1 Euclidean distance 19

3.1.2 Dynamic programming 20
3.2 Program implementation 23
3.2.1 Overall design 23
3.2.2 Recognition mode 25
3.2.3 Edit mode 26
3.2.4 Database 28
3.2.5 Implementation flow 29
3.2.6 Program interface 31
Chapter 4 34
Evaluations 34
4.1 Testing preparation 34
4.2 Questionnaire 35
4.3 Pilot testing 38
4.3.1 What is pilot testing? 38
4.3.2 Conducting the pilot testing 39
4.4 Experiment I 41
4.4.1 Objective 41
4.4.2 Hypotheses 41
4.4.3 Methodology 42
4.4.4 Results 48
4.5 Experiment II 52
4.5.1 Objective 52
4.5.2 Hypotheses 53
4.5.3 Methodology 53
4.5.4 Results 58
Chapter 5 65
Discussion 65
5.1 Discussion in Experiment I 65
5.2 Discussion in Experiment II 73
Chapter 6 82

Conclusion and Future Work 82
6.1 Conclusion 82
6.2 Future work 84
Bibliography 87
Appendix 91
Appendix A Program source code 91
Appendix B Approval letter of research application 120
Appendix C Questionnaire template in experiments 121
Appendix D Statistical analysis results 125
Appendix E Additional findings in program's performance for different user groups 134
Difference in recognition accuracies based on users' experience with video games 134
Difference in recognition accuracies based on users' experience with motion control devices 136

List of Tables

TABLE 4.1 DATA COLLECTED FROM QUESTIONNAIRES 45

TABLE 4.2 VARIATION IN RECOGNITION ACCURACY WITH INCREASING SAMPLE SIZE USING DATABASE 1 49

TABLE 4.3 VARIATION IN RECOGNITION ACCURACY WITH INCREASING SAMPLE SIZE USING DATABASE 2 50

TABLE 4.4 OVERALL RECOGNITION ACCURACY FOR EACH PARTICIPANT'S RECORD 59

TABLE 4.5 FREQUENCY ANALYSIS FOR OVERALL RECOGNITION ACCURACIES 61

TABLE 5.1 OPTIMAL SAMPLE SIZE FOR EACH CHARACTER USING DATABASE 1 69

TABLE 5.2 OPTIMAL SAMPLE SIZE FOR EACH CHARACTER USING DATABASE 2 69

TABLE 5.3 FREQUENCY OF OPTIMAL SAMPLE SIZE 71

List of Figures

FIGURE 2.1 LEAP MOTION COORDINATE SYSTEM 8

FIGURE 2.2 HAND TRACKING........ 9

FIGURE 2.3 FINGER TRACKING........ 10

FIGURE 2.4 TOOL TRACKING........ 10

FIGURE 2.5 A SWIPE GESTURE, A KEY TAP GESTURE, AND A SCREEN TAP GESTURE (FROM LEFT TO RIGHT) 11

FIGURE 2.6 THE SCALE MOTION, ROTATION MOTION, AND TRANSLATION MOTION (FROM LEFT TO RIGHT) 11

FIGURE 2.7 NATIVE APPLICATION INTERFACE 12

FIGURE 2.8 TEELINE SHORTHAND ALPHABET 16

FIGURE 2.9 REVISED TEELINE ALPHABET 17

FIGURE 3.1 EUCLIDEAN DISTANCE OF TWO TIME-DEPENDENT SEQUENCES X AND Y 20

FIGURE 3.2 TIME ALIGNMENT OF TWO TIME-DEPENDENT SEQUENCES X AND Y 21

FIGURE 3.3 PROGRAM OVERVIEW........ 24

FIGURE 3.4 RECOGNITION MODE........ 25

FIGURE 3.5 EDIT MODE........ 27

FIGURE 3.6 FLOWCHART OF THE PROGRAM 29

FIGURE 3.7 CONSOLE WINDOW 32

FIGURE 3.8 DISPLAY WINDOW (WHEN DRAWING A TEELINE CHARACTER “A”)........ 32

FIGURE 4.1 EVALUATION PROCESS IN EXPERIMENT I 47

Figure 4.2 Variation in recognition accuracy with increasing number of templates using Database 1 51
Figure 4.3 Variation in recognition accuracy with increasing number of templates using Database 2 52
Figure 4.4 Age distribution of participants 55
Figure 4.5 Gender of participants 56
Figure 4.6 Participants' Experience Playing Video Games 56
Figure 4.7 Participants' Experience Interacting with Motion Control Devices 57
Figure 4.8 Histogram of Recognition Accuracies using Database 1 62
Figure 4.9 Histogram of Recognition Accuracies using Database 2 63

List of Appendices

Appendix A Program Source Code 91

Appendix B Approval Letter of LU's REB 120

Appendix C Questionnaire Template in Experiments 121

Appendix D SPSS Statistical Analysis Results 125

Appendix E Additional Findings in Program's Performance for Different User Groups 134

Chapter 1

Introduction

1.1 Background

Human-Computer Interaction (HCI) refers to the process of information exchange between a person and a computer using certain dialogue, in a certain interactive way, to complete a certain task. HCI is more and more common in modern society; people communicate with one another by phone, work on the computer, and use advanced machines to improve production. The way people interact with computer is constantly developing over time.

The field of human-computer interaction has developed greatly, and has shifted from a time where people adapted to the computers to where, now, computers are adapting to human needs. The evolution has gone through several stages:

- Manual work;
- Use of job control language and interactive command language;
- Manipulating with graphical user interfaces;
- Human-computer interaction using multi-channel, multi-media intelligent stage.

The now ubiquitous direct manipulation interface is the direct manipulation of graphical objects; this is where objects visible onscreen are directly manipulated with a pointing device [1]. For example, a light-pen was used to manipulate objects, which included grabbing and moving objects, changing size, and using constraints with the support of a SketchPad [2]; following this, in 1965, the mouse was developed as a cheap replacement for light-pens and became famous as a practical input device in the 1970s [3]. The current international standard X Window System was developed in 1984, which allows drawing and moving windows on the display device with a mouse and keyboard.

In recent years, the emergence of motion control devices has made a big difference in the way people interact with computers. In 2006, the release of the Nintendo Wii had a massive effect on the gaming industry [4]. After the Nintendo Wii, the first-generation Kinect was introduced in 2010, which raised users' enthusiasm for motion control products. With these products, people can manipulate their virtual characters in game by changing their own body movements rather than remaining seated or holding a console or a mouse in one position. Motion-controlled games rapidly became popular in Europe and America, spreading to Asian countries [5]. In 2013, the launch of the Leap Motion controller (LM) once again expand the way of human-computer interaction. This small motion controller makes motion control practical not only in gaming, but also in many other fields.

The goal of this thesis is to design a program using the hand motion-sensing feature of the Leap Motion controller to detect people's writing gestures in the air and recognize them as plain texts in English.

1.2 Current situation

Since the release of the Leap Motion Controller (LM), more and more applications have been published on Airspace: the name of the applications store for LM. When Leap Motion Controller first launched, there were 75 apps in the store; today, in 2016, there are over 200 apps are shown in the store [6]. These applications are available on Windows, OS X or Web Link platforms, and cover education, games, music, and many other different categories. LM is becoming increasingly common in everyday life, and will be applied in various fields in the future.

The goal of gesture recognition is interpreting human gestures into computer language with mathematical algorithm for people and machines to interface more easily [7]. Hand gesture recognition is a challenging interdisciplinary research project, related to both computer science and language technology. Over the past few years, it has become commonplace technology in both entertainment and gaming markets.

Hidden Markov Model (HMM) and Dynamic Time Warping (DTW), two different algorithms, are widely applied in speech recognition systems. Since the hand gesture recognition is similar to the speech recognition with regards to process time variable data, HMM and DTW can be used in hand gesture recognition as well. The Hidden Markov Model is a statistical analysis model that can be used to describe the temporal and spatial variations of gesture signals. When applied to fingertip tracking and hand gesture recognition, this method has been proven to work well [8]. The Dynamic Time Warping is an algorithm for measuring similarity between two temporal sequences even though the lengths of the two sequences are different. According to a paper published in 2012, DTW performed better than HMM when applied to gesture recognition [9].

1.3 Thesis objective

With the development and popularization of motion controllers, interpreting hand or finger movements as character inputs using hand tracking devices will improve and evolve. The main objective of this thesis is to develop a program that employs the DTW algorithm to analyze and recognize three-dimensional hand gestures using Leap Motion Controller. The LM is able to detect and record hands' and fingers' movements in the air, and an abbreviated symbolic writing method called Teeline shorthand will be employed as gesture inputs. The program will perform the following functions:

- Reading templates from a database to identify hand gestures;
- Recording users' gesture inputs;
- Applying DTW to compare between templates and users' inputs to recognize Teeline shorthand;
- Allowing users to add their own templates to database.

The project is novel since it uses the Leap Motion Controller for gesture recognition. Although it works like other motion control devices, such as Kinect, Leap Motion focuses on tracking hand and finger movements, which makes it more precise and efficient than Kinect when tracking subtle movements. Many authors have illustrated the use of Kinect in finger-writing (see [10-12]). As a new motion control device, Leap Motion is rarely mentioned in hand gesture recognition. Next, rather than using English characters, this project requires the input of Teeline shorthand. The Teeline alphabet consists of characters that are simpler than the individual letters of the English alphabet, thereby simplifying the input and reducing the complexity of three-dimensional writing. In addition, the program allows users to build their own gestures into the database, which is a novel idea in the field of gesture recognition.

With this function, the usability and flexibility of this program can be improved.

1.4 Thesis outline

This thesis is organized as follows: The literature review is detailed in Chapter 2, which lists in detail the foundation of the Leap Motion Controller and Teeline shorthand used in this thesis. Chapter 3 introduces the method for gesture recognition applied in this thesis, and explains the functions of the program. The experiments that were conducted to test the recognition accuracy of the program and the results are presented in Chapter 4. Chapter 5 addressed further discussion about experiment results. Finally, Chapter 6 presents the conclusion of this project and suggests future areas of research.

Chapter 2

Literature Review

2.1 Leap Motion Controller

2.1.1 Overview

The Leap Motion controller is a small peripheral device (3 x 1.2 x 0.5 inches) that is designed to be placed on a physical desktop and connected to a computer. It can sense the objects observed in the device's field of view. The Leap Motion system tracks every movements of the hand, finger and finger-like tool, and reports discrete positions, gestures, and motions [13]. The Leap Motion controller uses optical sensors and infrared lights. The heart of the device consists of two cameras and three infrared LEDs. The sensors have a field of view of 150 degrees wide and 120 degrees deep when the LM is in its standard operating position, and it can detect objects in a range of 4 feet width by 1 inch to 2 feet height [14].

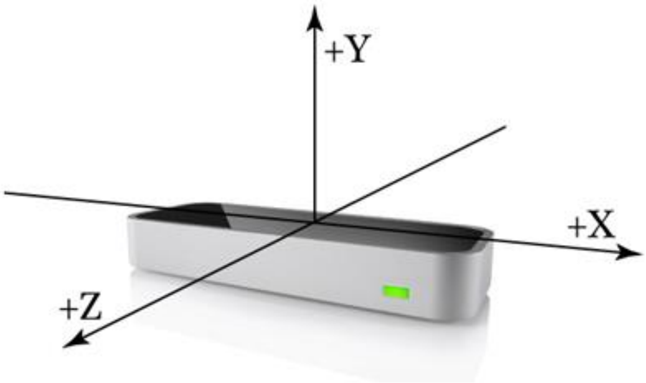

Figure 2.1 Leap Motion coordinate system

The Leap Motion works based on a right-handed Cartesian coordinate system (Figure 2.1), with the origin centered at the top in the center of the LM controller. The x- and z-axes lie in the horizontal plane, with the z-axis running parallel to the short edge of the device and increasing positive values toward the user. The y-axis is vertical and has positive values increasing upwards [15].

Leap Motion has the capability of tracking objects using the following units in terms of physical quantities: millimeters in distance, microseconds in time, and radians in angle. As stated by the manufacturer, the sensor's accuracy in fingertip position detection is approximately 0.01 millimeters; however, the studies in a paper published in 2013 demonstrated that under real conditions, the accuracy of Leap Motion is less than 0.2 millimeters for the static case and less than 1 millimeters for the dynamic case [16].

2.1.2 Motion tracking data

Leap Motion will provide a set of data as an update when it tracks hands, fingers, and tools in its field of view. This set of data is named *Frame* and it contains the measured coordinates of the current position and other information about each detected entity. The *Frame* object is essentially the root of the Leap Motion data model.

Hands Hands are the main entities tracked by the LM controller, and this model provides information about lists of the fingers associated with the hand (Figure 2.2). Since an internal model of a human hand is built inside the Leap Motion software to provide predictive tracking and validate the data from its sensors, LM is able to track finger positions even when parts of a hand are not visible [14].

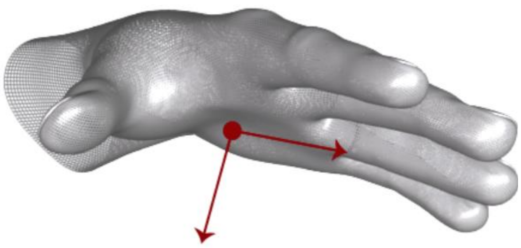

Figure 2.2 Hand Tracking

Fingers The Leap Motion controller provides information about each finger on a hand (Figure 2.3). With the internal model of hand, the finger positions will be estimated based on recent observations when part of a finger is out of LM's field of view. Fingers are identified by type name, i.e. thumb, index, middle, ring, and pinky.

Figure 2.3 Finger Tracking

Tools Tools are independent of hands, and these always recognized as being held like a pencil (Figure 2.4). The Leap Motion system defines tools as thin and cylindrical objects, longer, thinner, and straighter than a finger.

Figure 2.4 Tool Tracking

Gestures and Motions The Leap Motion controller also provides another two data models: gestures and motions. Gestures are classified as pre-defined movement patterns, and motions are recognized as the basic types of movements inherent in the change of a user's hands over a period of time. Representative gestures are swipe, key tap, and screen tap (Figure 2.5), for example, and motions include scale, rotation, and translation (Figure 2.6). By default, the recognition of 'Gestures and Motions' is disabled; however, this is a function that can be manually enabled.

Figure 2.5 A swipe gesture, a key tap gesture, and a screen tap gesture (from left to right)

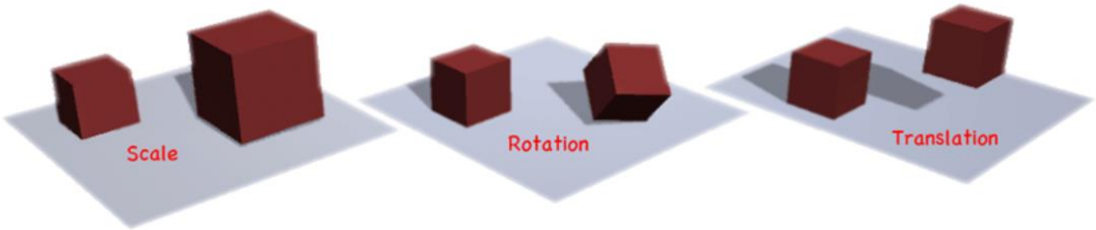

Figure 2.6 The scale motion, rotation motion, and translation motion (from left to right)

Since the tracking data for gestures and motions have not being applied in hand gesture recognition in this project, the gestures and motions model that allows LM to recognize them will be kept disabled. Furthermore, to simplify the recognition inputs, only hands and fingers data will be used in this thesis.

2.1.3 System architecture

The Leap Motion software receives motion tracking data via the USB bus that is connected to the Leap Motion controller device. This data is then transferred to a Leap-enable application. A Leap Motion Software Development Kit (SDK) is provided for the public to develop Leap-enabled applications; this comprises of two varieties of Application Programming Interfaces (API) in several programming languages including C++, Java, and JavaScript for getting the Leap Motion data from LM software [17].

A native interface is a dynamic library that developers can use to create new, Leap-enabled applications, and a WebSocket interface and JavaScript client library allow users to create Leap-enabled web applications. In this thesis, the writing recognition program uses the native interface through a dynamically loaded library; which will be expanded on later in this paper.

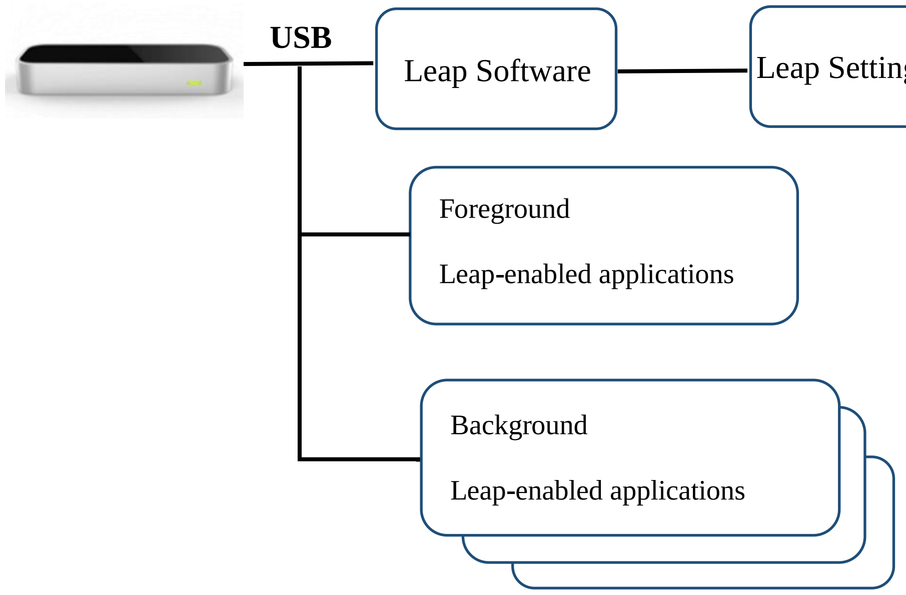

Figure 2.7 Native Application Interface

As shown in Figure 2.7, the Leap Motion application (Leap Setting App) is a Control Panel on Windows that allows users to configure the Leap Motion operations. The Leap Motion software (Leap Motion Service) takes advantage of the dynamically loaded library which is connected to it, to process information received from the controller and send it to the running foreground Leap-enabled applications by default. The foreground Leap-enabled application can receive the motion tracking data from the software and connect to the software to execute commands using the native library. Unless it receives a request from the application, the software does not send tracking data to a background Leap-enabled applications. Configuration settings for applications in background are determined by the foreground application [17].

The Leap Motion SDK supports many kinds of commonly used programming languages, such as Unity, C#, C++, and Java, and so forth. Since the Native Application Interface allows Leap-enabled applications to directly link to the library in C++, it was chosen as the programming language for this project.

2.2 Shorthand

2.2.1 Introduction to shorthand

Shorthand is any system of abbreviated symbolic rapid handwriting that can be used to transcribe the spoken word [18]. Many forms of shorthand exist. A typical shorthand system uses simplifying symbols or abbreviations for letters and characters, which, for instance, would help a well-trained journalist to accurately speed-write at the rate of the spoken word at press conferences or similar scenarios, without recorders or computers. Although primarily devised and used to record oral dictation or discourse, some systems of shorthand are used for compact expression; for example, healthcare professionals use shorthand notes for medical charts and correspondence [19].

The use of simplifying symbols in shorthand makes it easier to record English; this notion is also the case for Leap Motion: abbreviated letters and ‘shorthand’ codes in three-dimensional recognition makes the program easier to use. Moreover, hand gesture recognition using the Leap Motion controller, the simpler strokes are needed so users can avoid long periods of time in an uncomfortable position holding their hands up in the air. In conclusion, shorthand is a better option for inputting than English, and therefore, in this project, it will be applied in the writing recognition program.

2.2.2 Classification

The earliest known indication of shorthand systems is from the Parthenon in Ancient Greece, which lays out a writing system primarily based on vowels, using certain modifications to indicate consonants [20]. Many languages have their own shorthand systems. For example, an abbreviated, highly cursive form of Chinese characters were used for recording court proceedings in Imperial China, and an interest in shorthand developed towards the end of the 16^{th} century in England [20]. There are a number of different systems currently in use, and according to the shape, these can be classified as geometric, script, and semi-script shorthand.

The first modern shorthand systems were geometric. These are based on circles, parts of circles, and straight line placed strictly horizontally, vertically, or diagonally. Script shorthand was devised based on the motions of ordinary handwriting, and it is commonly used in countries such as Austria, Italy, and Russia now. Semi-script shorthand is also named Script-Geometric shorthand, and this system is a combination of the geometric systems and the script systems. The Teeline shorthand applied in this project is one example of a semi-script shorthand system. It is the most recommended shorthand method for journalists in the UK and New Zealand [21].

2.2.3 Teeline shorthand

Developed by James Hill in 1968, Teeline shorthand became a widely recognized method based on the English alphabet [22]. Teeline is a system that depends on reducing the letters of the English alphabet to their simplest possible forms, jointing characters together using a streamlined way to transcribe the words. Due to its flexibility and comparatively simple theory, Teeline shorthand gained popularity for its ability to adapt to the individual's own pattern of use.

Figure 2.8 illustrates the Teeline shorthand alphabet. As can be seen from this alphabet, some characters are similar to those in the English alphabet, such as ‘c’ and ‘v’, and some characters are far from their original forms like ‘f’ and ‘s’. However, all characters in Teeline shorthand are simple lines (curved or straight), which can be easy and logical to learn, and fast and accurate to use for beginners. The simplicity and understandability of Teeline shorthand are the critical reasons for choosing it for hand gesture recognition in this project, rather than other shorthand systems.

Figure 2.8 Teeline Shorthand Alphabet

In order to take advantage of the Teeline alphabet in the hand gesture recognition program, some revisions of the system were made. A revised version of the Teeline shorthand alphabet is shown in Figure 2.9. In the revised Teeline alphabet, character ‘x’ was modified from two strokes to one stroke, so user just need to do one gesture to draw a ‘x’ like drawing other characters. Furthermore, a new character to represent ‘space’ was added to this revised alphabet, allowing people to insert a space between the two words they draw by waving this gesture rather than having to move hands to a keyboard and type it in. The arrow beside each character in the Teeline alphabet indicates the direction in which a character should be written. In maintaining this, the input will be formatted and therefore recognized by the program.

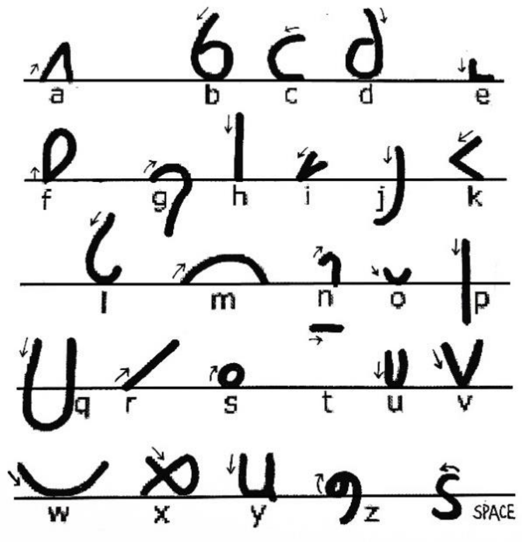

Figure 2.9 Revised Teeline Alphabet

2.3 Programming language and analysis software

In the last few decades, various programming languages have been created, superseded, modified or combined; and one of them is the C++ programming language. C++ can be easily understand and developed; it is supported by the LM's SDK; and its library can be directly linked by the Leap-enabled application; all of the above make it a powerful and effective programming language for this project.

In order to more efficiently analyze the research data, statistical analysis will be applied using IBM SPSS Statistics 21. It is a highly efficient program that allows users to enter data, run analyses, as well as display results in tables and graphs within a matter of minutes [23, 24]. The benefits of SPSS including effective data management, wide range of options, and clear output organization, which makes it an effective software for the statistical analyses in this thesis [25].

Chapter 3

Design

3.1 Program algorithm-Dynamic Time Warping

Gesture recognition interprets human gestures by using mathematical algorithms to translate these gestures into computer language; the aim is to enable people and machines to communicate more easily [26]. The hand gesture recognition algorithms can be divided into three main categories: template matching-based algorithms, statistics-based algorithms, and data classification-based algorithms. Dynamic Time Warping (DTW) is a well-known template matching technique with the advantages of simple principle and flexible operation [27]. DTW was originally developed in automatic speech recognition to cope with different speaking speeds [28, 29], and has been widely used in handwriting recognition, image analysis, and many other fields [30-33].

3.1.1 Euclidean distance

In mathematics, the Euclidean distance is the straight-line distance between two points in Euclidean space.

If $p = (p_1, p_2, \dots, p_n)$ and $q = (q_1, q_2, \dots, q_n)$ are two discrete points in Euclidean space, then the distance (d) from p to q, or from q to p is given by the formula [34]:

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

3.1.2 Dynamic programming

Euclidean distance is an efficient method for calculating the distance between two sequences with same length; however, in many cases, there are possibilities where the lengths of two time-series are unequal, see Figure 3.1 and Figure 3.2.

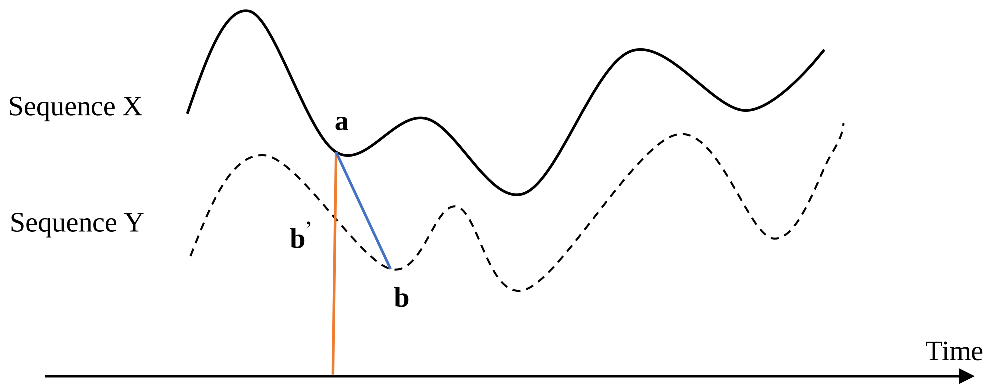

Figure 3.1 Euclidean distance of two time-dependent sequences X and Y

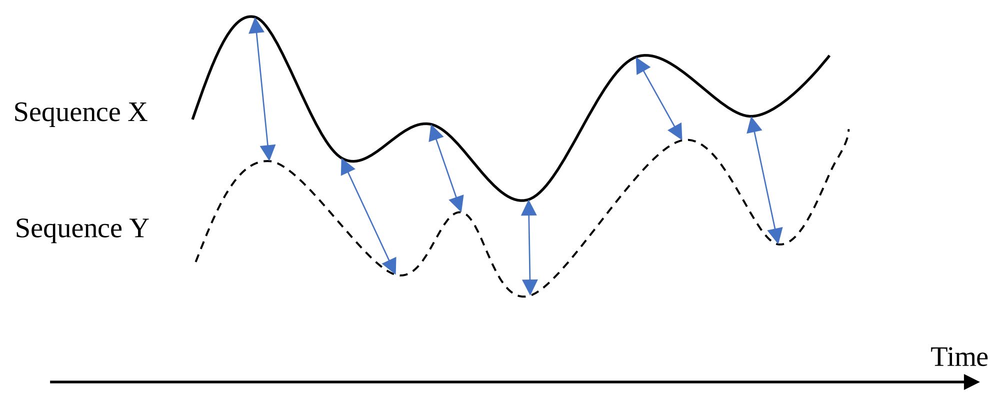

Figure 3.2 Time alignment of two time-dependent sequences X and Y

Sequence X and sequence Y are two similar time-dependent sequences. When using Euclidean distance to decide the distance between X and Y, the corresponding element of a in sequence X is b' in sequence Y. However, the actual correspond element of a in sequence Y is b. Therefore, Euclidean distance becomes ineffective for calculating $distance$ between two sequences in different lengths. Unlike Euclidean distance, Dynamic Time Warping (DTW) is an algorithm that uses dynamic programming to find an optimal alignment between two given (time-depend) sequences.

Consider two time-dependent sequences $X \coloneqq (x_1, x_2, \dots, x_m)$ of length $m \in \mathbf{N}$ and $Y \coloneqq (y_1, y_2, \dots, y_n)$ of length $n \in \mathbf{N}$, where x_i and y_j are elements at index i and j in X and Y, respectively. Each element can be a vector with dimension K, which represents a measurement at a certain time or position. In order to align these two sequences, a $n \times m$ matrix needs to be built. The element $d(i, j)$ in position (i, j)

of this matrix represents the distance between elements x_i and y_j, it is determined by Euclidean distance: $d(i,j) = \sqrt{(x_i - y_j)^2}$. In other words, dynamic programming is an algorithm that looks for a warping path through numbers of elements in the matrix; the elements are the distances when the $i-th$ element in X is aligned to the $j-th$ element in Y [35].

An alignment from X to Y can be represented by a warping path $w = \{w(1), w(2), \dots, w(k), \dots, w(K)\}$, where $max(m,n) \leq K \leq m+n-1$. The warping path satisfies the following conditions [36]:

1) Boundary condition: $w_1 = (1,1)$ and $w_K = (m,n)$.
2) Monotonicity condition: if one point in the path is $w_{k-1} = (a', b')$, then for the next point $w_k = (a,b)$ in this path, the conditions $(a-a') \geq 0$ and $(b-b') \geq 0$ are always true.
3) Local continuity condition: if one point in the path is $w_{k-1} = (a', b')$, then for the next point $w_k = (a,b)$ in this path, the conditions $(a-a') \leq 1$ and $(b-b') \leq 1$ are always true

Constrained by the three conditions above, there are only three directions for a point to follow at position (i,j) in a warping path, including $position(i+1,j)$, $position(i,j+1)$, and $position(i+1,j+1)$. When the number of the elements increase in a sequence, however, the valid warping paths will increase exponentially.

For instance, with the distance measure (e.g. Euclidean distance) $d(i, j)$, the accumulated distance $D(i, j)$ along warping path w can be calculated by [35]:

$$D(i, j) = d(i, j) + min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\}$$

$$for\ 1 \leq i \leq m, 1 \leq j \leq n$$

The object of DTW is to find the warping path w, which minimizes the distance $D_w(X, Y)$, and the DTW distance between X and Y is calculated by:

$$DTW(X, Y) = D(m, n)$$

Typically, when X and Y are more similar to each other, $DTW(X, Y)$ is smaller; otherwise $DTW(X, Y)$ will be larger [27].

3.2 Program implementation

In this section, the implementation of the project will be presented. First, a brief introduction to the framework of the project will be illustrated. Following this, two different modes of the program will be detailed. In the third section, the database used in this project will be described. Finally, the interface of the program will be explained.

3.2.1 Overall design

There are five main components of the hand gesture recognition program: the gesture data input by the user, the Leap Motion controller that tracks and records hand movements, the display window that shows the movement path, the console window that receives commands and gives output, and the database that stores templates for the matching algorithm.

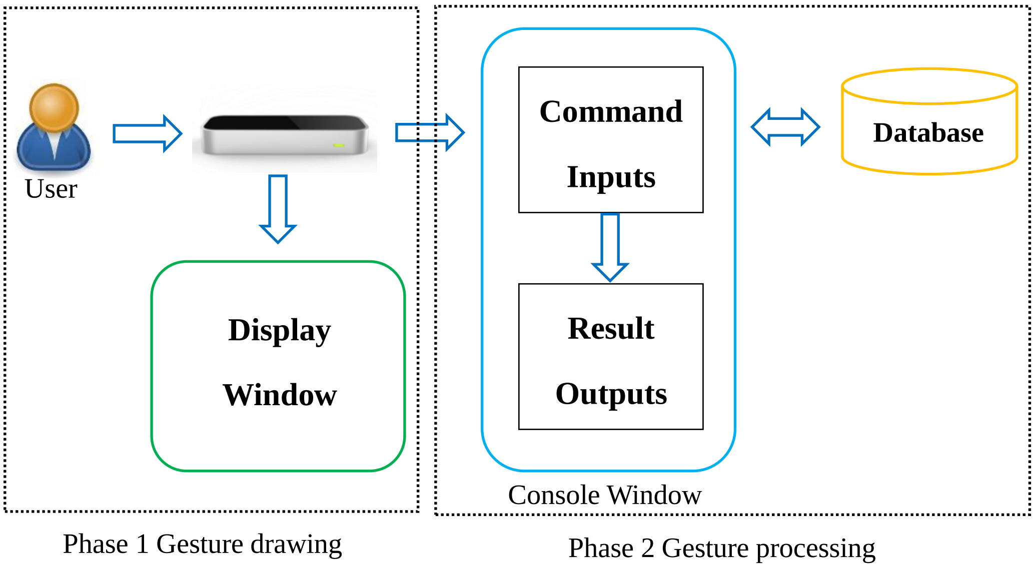

Figure 3.3 Program Overview

Figure 3.3 demonstrates how the program works. It can be divided into two phases. During the first phase, the Leap Motion Controller (LM) captures Teeline shorthand gestures done in the air. The LM detects and records the user's hand movements and then sends the data to the display window on computer monitor where the user's hand movement path will be depicted in 2D space. In the second phase, with the hand gestures completed, the user inputs commands using a separate window from the display window called console window. The program will compare the gesture tracking data to the templates in the database and then output a recognition result in English letter(s) to display on the console window. In Recognition mode, the database includes sample gesture data for analyzing users' input gestures data. In Edit mode, the program allows users to save gesture information and thus add new templates to the database. These two modes of the program, Recognition mode and Edit mode, will be illustrated in the next two subsections.

3.2.2 Recognition mode

Recognition mode is the core function of the program. In this mode, there are two different input sources: the gesture information drawn by the user, and the database includes templates for matching algorithm. In Recognition mode, only one output is allowed: the recognition result analyzed by the program using the Dynamic Time Warping algorithm. In order to deliver a clear explanation, the program's work process in Recognition Mode is expounded in Figure 3.4.

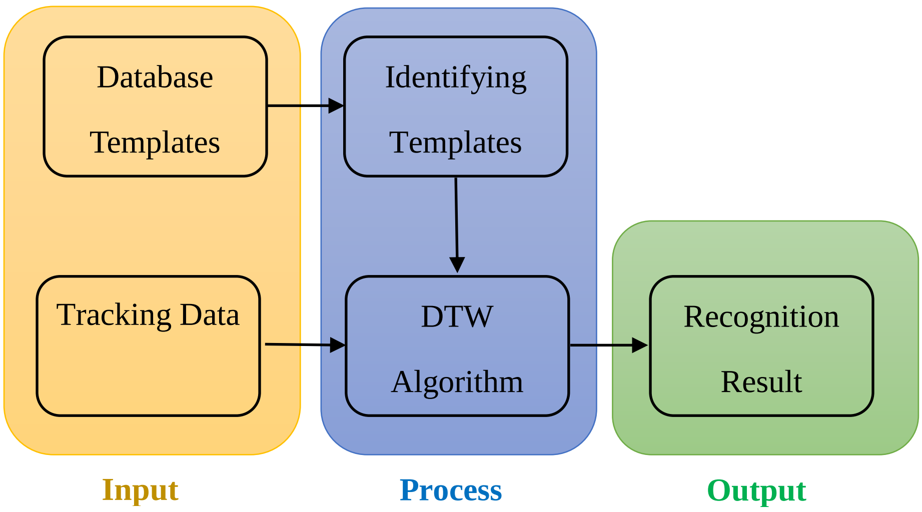

Figure 3.4 Recognition Mode

At the beginning of this mode, the program will retrieve each template from the database. It matches each letter in English with the corresponding Teeline shorthand gesture template in the database, thus assigning each Teeline shorthand template an English letter as its name. This process can be recognized as the program is identifying templates. When the user finishes her/his Teeline shorthand gestures, the Leap Motion Controller will record the gesture tracking data and send it to the program as test samples. Following this, the DTW algorithm will be called to compare the similarities of each test sample to each template in the database. The similarities are represented by the DTW distance, and the shorter the distance, the higher the similarity. When the smallest value of the DTW distance between the test sample and a template in database is found, the English letter name of that template will be assigned to the test sample. After the program processes through all the test samples, it will output the name of each sample in sequence as the recognition result.

3.2.3 Edit mode

Some gesture recognition systems suffer from the severe limitation that an end-user is unable to add any new gestures to the pre-existing database of gestures [37]. Such limitations mean the system are unadaptable; users are restricted and cannot customize the program based on their own habits and preferences, thus lowering its usability. For this reason, several applications sold in Apple App Store have added a function that allows end-users to define new gestures in the application library. For example, an app called Short Hand by LizzardWerks in the Apple Store allows users to create their own shortcuts. Once installed and programed, the user can type out these shortcuts commands and they will be changed into full words and sentences [38].

Taking the above into consideration, for the purpose of increasing the utility of the program, an Edit mode is designed besides the Recognition mode. This mode allows users to create their own gesture templates and add them into the database for future use.

Figure 3.5 Edit Mode

It can be seen from Figure 3.5, the program’s work process in Edit mode is quite simple. Users design their own gestures with the Leap Motion Controller, which records the tracking data and then sends it to the program. The program saves this tracking data to a file as a template that specified by user using a filename and then outputs it to the database. At the end of Edit mode, a new template is added to the database, and if the new gesture is used again, the template can be identified by the program in the Recognition mode.

3.2.4 Database

In this project, the database is a folder under the C++ release folder. There are 270 Teeline shorthand templates included in this folder, and every template is a text file. The file contains the movement path for each Teeline shorthand character in the form of points with x-axis and y-axis positions. Recalling Figure 2.9 in Chapter 2, a revised Teeline shorthand alphabet of 27 characters was applied in this project. Therefore, 10 templates are allocated for each Teeline character from “a” to “z” plus a space.

At the beginning of this project, all 270 templates in the database were built by one person: the author. With regard to the technique applied for gesture recognition in this program, Dynamic Time Warping, which yields results by comparing the similarities between templates and test samples. Therefore, there is a need to increase the diversity of the templates for each Teeline character by building another database by various persons instead of one. A database built by different people includes diverse writing styles, which may help recognize various gesture test samples and increase the recognition accuracy of the program. In the next chapter, however, the building of an additional database by collecting samples from different users will be described; that is, collecting one template for each character in the Teeline alphabet from each person and building a database of 270 samples from ten different people. These two databases are applied respectively in order to compare and review the recognition accuracy of the program, this will be further detailed in Chapter 4.

3.2.5 Implementation flow

Figure 3.6 gives the flowchart of the whole program:

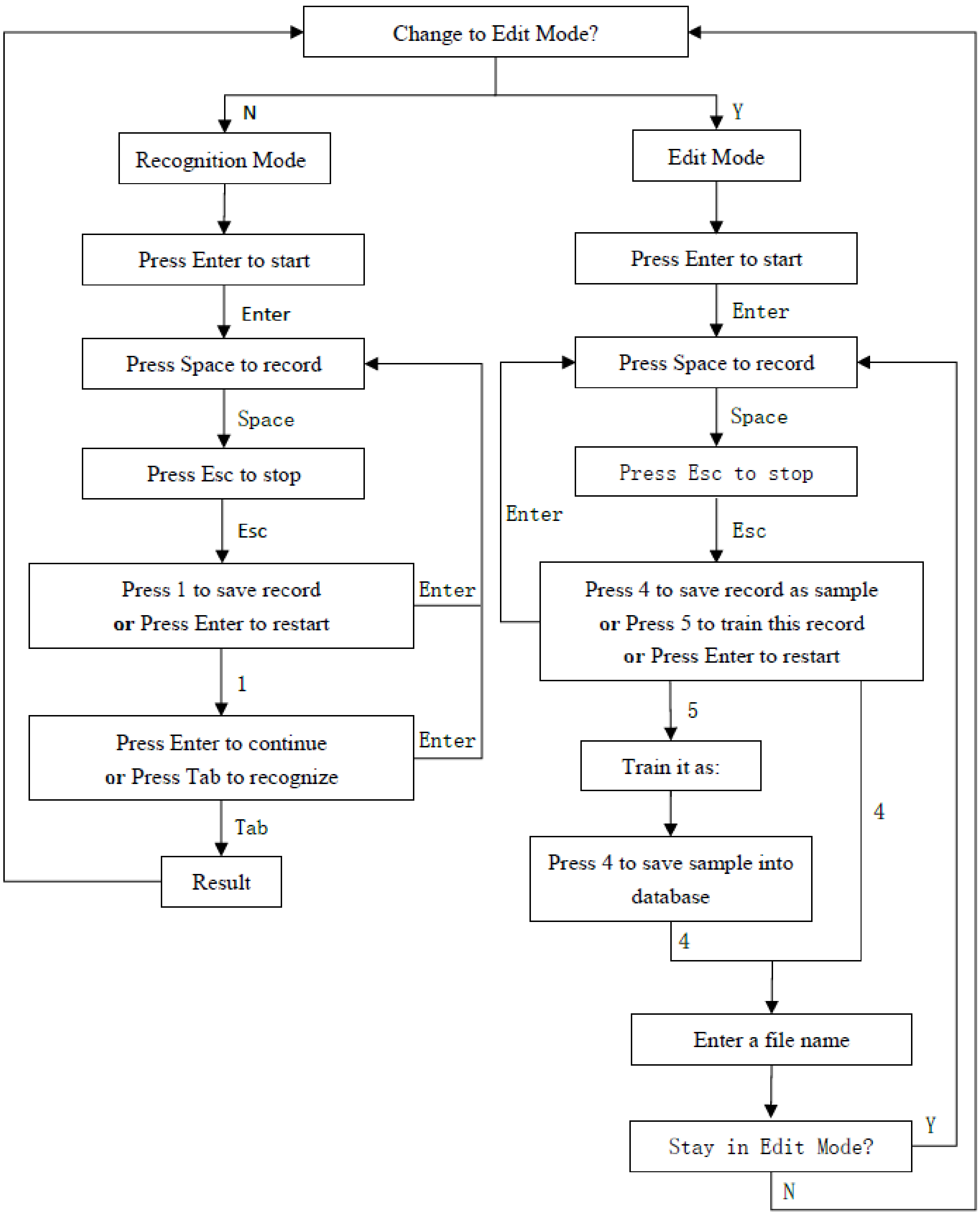

Figure 3.6 Flowchart of the Program

The whole program executes as shown above. At the beginning of the program, users can choose one of the two modes by pressing “Y” or “N” keys according to their demands. Whether in Recognition mode or in Edit mode, there are several steps that are the same for the users to follow; in other words, both modes need the same input commands, as follows:

- Enter command: this command is used for activating the display window on screen.
- Space command: this command is used for directing Leap Motion to start to track hand movement and record tracking data.
- ESC command: this command is used for stopping the hand tracking and guiding the program to proceed according to the following instructions.

In the Recognition mode, the user can draw one Teeline character at a time and save it temporarily using input command “1”. If the user wants to draw more than one character, she needs to input the “Enter” key again to draw the next character and followed by “1” to save the next record until all characters the user intends to draw have been completed. The total number of characters drawn is displayed on the console window by inputting the “Tab” and the recognition results will be given as well.

In Edit mode, there are three options for the user after s/he creates a new gesture. By inputting command “4”, the program will guide the user to save the record as a template in the database; and the user will need to name the new template file. Command “5” gives the user access to train the new gesture template, which allows the user to assign a meaning (an English letter or word) to the new gesture. Furthermore, command “5” is followed by command “4”, which means the user is still asked to save it into database. If the user is unsatisfied with the gesture, the “Enter” key command allows the user to discard it if preferred. If the user intends to add more than one new gesture into the database, s/he only needs to input “Y” after the prompt “stay in Edit Mode”, and then repeat the above steps.

3.2.6 Program interface

The program interface consists of two parts: the console window and the display window. They are illustrated in Figure 3.7 and Figure 3.8 respectively.

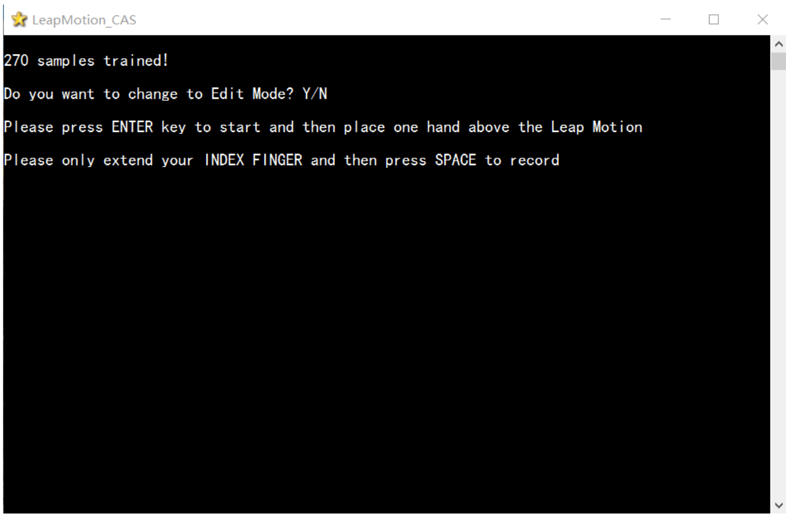

Figure 3.7 Console Window

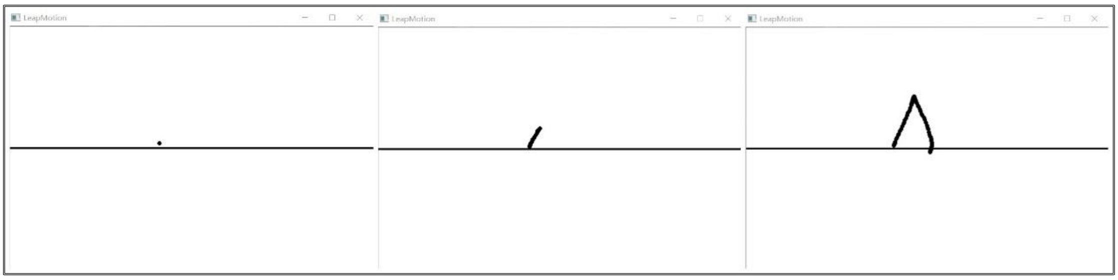

Figure 3.8 Display Window (when drawing a Teeline character “a”)

Using the console window in Figure 3.7, users are able to input commands to control the flow of the program and read the recognition result at the end. The hand gestures are made in the air using the motion control device and are consequently invisible; the display window, therefore, is needed for users to monitor their hand movements and help them finish the gestures. The user’s hand movement path is illustrated in the Display Window, Figure 3.8. The line on this window is a reference line correspond to the line on Figure 2.9, which helps the user decide the position of each Teeline shorthand character. The little black point on the screen is like a penpoint indicating the position of the user’s index fingertip. When the Space key command is received by the program, the display window will show the user's hand movement path as the user drawing Teeline gestures. The action mimics the behavior that drawing on the screen, but without the user actually touching the screen itself. When a gesture is done and the "ESC" key command is received by the program, the display window will only show the path that the user just drew. Until the next "Enter" key command is made and user put their hand above the Leap Motion, the display window will be immediately refreshed as in Figure 3.8.

Chapter 4

Evaluations

In user-centered interaction design, usability testing is a technique to evaluate a product or system by running trials with users. This gives designers the opportunity to see how end-users interact with a product or system, and is thus an irreplaceable usability practice [39]. Usability testing focuses on measuring a human-made application's capacity to meet its intended purpose. The crucial objective for a hand gesture recognition system like the one in this project is to correctly recognize users' gestures. For this program, the most important aspect of the usability test is testing recognition accuracy.

4.1 Testing preparation

In order to test the recognition accuracy of the program in this project, a number of users will try out the application and evaluate the system. The usability testing will be conducted at Laurentian University, and students will be recruited as the participants of the experiments. According to the requirements of the Office of Research Services, all research involving human subjects that is conducted at Laurentian University must be reviewed and approved by Laurentian University Research Ethics Board (LU's REB) in advance to ensure compliance with the highest ethical standards of the

Tri-Council Policy Statement (TCPS).

Before proceeding with the usability testing, a few documents about the project were submitted to LU's REB for consideration. The documents included the research proposal, and the ethics form that describes the potential risks and benefits to human participants. The documents were reviewed by the REB and an approval letter for this usability test was issued (Appendix 2).

4.2 Questionnaire

Questionnaires are one of the most important parts of the market research process. They are the means by which the responds of target respondents are transformed into quantifiable variables, and they are the measuring device for things that are not directly observable [40]. For this project's usability test, a questionnaire will be used to collect the participant's background information and related experience on games and motion control products; the information collected will be helpful for further comparison and analysis among all participants to find out the relation between users' experience and the recognition accuracies.

Following the suggested questionnaire design principles (see [41-44]), and based on the information the testing was required to collect, a questionnaire was designed for the experiments in this paper (Appendix 3). Concentrating on the participants, the questionnaire focused on five categories: their English capability (Q1-Q2), their demographic profiles (Q3-Q7), their knowledge of shorthand (Q8-Q9), their experience of video games (Q10), and previous experience with any kinds of gesture-controlled products (Q11-Q13). All the questions were closed-ended questions, except for Question 3, which inquired about the program the participant is in. All questions were organized into groups based on categories and were ordered in a logical sequence.

The first two questions were designed to see if the participant would be able to use the program independently, since the prompts on the console window are in English and the participant would need to input commands according to the different prompts.

The demographic profile of the participants includes their program, age range, gender, mother tongue and dominant hand, all of which will be used for classifying participants into various groups to compare the differences in program recognition accuracies. The target responders of this project are students at LU, therefore age has been broken up into eight balanced groups, with each group having a five-year range. The mother tongue information will indicate participants' normal writing habits, such as from left to right or from right to left. A participant's dominant hand is crucial information to analyze if the program is accommodating to both right-handed and left-handed people.

This project's hand gesture recognition program uses Teeline shorthand; therefore, it is necessary to collect data regarding the participants' experience with shorthand. The data will be applied in analyzing whether the program can accommodate to new as well as experienced shorthand users.

Much of psychological research has been focused on the negative aspect of gaming effects (see [45-47]), but Isabela Granic has suggested that people may need a more balanced perspective to understand the influences of video games [48]. Some authors have focus on exploring both the positive and negative effects of video games, and the authors found that playing video games benefits people in several ways [48, 49]. In comparison with those who do not play video games at all, people who play experience cognitive benefits, motivational benefits, emotional benefits, and social benefits [48]. Thus, Question 10 was included in the questionnaire to test that whether people who play video games will better handle the program (gain higher recognition accuracy).

Since this project centers on motion control and gesture recognition, the three final questions of the questionnaire are related to the participants' experience with gesture-based interfaces and motion control technology. Data collected from these questions will be used to validate that whether or not the more experience a user has with gesture interfaces or motion control devices, the easier the program is for him/her to use and the more accurate the recognition result will be.

In total, the questionnaire consists of 13 questions, each of which should be easily answered, so the predicted time for completion is 5 minutes. This duration is long enough for participants to fill the questionnaires and does not take up too much of the whole experiment's time, which is acceptable for the usability testing.

4.3 Pilot testing

4.3.1 What is pilot testing?

'Pilot testing', also referred to as a 'pilot experiment' or 'pilot study', is a small-scale trial, where a few participants take the test and comment on the mechanics. It is a quick and convenient way to evaluate feasibility, time, cost, adverse events, and statistical variability in an attempt to predict an appropriate sample size and improve upon the study design prior to performing a full-scale research project [50].

Pilot testing is particularly important in the following situations [51]:

- If the conductor will run a usability test for the first time;
- If the conductor will test an unfamiliar subject area;
- If a remote, unmoderated study needs to be conducted;
- If a high-visibility project will be involved in the test;
- If conductor is prepared to work on a one-shot research project.

Not only the novel practitioners, but also veteran usability practitioners can benefit from running pilot tests.

4.3.2 Conducting the pilot testing

In order to find out if the tasks are clear in the experiment, if the data collected from the tests can be used, and how much time to schedule for testing one user, a few pilot experiments were conducted prior to the full-scale study.

For the pilot testing, four participants from Laurentian University were recruited; two females and two males, from different departments including Science Communication, Computational Science and Business Administration. Three participants were in age range of 21-25 years old and one participant was in the age range of 26-30 years old. Participants were tested one by one, each one given the same tasks in the same order throughout the whole process.

Based on the pilot studies, the average time for participants to complete all the tasks was about 30 minutes. Each participant had to complete the following tasks: reading a consent form and completing a questionnaire (about 10 mins), looking at the Teeline shorthand alphabet and practicing doing gestures using Leap Motion controller (about 5 mins), and recoding Teeline characters using the program (15 mins). Depending on the participant’s practice time and their speed, each test duration varied slightly. In addition, the following deficiencies were also found after the pilot tests:

- For most people who do not know about shorthand, Question 8 in the questionnaire seems unclear for them to answer;
- Since the program needs keyboard inputs as well as gesture control, new obstacles were brought about, wherein users had to execute the correct commands in order to keep the program running fluently; for instance, the program’s response did not meet the user’s expectation if no/incorrect commands were given to the program;
- The input method of the laptop affects the operations of the program.

Some modifications were made to solve the problems found in pilot testing:

- Being present when the participants filled out the questionnaire and explaining the meaning of shorthand to them in detail;

- Connecting an extra keyboard to the laptop so that command inputs can be done by the conductor instead of the participants, this allowing participants to focus on drawing gestures alone;
- Ensuring the input method is in English before starting the experiment.

The changes listed above were applied in the full-scale experiments which will be described in next two subsections.

4.4 Experiment I

4.4.1 Objective

The objective of Experiment I is to test the influence sample size in the database has on the recognition accuracy of the program. As stated in subsection 3.2.4 Database, for this project one database (Database 1) has already been built. It contains10 samples for each symbol in Teeline alphabet (Figure 2.9), for a total of 270 samples for the 27 Teeline characters. In order to compare the difference between the recognition accuracies using different databases, another database will be built by the researcher using samples from population. Both databases will be applied in this experiment, and the next experiment in section 4.5 Experiment II.

4.4.2 Hypotheses

Referring to subsection 3.2.2 Recognition Mode, the templates in the database will be processed and identified in Recognition mode and then compared to the user inputs to obtain a recognition result. Two databases are used in this project, and they have the same properties, but differ in the writing styles. The database built by one individual has all the samples in a unified style since it was created by one person. However, the database of templates made by the population has ten different styles of writing each character from ten individuals. As a result, the following hypotheses arose:

Hypothesis 1: The more templates the database includes for matching algorithm, the higher the program recognition accuracy will be;

Hypothesis 2: There is an optimal sample size for each Teeline character, so that the recognition accuracy remains nearly constant even when the sample size for each character increases past that number in the database;

Hypothesis 3: The source of the templates in the database will not significantly affect the optimal sample size.

In order to verify the above hypotheses, the process of Experiment I is illustrated in subsection 4.4.3 Methodology.

4.4.3 Methodology

Eleven participants were recruited for Experiment I. In the first part of Experiment I, the first ten participants were asked to create templates for each Teeline shorthand character to build a database; and in the second part of Experiment I, one participant was recruited as the tester to evaluate the hypotheses mentioned in subsection 4.4.2 Hypotheses.

Ten out of eleven practitioners will independently accomplish the same tasks step by step as follows:

1. Read through a consent form and sign it if s/he agrees to participate.
2. Fill out the questionnaire (Appendix 3).
3. Review the Teeline shorthand alphabet as in Figure 2.9 (the alphabet was reproduced in a large-sized font, and attached to the wall or placed on the participant's table as desires)
4. Practice drawing Teeline gestures using Leap Motion controller to get familiar with the LM's sensing area and the program's running process.
5. Draw one Teeline character at each time, which the conductor will save to the new database (Database 2); only one sample is needed from each person for each character. Once all 27 Teeline shorthand symbols have been completed and saved, the participant will have completed Experiment I.

In the first part, five females and five males were recruited to be the first ten participants to build Database 2. They had diverse academic backgrounds including

Economics, Business Administration, Computational Science, Biomedical Biology, Chemical Engineering, Mining Engineering, Zoology and Ecology. All of the participants' English skills were good enough to successfully carry out the tasks of the experiment. The ages of the participants ranged from 16 to 25, with the exception of one participant, whose age range was 31 to 45. All of the participants were right-handed, and none of them previously knew or had used any form of shorthand. The other data collected from their questionnaires has been summarized in Table 4.1 below.

Table 4.1 Data Collected from Questionnaires

CATEGORY / ID NUMBER	EXPERIENCE WITH VIDEO GAMES (PER DAY)	EXPERIENCE WITH TOUCHED GESTURE INTERFACES	EXPERIENCE WITH MOTION CONTROL DEVICES	EXPERIENCE WITH LEAP MOTION CONTROL
1	3-5hrs.	Yes	Few times a year	None
2	Less than 1 hr.	Yes	None	None
3	1-3 hrs.	Yes	Few times a year	Several times
4	None	No	None	None
5	None	Yes	Few times a year	None
6	3-5 hrs.	Yes	None	None
7	None	Yes	Few times a year	None
8	Less than 1 hr.	Yes	Few times a year	None
9	None	Yes	Few times a year	None
10	None	Yes	Few times a year	None

All of the information collected from Question 10 to Question 13 in the questionnaires is listed above. The first column of the table represents the participant's ID, which was given based on the order in which they took part in the experiment, while the other columns represents their answers for specific questions. In Experiment I, the records of all ten participants were used as the templates in Database 2 rather than as test samples. Therefore, the information provided in the second column, which details each participant's experience playing video games

(Question 10), is not particularly relevant and will not be referred to within this subsection. After observing the data, the answers that were provided for Questions 11 to 13 were quite similar across participants. The answers provided regarding participants' experience with motion control devices, which were "few times a year" and "no experience", both mean that they had very little experience in this aspect. It can then be concluded that those individuals who built Database 2 have all had prior experience carrying out gestures on a touch-screen device, although they are novices to shorthand and motion control products, specifically, the Leap Motion controller. From now on, this research will regard Database 1 having been built by an experienced user (in both two aspects including manipulating motion control product and using shorthand), whereas Database 2 will be considered to have been created by novice users.

The new database (Database 2) had the same total number of samples as the older database (Database 1). The eleventh participant in Experiment I was recruited to test the performance of the program using the different databases. The tester was asked to do the following tasks:

1. Read through a consent form and sign it if s/he agrees to participate.
2. Fill out the questionnaire (Appendix 3).
3. Review the Teeline shorthand alphabet as in Figure 2.9.

4. Practice drawing Teeline gestures using the Leap Motion controller to get familiar with the LM's sensing area and the program's running process.
5. Draw each Teeline character ten times; the record will be saved by the conductor.

In the second part, the eleventh participant was independent of any of the templates in the two existing databases. In other words, no sample created by the participator was in any of the two databases. The questionnaire reveals that the eleventh participant has had no prior experience with Teeline shorthand or the Leap Motion controller.

Once the tester finished all 27 characters, there were 270 test samples in the records. The recognition accuracies of the program using each of the databases were evaluated as shown in Figure 4.1.

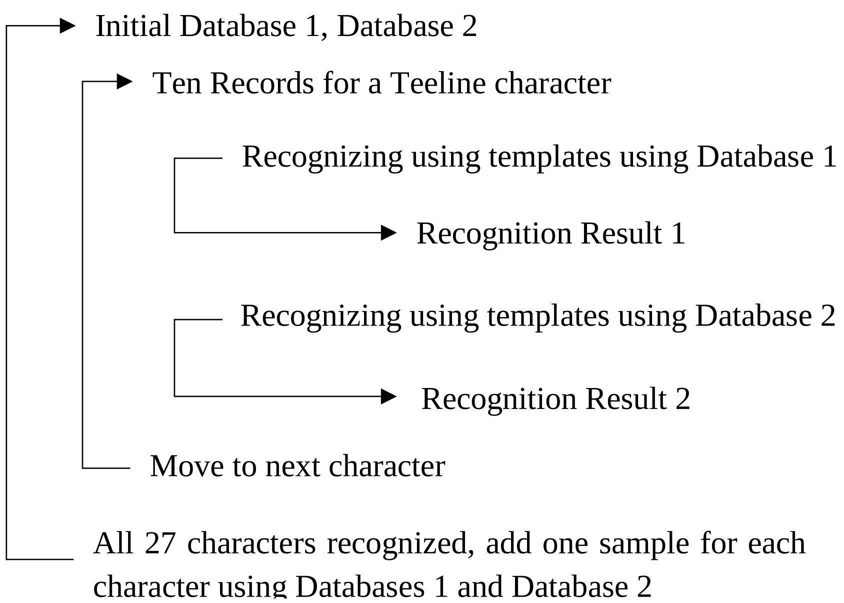

Figure 4.1 Evaluation Process in Experiment I

As seen in Figure 4.1, the 27 Teeline characters are recognized separately. The two initial databases include only one template for each character, that is, 27 samples in total in each database at first. The initial databases were used to recognized 10 test samples for each Teeline shorthand character done by the eleventh participant from ‘a’ to ‘z’ plus ‘space’. The recognition results were recorded and the accuracies are obtained. Then one more sample is added for each Teeline character using Database 1 and Database 2 respectively (a total 54 samples in each database), and the recognition process above is repeated. These final recognition accuracies are obtained until there are 10 samples for each character in the two databases, and the loop was ended. Following this procedure, the recognition accuracies of each Teeline shorthand characters at various database sample sizes (from 27 to 270) using two different databases are recorded. The results will be addressed in next subsection.

4.4.4 Results

The test samples that were drawn by the eleventh participant in Experiment I were recognized using Database 1 and Database 2 with a gradually increasing number of template gestures included. The recognition accuracies are listed in Table 4.2 and Table 4.3.

Table 4.2 Variation in Recognition Accuracy with Increasing Sample Size using Database 1

Sample Size / Character	1	2	3	4	5	6	7	8	9	10
A	50%	50%	50%	60%	60%	100%	100%	100%	100%	100%
B	80%	80%	70%	60%	70%	80%	80%	90%	90%	90%
C	0%	0%	20%	70%	90%	80%	80%	90%	90%	80%
D	10%	40%	70%	80%	90%	90%	90%	100%	100%	100%
E	40%	40%	80%	90%	80%	80%	90%	90%	90%	100%
F	20%	20%	50%	60%	80%	90%	90%	90%	90%	90%
G	90%	90%	100%	100%	100%	100%	100%	100%	100%	100%
H	100%	80%	90%	90%	90%	100%	100%	100%	100%	80%
I	90%	100%	90%	100%	100%	100%	100%	100%	100%	100%
J	70%	90%	90%	100%	100%	100%	100%	100%	100%	100%
K	100%	90%	40%	20%	40%	50%	30%	20%	80%	90%
L	100%	100%	100%	100%	90%	100%	100%	100%	100%	100%
M	0%	40%	30%	20%	40%	40%	100%	100%	100%	100%
N	80%	80%	90%	90%	90%	90%	100%	100%	100%	100%
O	40%	70%	70%	80%	80%	80%	80%	80%	80%	80%
P	90%	90%	100%	90%	90%	90%	90%	90%	90%	90%
Q	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
R	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
S	70%	80%	90%	100%	100%	100%	100%	100%	100%	100%
T	60%	60%	60%	60%	60%	60%	60%	60%	60%	60%
U	90%	100%	100%	100%	100%	100%	100%	100%	100%	90%
V	20%	70%	40%	70%	90%	100%	100%	100%	100%	100%
W	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
X	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Y	100%	80%	90%	90%	90%	90%	90%	90%	90%	90%
Z	60%	60%	50%	70%	80%	90%	90%	90%	90%	90%
Space	100%	70%	90%	90%	90%	100%	100%	100%	100%	100%

Table 4.3 Variation in Recognition Accuracy with Increasing Sample Size using Database 2

Sample Size / Character	1	2	3	4	5	6	7	8	9	10
A	50%	60%	70%	70%	70%	70%	70%	70%	70%	100%
B	50%	70%	80%	80%	80%	80%	90%	80%	80%	70%
C	60%	70%	80%	80%	80%	90%	90%	100%	100%	100%
D	20%	70%	100%	90%	90%	100%	100%	90%	90%	90%
E	60%	70%	70%	70%	70%	80%	80%	80%	80%	80%
F	80%	80%	90%	90%	90%	90%	90%	90%	90%	90%
G	0%	40%	20%	80%	50%	50%	50%	50%	60%	70%
H	10%	10%	30%	70%	100%	100%	100%	100%	100%	100%
I	50%	100%	90%	90%	100%	100%	100%	100%	100%	100%
J	0%	100%	100%	100%	100%	100%	100%	100%	100%	100%
K	0%	50%	80%	100%	100%	100%	100%	100%	100%	100%
L	50%	30%	70%	100%	100%	100%	100%	100%	90%	90%
M	40%	50%	80%	90%	100%	100%	100%	100%	100%	100%
N	60%	50%	100%	90%	100%	100%	100%	100%	100%	100%
O	50%	70%	60%	50%	60%	70%	70%	80%	80%	80%
P	70%	80%	30%	60%	40%	10%	80%	80%	90%	90%
Q	100%	100%	100%	90%	90%	90%	90%	100%	100%	100%
R	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
S	60%	60%	60%	60%	70%	100%	100%	100%	100%	100%
T	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
U	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
V	90%	90%	90%	100%	100%	100%	100%	100%	100%	100%
W	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
X	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Y	70%	70%	70%	70%	100%	90%	100%	100%	100%	100%
Z	10%	10%	20%	20%	20%	50%	50%	60%	70%	70%
Space	100%	100%	90%	80%	90%	90%	90%	90%	90%	90%

The first columns in Table 4.2 and Table 4.3 list all of the 27 characters in the Teeline shorthand alphabet, and the first rows of the tables represent the number of templates for each character in the two databases. For example, the “1” represents that there was only one sample for each Teeline character using Database 1 and Database 2, respectively, while the “10” represents that there were ten templates for each Teeline character in each of the two databases. Since there were ten test samples generated by Participant 11 for each Teeline character, the recognition accuracy for each character at each sample size is calculated by the following equation:

$$Acc = (\text{the number of the correctly recognized test samples}/10) \times 100\%.$$

In order to verify the hypotheses in subsection 4.4.2 Hypotheses, and examine the true nature of the relationship between the recognition accuracies and the sample size for each character, the data in Table 4.2 and Table 4.3 were imported to SPSS to be analyzed. The scatter plots are displayed in Figure 4.2 and Figure 4.3.

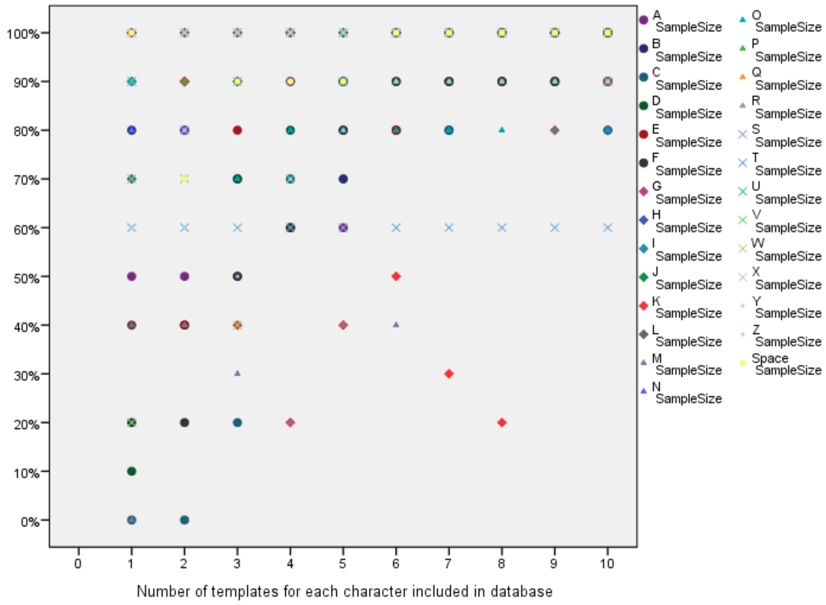

Figure 4.2 Variation in recognition accuracy with increasing number of templates using Database 1

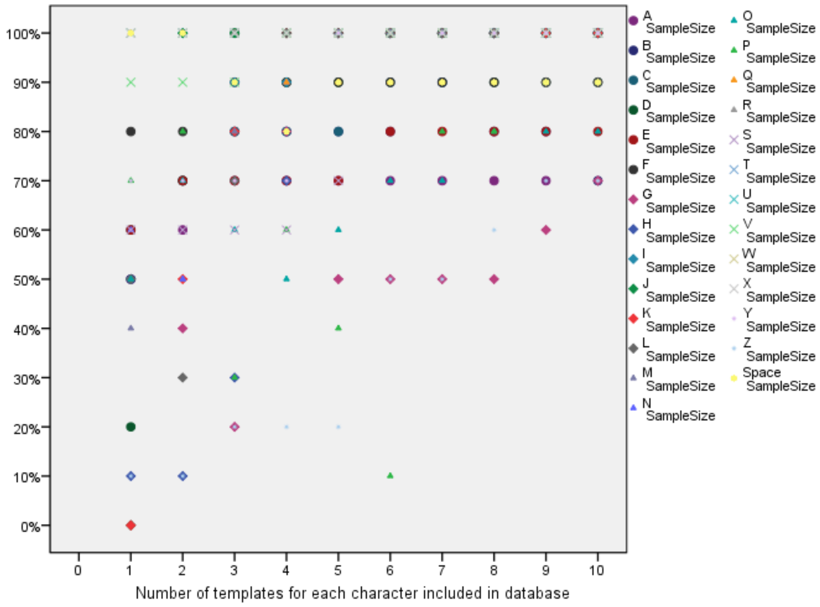

Figure 4.3 Variation in recognition accuracy with increasing number of templates using Database 2

Since the shapes of the points in the two figures above are similar to ellipses, the relationship between the recognition accuracies and the sample sizes for each Teeline shorthand character is linear, and the linear relation between them will be further confirmed and discussed in Chapter 5.

4.5 Experiment II

4.5.1 Objective

The aim of the second experiment is to test the recognition accuracy of the program. As previously mentioned, accuracy is the most significant property of a hand recognition program; it is the crucial factor in evaluating the utility, and usability of a product. The two databases applied in Experiment I will be used in this experiment as well. To test the recognition accuracies for all the characters in the database, a pangram is an ideal choice for this study. Experiment II will use the 35-letter pangram: “The quick brown fox jumps over the lazy dog”.

4.5.2 Hypotheses

In this second test, one sentence in Teeline shorthand is recognized using two databases. The databases were already introduced in section 4.4 Experiment I. Based on the fact that Database 1 and Database 2 have the same properties except for the sample source, the hypotheses are made as follows.

Hypothesis 1: The recognition accuracies have no significant difference between different sourced databases used in the program.

Hypothesis 2: It is consistent when comparing the program’s performance (overall recognition accuracies) for users in different groups

4.5.3 Methodology

For Experiment II, 30 participants were recruited. Each participant was asked to write the pangram in the Teeline shorthand alphabet once. The gestures drawn by each participant were recorded and recognized using Database 1 and Database 2 respectively. For this experiment, a laptop was placed on the table, with a keyboard and a Leap Motion Controller attached to it; all were placed in front of the participants. Participants were seated at the table and had the Leap Motion Controller placed directly in front of them. The Teeline shorthand alphabet was either placed on the table beside the laptop or attached on the wall for participants' convenience. The space between words is a separate character, and also need to be drawn as part of the Teeline alphabet; therefore, participators have a total of 43 Teeline shorthand gestures to draw to complete the pangram. The tasks that they performed are as follows:

1. Read through a consent form and sign it if s/he agrees to participate.
2. Fill out the questionnaire (Appendix 3).
3. Review the Teeline shorthand alphabet as in Figure 2.9 (the alphabet was reproduced in a large-sized font, and attached to the wall or placed on the participant's table as desires)
4. Practice drawing Teeline gestures using Leap Motion controller to familiarize themselves with the LM's sensing area and the program's running process
5. Write the whole pangram in Teeline shorthand alphabet, drawing one character at a time from the beginning to the end.

All 30 participants were randomly selected, and they came from sixteen different programs including Biomedical Biology, Computational Science, Economics, Health Promotion, Philosophy, Science Communication, and Mathematics, and others. There were 27 right-handed participants involved in this research, while the other three tended to use their left hand more frequently when carrying out tasks. All of the participators had no prior knowledge about shorthand, as well as no experience with the Leap Motion Controller. Other information collected from questionnaires for all participants were illustrated from Figure 4.4 to Figure 4.7.

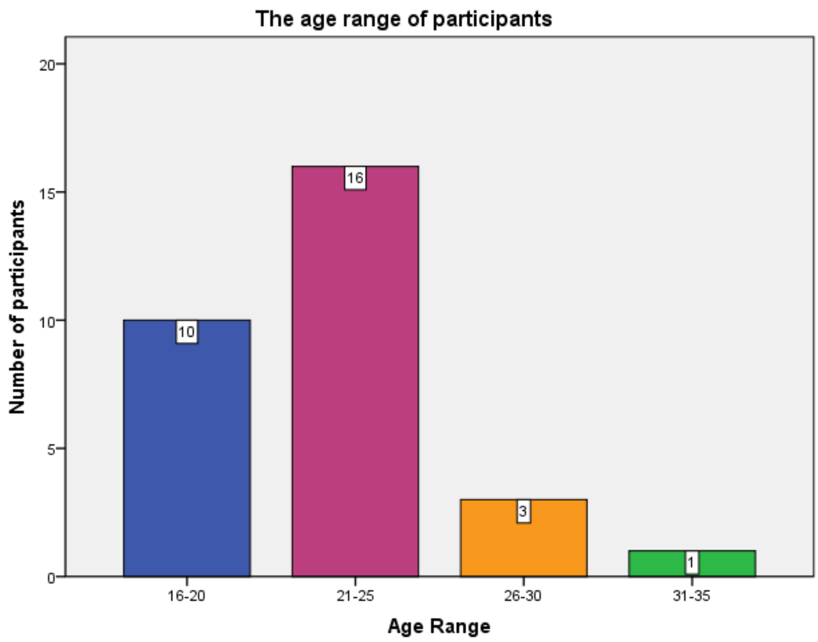

Figure 4.4 Age distribution of participants

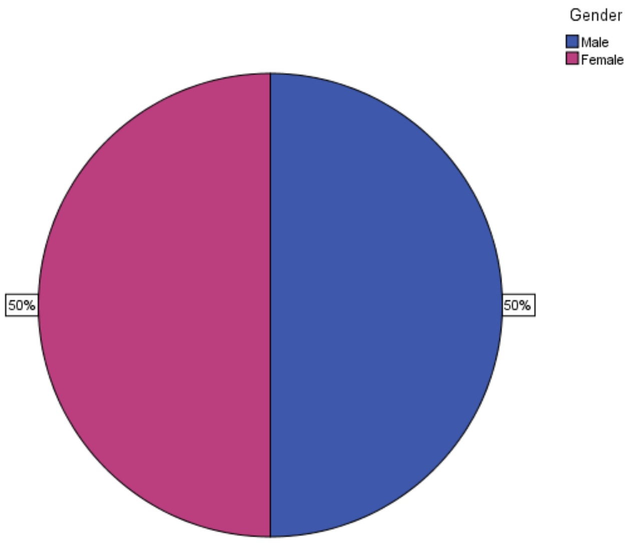

Figure 4.5 Gender of participants

Figure 4.6 Participants' Experience Playing Video Games

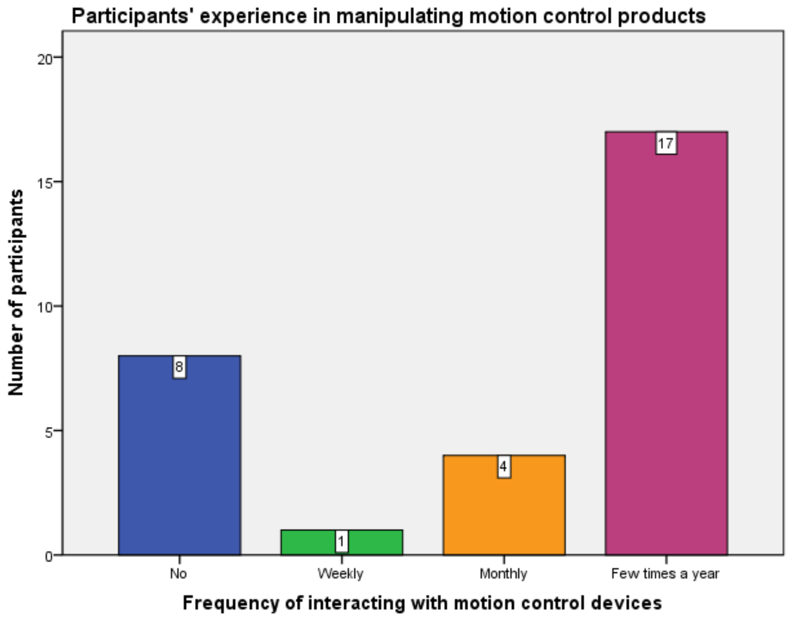

Figure 4.7 Participants' Experience Interacting with Motion Control Devices

It can be concluded from observing the above charts that there were an equal number of female and male participants. Ten participants were aged between 16 to 20, sixteen participants were aged between 21 to 25, three participants were aged between 26 and 30 and one participant is aged between 26 and 30. Out of the participants, twelve had never played video games, nine played video games for a maximum of one hour per day on average, six averagely spent one to three hours on video games per day, while three participants claimed to play video games for longer than three hours each day on average (two of them played three to five hours and one took over five hours to play each day on average). With regard to their experience with motion control devices, 22 out of the 30 participants had previously tried motion control devices (17 participants played them several times a year, 4 participants played on a monthly basis, and 1 participant played weekly); however, the other 8 participants had no prior experience with motion control products.

All characters from the participants were recorded and saved by the conductor, and those records were recognized using the two databases. The recognition results from Experiment II will be addressed in next subsection.

4.5.4 Results

After utilizing the two databases to recognize those test samples in Experiment II, the overall recognition accuracy (including the “Space” character) for each participant’s test samples is listed in Table 4.4.

Table 4.4 Overall Recognition Accuracy for Each Participant's Record

Participant's	Recognition Acc. Using	Recognition Acc. Using
1	83.72%	83.72%
2	58.14%	76.74%
3	90.70%	93.02%
4	72.09%	69.77%
5	67.44%	72.09%
6	72.09%	79.07%
7	72.09%	65.12%
8	90.70%	86.05%
9	83.72%	74.42%
10	81.40%	74.42%
11	69.77%	69.77%
12	86.05%	67.44%
13	97.67%	95.35%
14	86.05%	79.07%
15	93.02%	79.07%
16	86.05%	76.74%
17	95.35%	86.05%
18	93.02%	93.02%
19	88.37%	67.44%
20	95.35%	90.70%
21	90.70%	90.70%
22	90.70%	90.70%
23	76.74%	65.12%
24	72.09%	60.47%
25	79.07%	67.44%
26	83.72%	65.12%
27	90.70%	83.72%
28	97.67%	90.70%
29	83.72%	88.37%
30	90.70%	88.37%

The recognition accuracies in Table 4.4 describe the overall recognition accuracies for the pangram (including the character “Space”). The total number of characters in the pangram is 43; therefore, the accuracy is calculated by a formula:

$$accuracy = \frac{the\ number\ of\ correctlt\ recognized\ characters}{the\ total\ number\ of\ characters}$$
$$= \frac{the\ number\ of\ correctlt\ recognized\ characters}{43}$$

It is evident from observing the results that the recognition accuracies are not consistent across Database 1 and Database 2 for the 30 records collected from 30 different participants. Using Database 1, the recognition accuracy for the given pangram was as low as 58.14% and as high as 97.67%. However, using Database 2, the lowest recognition accuracy was 60.47% and the highest was 95.35%. It is difficult to determine from examining these numbers which database performs more efficiently with regards to the overall recognition accuracy. Therefore, a frequency analysis was carried out in SPSS in order to further evaluate the data.

Table 4.5 Frequency Analysis for Overall Recognition Accuracies

Statistics

		Overall recognition accuracies using Database 1	Overall recognition accuracies using Database 2
N	Valid	30	30
	Missing	0	0
Mean		83.9535%	78.992%
Std. Deviation		9.98592%	10.37309%
Skewness		-.748	-.074
Std. Error of Skewness		.427	.427

Overall recognition accuracies using Database 1

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	58.1395%	1	3.3	3.3	3.3
	67.4419%	1	3.3	3.3	6.7
	69.7674%	1	3.3	3.3	10.0
	72.0930%	4	13.3	13.3	23.3
	76.7442%	1	3.3	3.3	26.7
	79.0698%	1	3.3	3.3	30.0
	81.3953%	1	3.3	3.3	33.3
	83.7209%	4	13.3	13.3	46.7
	86.0465%	3	10.0	10.0	56.7
	88.3721%	1	3.3	3.3	60.0
	90.6977%	6	20.0	20.0	80.0
	93.0233%	2	6.7	6.7	86.7
	95.3488%	2	6.7	6.7	93.3
	97.6744%	2	6.7	6.7	100.0
	Total	30	100.0	100.0	

Overall recognition accuracies using Database 2

		Frequency	Percent	Valid Percent	Cumulative Percent
Valid	60.4651%	1	3.3	3.3	3.3

65.1163%	3	10.0	10.0	13.3
67.4419%	3	10.0	10.0	23.3
69.7674%	2	6.7	6.7	30.0
72.0930%	1	3.3	3.3	33.3
74.4186%	2	6.7	6.7	40.0
76.7442%	2	6.7	6.7	46.7
79.0689%	2	6.7	6.7	53.3
79.0698%	1	3.3	3.3	56.7
83.7209%	2	6.7	6.7	63.3
86.0465%	2	6.7	6.7	70.0
88.3721%	2	6.7	6.7	76.7
90.6977%	4	13.3	13.3	90.0
93.0233%	2	6.7	6.7	96.7
95.3488%	1	3.3	3.3	100.0
Total	30	100.0	100.0	

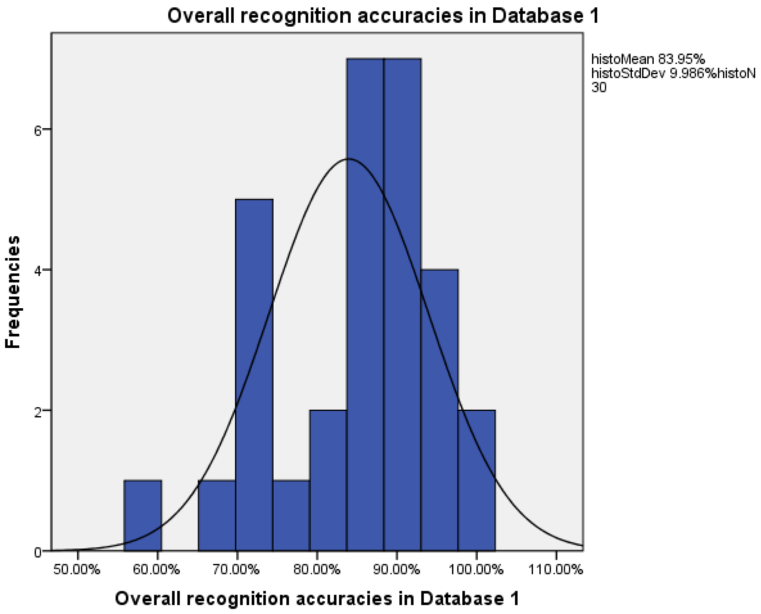

Figure 4.8 Histogram of Recognition Accuracies using Database 1

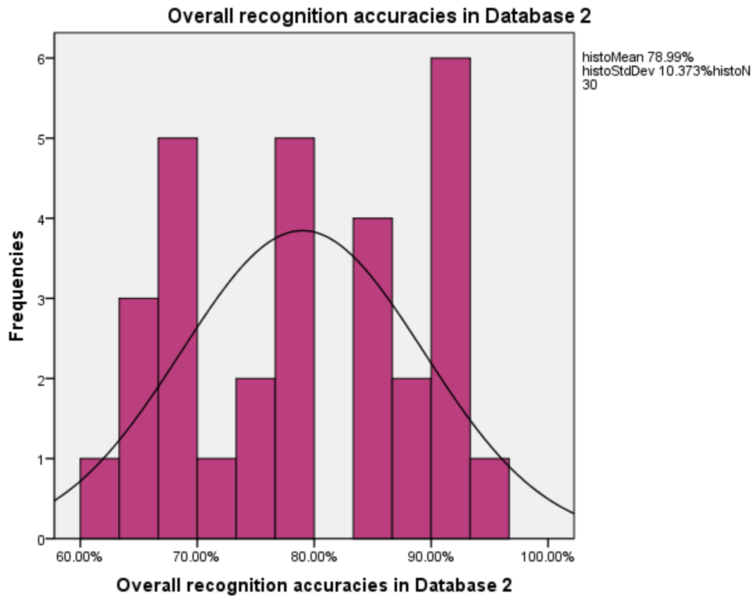

Figure 4.9 Histogram of Recognition Accuracies using Database 2

Table 4.5 displays the means of the overall pangram recognition accuracies using Database 1 and Database 2 for Experiment II. The mean of the recognition accuracies using Database 1 was 83.9535% while the standard deviation was 9.9859%. The Database 2 which was built by different participants, had a mean recognition accuracy of 78.95922% and a standard deviation of 10.3731%. There was indeed a difference between the means in the two databases. It is worth mentioning that the "skewness" of each database indicates the position where several recognition accuracies had the same values. Both the recognition accuracies for Database 1 and Database 2 contain negative skewness, suggesting that the majority of the recognition results tend to produce a higher level of recognition accuracy than mean while the recognition accuracies are claimed to be "negatively skewed". A positive skewness, although did not shown in this case, reveals that the majority of the recognition results tend to produce a low accuracy while the recognition accuracies are said to be “positively skewed”. The skewed distributions in the two databases can be observed in Figure 4.8 and Figure 4.9. The histogram of the distribution of recognition accuracies using Database 1 reveals a negatively skewed distribution. With regard to Database 2, it is important to note that the absolute value of the skewness is quite small even though it is negative as well. The histogram of the distribution of recognition accuracies using Database 2 also shows a skewed distribution. Further discussion on the overall recognition accuracies using the two databases will be undertaken in Chapter 5.

Chapter 5

Discussion

5.1 Discussion in Experiment I

It is important to note that if the recognition accuracy is related to the number of templates included in database, then a change in the sample size would tend to be accompanied by a change in the recognition accuracy. Since there are circumstances in which statistical measurement can be highly misleading [52], a scatter plot is always employed before finding the correlation coefficient. Referring to Figure 4.2 and Figure 4.3 in subsection 4.4.4 Results, it is obvious that there is a linear relationship between these two variables using the two databases, a Pearson coefficient would be suitable for calculating the overall strength of the relationship. The Pearson correlation coefficient is a commonly-used parameter for evaluating the strength and direction of the relationship between two variables that are linearly related to one another. The value of the Pearson correlation coefficient ranges from -1 to +1. A coefficient of -1 indicates a perfectly inverse relationship; a coefficient of +1 indicates a perfectly positive relationship; and a coefficient of 0 indicates that there is no linear relationship between the variables.

The results of bivariate correlate analysis that have been obtained from employing SPSS are presented in Table 1 and Table 2 in Appendix D. It is important to note that the type of measure of correlation that has been applied here is the Pearson product-moment correlation coefficient (r_{xy}), which is commonly used to describe the linear relationship between two quantitative variables [53]. The correlation coefficient of two variables X and Y is $r_{xy} = \frac{Cov_{xy}}{SD_x SD_y}$ [53].

Where: $Cov_{xy} = \frac{\sum(X-\bar{X})(Y-\bar{Y})}{N-1}$ or $\frac{\sum xy}{N-1}$

$$x = (X - \bar{X})\ and\ y = (Y - \bar{Y})$$

$$N = number\ of\ pairs\ of\ measurements$$

$$SD_x = standard\ deviation\ of\ the\ first\ variable\ (X)$$

$$SD_y = standard\ deviation\ of\ the\ second\ variable\ (Y)$$

After observing the results (see Table 1 in Appendix D), it is evident that with regard to the majority of the Teeline characters, there is a strong positive relationship between the recognition accuracies and the sample size using Database 1. The correlation coefficient could be as high as 0.906, which is statistically significant at the 0.01 level. Considering the Teeline characters “K”, “P” and “Y”, the relationship between the recognition accuracies and the sample sizes is inverted using Database 1. However, the absolute values of the correlation coefficients are 0.13, 0.29 and 0.078 respectively, which are not significant. In addition, no linear relationship can be found between the two variables for five of the Teeline characters, “Q”, “R”, “T”, “W” and “X”. After examining the data in Table 5.2, it is evident that the recognition accuracies for these characters are constant and do not change in relation to sample size. The Pearson correlation coefficient between the recognition accuracy and the sample size of “U” is 0, thereby suggesting that the two variables could be related in a curvilinear manner.

There is an inverse relationship between the recognition accuracy and the sample size for the Teeline characters “Space” using Database 2 (see Table 2 in Appendix D). The correlation coefficient is -0.420. Although the absolute value of the correlation coefficient for the character “Space” is larger than those for characters “K”, “P” and “Y”, it is evident from observing Table 4.3 that the recognition accuracies are significantly higher for this character in contrast to others. The Pearson correlation coefficient between the recognition accuracy and the sample size of “Q” is 0, which suggests that the two variables could be related in a curvilinear relationship. In addition, the recognition accuracies are maintained at 100% for the four characters “R”, “T”, “U”, “W” and “X”, regardless of sample size. However, with respect to the other 20 Teeline shorthand characters, the recognition accuracies are indeed proportional to the sample size.

To conclude, for six Teeline characters “Q”, “R”, “T”, “U”, “W” and “X”, there is no linear relationships found between the recognition accuracies and the sample size in both databases. Based on the results in Table 1 and Table 2, the recognition accuracies for these Teeline characters are either constant or less volatile, which are not affected by the sample size in database. Therefore, the relationships between the recognition accuracies and the sample size in database for Teeline characters “Q”, “R”, “T”, “U”, “W” and “X” are not effective to reject the Hypothesis 1 in subsection 4.4.2 Hypotheses. Even though some inverse relationships are found, the Pearson correlation coefficients are not statistical significant to prove that the recognition accuracy is inversely proportional to the sample size for each Teeline character using the two databases. However, there are positive relationships between the recognition accuracies and the sample size in database for the rest eighteen and twenty Teeline characters in two tables; it is evident that in general, the changes that occur in recognition accuracy are directly related to changes in sample size in the two databases. The hypothesis 1 in subsection 4.4.2 Hypotheses is true for both databases.

Please refer to subsection 4.4.4 Results for the various recognition accuracies associated with changes in sample size in two of the databases (the results for Database 1 are shown in Table 4.2, while the results for Database 2 are presented in Table 4.3). It is apparent that for every character in each of the two databases, there exists a value for the sample size that results in the recognition accuracies remaining unchanged. In other words, the recognition accuracy for a Teeline character can reach its maximum value when a specific sample size is reached. In this case, the specific sample size is referred to as the optimal sample size for this Teeline shorthand character. The optimal sample size for recognizing each character of the Teeline alphabet using Database 1 and Database 2 are summarized in Table 5.1 and Table 5.2.

Table 5.1 Optimal Sample Size for Each Character using Database 1

A	B	C	D	E	F	G	H	I
6	8	10	8	10	6	3	10	4
J	K	L	M	N	O	P	Q	R
4	10	6	7	7	4	4	1	1
S	T	U	V	W	X	Y	Z	Space
4	1	10	6	1	1	3	6	6

Table 5.2 Optimal Sample Size for Each Character using Database 2

A	B	C	D	E	F	G	H	I
10	10	8	8	6	3	9	5	5
J	K	L	M	N	O	P	Q	R
2	4	4	5	5	8	9	8	1
S	T	U	V	W	X	Y	Z	Space
6	1	1	4	1	1	7	9	5

According to the tables above, each Teeline character has its own optimal sample size to achieve the highest recognition accuracies. Hypothesis 2 (an optimal sample size of the sample size for each Teeline character exists so that the recognition accuracy remains constant) has therefore been proven to be correct in the cases of both databases.

Based on the previous discussion, there is an optimal sample size for each character in the two databases. However, an overall optimal sample size is required in order to verify Hypothesis 3. A representative for all the optimal sample sizes for all Teeline characters is an efficient way for verifying whether or not the optimal sample sizes of two databases are significantly different. SPSS was applied to analyze the frequencies of the optimal sample sizes in Table 5.1 and Table 5.2, as shown above. Database 1 and Database 2 are regarded as two variables in this analysis, and the central tendencies are chosen as the Mean, Median and Mode. The output is presented in Table 5.3.

Table 5.3 Frequency of Optimal Sample Size

Statistics

		Database1	Database2
N	Valid	27	27
	Missing	27	27
Mean		5.44	5.37
Median		6.00	5.00
Mode		6	1[a]

a. Multiple modes exist. The smallest value is shown

It is important to refer back to what was previously mentioned in last chapter, in that the sources of the templates in the two databases are different; for example, Database 1 was built by an experienced user while Database 2 was built by novices. After observing Table 5.3, it can be concluded that one of the central tendencies using Database 1 and Database 2 can be represented as being the optimal sample size for all characters associated with the two databases. With regard to Database 1, the mean value of the optimal sample sizes was 5.44, the median was 6 and the mode was also 6. Therefore, the representative for all the optimal sample sizes for all Teeline characters in Database 1 is 6. In addition, with regard to Database 2, the mean value of the optimal sample sizes is 5.37, the median is 5 and the mode is 1 and 5. Even if there are multiple modes existed, the mean and median can be taken as the representative. Therefore, the optimal sample size for all of the Teeline characters presented in the Database 2 is 5.

It is evident that the optimal sample sizes for the two databases differ from one another. In order to determine if the means of these two optimal sample sizes differ to a statistically significant degree, a T test was carried out in SPSS. Since the test sample for Database 1 and Database 2 is the same one, the Paired-Samples T test was selected.

The Paired-Samples T test consists of a set of mathematical procedures that yields a numerical value, which is referred to as t_{obt} in this particular case. The larger the absolute value of t_{obt}, the more likely it is to reflect a statistically significant difference between the two groups compared [53]. The formula for the T test with regard to the dependent (matched) samples can be utilized with samples of equal and unequal sizes:

$$t_{obt} = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{S_{\overline{X_1}}^2 + S_{\overline{X_2}}^2 - 2r_{12}S_{\overline{X_1}}S_{\overline{X_2}}}}$$

$Where$: $\overline{X_1}$, $\overline{X_2}$ are the means of the two measurements

$S_{\overline{X_1}}S_{\overline{X_2}}$ are the standard errors of the means $\left(\frac{SD}{\sqrt{N}}\right)$

r_{12} is the correlation between the two measurements

Determining which sample mean is subtracted from the other is entirely subjective. In this case, the direction of the difference is unimportant, the T test is nondirectional and the absolute value of t_{obt} was employed.

The result reveals the relationship between the two paired variables (see Table 3 in Appendix D). The Pearson correlation coefficient is 0.29 and $p = 0.251$. Since $p > 0.05$, the two variables are not significantly related. The df is defined as "degrees of freedom" which in this particular case is 26 (see Table 3 in Appendix D). The Paired-Samples T test revealed that the mean difference between the two paired variables is 0.074, $t(26) = 1.102$ and $p = 0.919$. The 95% confidence interval on the difference was $[-1.414, 1.562]$, which includes the value of zero. Since $p > 0.05$, it can be concluded that there are no significant differences between the means of the two related samples (the means of the optimal sample sizes using Database 1 and Database 2). Therefore, a database built by an experienced user of using motion control devices and Teeline shorthand had a similar mean value of the optimal sample size as a database built by novices to motion control devices and shorthand. Hypothesis 3 has been proven to be correct.

5.2 Discussion in Experiment II

Even if the means of the recognition accuracies in the two databases are objectively different based on Table 4 (see Appendix D), they cannot be used to decide whether the differences are statistically significant. A paired-samples T test was conducted to compare the two recognition accuracy means in this case.

The means of the recognition accuracies using Database 1 and Database 2 are 83.9535% and 78.9922%, respectively (see Table 4 in Appendix D). The standard deviations in the two databases are 9.9859% and 10.3731%, respectively. The table of Paired Samples Correlations shows that the Pearson correlation coefficient between the means of the two databases is 0.668 ($p = 0.000$). Due to the fact that $p < 0.05$, the two means are significantly statistically related. A Paired Samples T test proves that the mean difference between the two paired variables is 4.9613% and the standard deviation of paired differences is 8.3008%. In addition, the results indicate a significant relationship beyond the 0.05 level: $t(29) = 3.274$ and $p = 0.003$ (2-tailed) which is smaller than 0.05. The 95% confidence interval on the difference is $[1.8617\%,\ 8.0609\%]$, which does not include the value of zero. Therefore, it can be concluded that Hypothesis 1 in subsection 4.5.2 is false. According to the above statistical results, the means of the recognition accuracies contain significant differences between Database 1 and Database 2, which were built by an experienced user and novices. Specifically, the overall recognition accuracy for this specific pangram is higher when using a database built by an experienced user with two aspects including using motion control devices and Teeline shorthand rather than novices.

It was revealed in the previous section that the program had better performance when

Database 1 was utilized in Experiment II. Moreover, other information collected from the questionnaires filled by the participants in Experiment II were already illustrated in subsection 4.5.3 Methodology. Discussion of further findings based on the classified user groups will be clarified in this part, and Hypothesis 2 in subsection 4.5.2 will be analyzed here as well.

1. Differences in program's performance based on users' age ranges

In the previous discussion, the Paired-Samples T test was applied in SPSS in order to compare the two means of the optimal sample sizes in two databases. However, after observing Figure 4.4, the independent variables, which are related to this case, have to be divided into more than two groups. The T test is not suited here. The analysis of variance (ANOVA), which is used for testing when the independent variable contains more than two groups, will be employed here instead of the T test.

The objective of ANOVA is to test for statistical significance of the differences between the means of two or more groups [53]. The test determines the variance between groups, and compares it with the variance within groups. The most important step in carrying out an ANOVA is to compute the variance of the total number of subjects in the study: $s_T^2 = \frac{\sum(X-\overline{X_T})^2}{N_T-1}$. $\sum(X-\overline{X_T})^2$ is called the "total sum of squares" and is represented by SS_T, since it's calculated across the total values of each subject, regardless of the group in which the subject is. N_T is the total number of subjects in all groups. In detail, the SS_T can be broken down into two parts: $SS_T = SS_W + SS_B$. In this equation, SS_W is the sum of squares within groups, which shows the degree of variability within groups; SS_B is the sum of square between groups, which reflects differences between groups [53]. The total degrees of freedom is equal to $N_T - 1$, and can be divided to the degrees of freedom within all the groups and the number of groups minus 1. The SS_W and SS_B are calculated by the following formulas in an ANOVA:

$$SS_W = \sum\sum X^2 - \sum\frac{(\sum X)^2}{N}$$

$$SS_B = \sum\frac{(\sum X)^2}{N} - \frac{(\sum X_T)^2}{N_T}$$

$where\ N$ is the number of participants in each group

$\sum X_T$ is the total value for all groups

Dividing SS_W by the degrees of freedom within all groups gives a measure of the variability within groups, called the mean square within, represented by MS_W. Dividing SS_B by the number of groups minus 1 gives a measure of the variability between groups, called the mean square between, represented by MS_B. When comparing if the between-group differences are significantly greater than they would be by chance, the ratio of a mean square between groups to a mean square within groups is given by $F_{obt} = \frac{MS_B}{MS_W}$. The *obt* subscript means that it will be compared with a critical value to test how likely it is that the event represented by F_{obt} could have happened by chance.

In order to discover whether or not there are significant differences between the recognition accuracies obtained by the participants in each of the age groups, an ANOVA was conducted on each of the two databases. Since there is only one independent variable in each test, the One-Way ANOVA seems to be appropriate. In this case, the participants' age range was selected as the independent variable.

The ANOVA tables (see Table 5 in Appendix D) show the results of the overall analysis of variance, including between groups, within groups, as well as the total sum of squares, degrees of freedom and mean squares. The F-ratios for the analysis using the two databases are 0.337 and 0.793, respectively, with the probabilities of 0.799 and 0.509 using Database 1 and Database 2. Both probabilities exceed the requirement of a probability to be less than 0.05 in order to be statistically significant; therefore, the participants who were in different age ranges obtained similar mean accuracies. Therefore, the program has consistently performance for users in different age groups.

2. Differences in program's performance based on users' gender

The pie chart in Figure 4.5 displays that there were equal numbers of male and female participants in Experiment II (15 participants of each gender). In this section, a T test was applied to examine whether or not there are significant differences between recognition accuracies of males' and females' test samples within the two databases. Since the overall recognition accuracies of males' and females' test samples are two pairs of independent variables, the Independent-Samples T test in SPSS had been selected.

It is important to note that the Independent-Sample T test in this case is different from the previous Paired-Sample T test. The t-value of the Independent-Samples T test is calculated using the following formula [53]:

$$t_{obt} = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\left[\frac{SD_1^2(n_1 - 1) + SD_2^2(n_2 - 1)}{n_1 + n_2 - 2}\right]\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

$where$: t_{obt} is the t-value calculated based on the data

$\overline{X_1}, \overline{X_2}$ are means in the two groups of data

n_1, n_2 are the number of participants in two groups

SD_1^2, SD_2^2 are variances in the two groups of data

When the direction of the difference is unimportant such as the cases in this paper, the Independent-Samples T test is nondirectional and the absolute value of t_{obt} is used.

Depending on whether or not the two groups of data have similar variances for the dependent variables, there are two different methods for computing the t-value in SPSS [54]. With regard to the Independent-Samples T test in SPSS, it adopts Levene's

Test for Equality of Variances in order to determine if the two groups of the independent variable have about the same or different amounts of variability between values. The Levene's Tests yield probabilities of 0.131 and 0.277, respectively (see Appendix D: Table 6 and Table 7). Since the two possibilities are greater than 0.05, the results suggested that the difference between the variances of the recognition accuracies of males' and females' test samples using Database 1 and Database 2 are not significant. Therefore, the Independent-Samples T test in SPSS computed the t-value based on the assumption that the variances of the recognition accuracies of male's and females' test samples using two databases are equal, and the results corresponding to the row "Equal variances assumed" in Independent Samples Test tables are valid in this case. In Table 6 in Appendix D, the result is a $t(28) = -0.89$ and has a probability of $p = 0.381$ (two-tailed). As this is greater than 0.05, it can be concluded that the program has similar overall recognition accuracies for both males' and females' test samples when employing Database 1. In Table 7 in Appendix D, the result is a $t(28) = -0.525$ with a probability of $p = 0.603$ (two-tailed). As this is also greater than 0.05, it can be concluded that there is no significant difference between the overall recognition accuracies for the test samples of males and females when employing Database 2. To summarize, there are no significant differences in statistics in the program's performance when it is used by users with different gender.

3. Differences in program's performance based on users' handedness

It was stated in subsection 4.4.3 Methodology that all of the participants who built the templates for Database 2 were right-handed. This therefore raises the question of whether the program has same performance for right-handed users and left-handed users or not. To answer such a question, the differences between the means of overall recognition accuracies for the two groups (right-handed participants and left-handed participants) requires further investigation.

In total, 30 participants were involved in Experiment II. According to the questionnaire results, three of the participants claimed to use their left hands more frequently while the others used their right hands more often. Since the overall recognition accuracies for the two groups are independent from each other, along with the fact that the Independent-Samples T test can be applied when two groups have different sample sizes, it was selected to analyze the above question on the two databases separately.

As the results shown (see Table 8 and Table 9 in Appendix D), the Levene's Tests for Equality of Variances yield probabilities of 0.365 and 0.209, respectively; due to the fact that they are both greater than 0.05, they suggest that the variances of the recognition accuracies of two groups' test samples are not significantly different from one another using the two databases while the Independent-Samples T test should be applied based on the assumption of equal variances. In Table 8, the result is a $t(28) = 0.461$ and has a probability of $p = 0.648$ (two-tailed). As this is greater than 0.05, it can be concluded that the program has similar overall recognition accuracies for both left-handed and right-handed users when employing Database 1. In Table 9, the result is a $t(28) = 0.390$ with a probability of $p = 0.700$ (two-tailed). As this is also greater than 0.05, it can be concluded that the program has similar overall recognition accuracies for left-handed and right-handed users when employing Database 2. To summarize, there are no statistically significant differences in the program's performance when it is used by left-handed and right-handed users.

In addition, there are no significant differences found between the recognition accuracies for users with different experience with video games and motion control devices using two databases (Appendix E). Therefore, the program's performance is consistent when used by users in different groups (groups in different age ranges, gender, handedness, experience with video games and motion control devices). Hypothesis 2 in subsection 4.5.2 Hypotheses was proven to be correct.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Hand gesture recognition is a novel technique in Human-Computer Interaction (HCI). It has become an important aspect of HCI since gesture recognition is an efficient method to carry out control features without the usage of a keyboard, and will be a new trend in the future of user interfaces [55]. The Leap Motion (LM) controller is part of a new generation of motion control products, which provides a new method for users to interact with the computers. By using motion control sensors like LM, the interactive program can collect motion data in a fast, easy and accurate way, which provides a novel goal and direction in the development of interactive software and the research in pattern recognition in the coming decades. This paper applied a kind of abbreviated symbolic writing called Teeline shorthand, utilized the hand tracking function of the Leap Motion controller, and developed a hand gesture recognition program used for interpreting users' 3D Teeline gestures into English words.

The program in this project applied a template matching method called Dynamic Time Warping (DTW) to implement the recognition capability. There were two modes built in this program: Recognition mode (RM) and Edit mode (EM). The RM is the main function performed by the program, in which users drew Teeline shorthand gestures using their fingers or hands, and those gestures were recognized as English letters, words and sentences by the program. The EM resulted from the idea of building a flexible application where users are allowed to create their own gesture commands in this mode. It was designed so that end-users have access to database and enlarge their gesture vocabulary.

In order to test the program's performance, specifically the recognition accuracy, a series of experiments were conducted using two different databases. One database was built by an experienced user of using motion control devices and Teeline shorthand, and the other database was built by novices. All the other properties of the two databases were the same. The experiment results were analyzed in SPSS using different means, such as T test and ANOVA, and the analysis revealed the following findings:

- The recognition accuracy of the program has a direct relationship with the sample size in the database to some extent, and there are optimal sample sizes for each Teeline characters in two databases at which further increases in sample size doesn't leas to big increases in recognition accuracy.
- The program showed better performance when using Database 1 than using

Database 2; therefore, a database built by experienced users would be more appropriate for the program to achieve high recognition accuracy.

- The program's recognition accuracy is uniform for users in different age ranges, gender, handedness and experience with video games and motion control devices.

To summarize, the hand gesture recognition program based on the DTW algorithm in this paper shows consistent performance almost at all times, in cases of using different databases and facing various user communities. It can be successfully applied in interpreting Teeline shorthand gestures into English language.

6.2 Future work

Although the recognition program in this paper reaches the basic requirements of the project, there are still some aspects of the program that need to be improved. The primary improvements will focus on the following aspects:

- Improving the recognition algorithm to pursue better accuracy

  The recognition algorithm in this paper is based on DTW since it is an appropriate means in the current situation. However, in a more complex situation in the future (i.e. recognizing complex gestures rather than Teeline), DTW may not be sufficient to recognize gestures. Combining DTW with other mainstream algorithms, such as HMM, would be a better method for gesture recognition.

- Adding more gestures to extend the program functions

  As can be seen from this paper, the program cannot be completely independent from keyboard at this moment. The more important objective pursued in this project is the recognition accuracy of the program; some auxiliary functionalities were done by using a keyboard instead of gesture inputs in order to reduce the factors which have effects on accuracy. In the future, the author will work on adding specific gestures to auxiliary functionalities under the requirement of a high total accuracy and improving the program to one that is not reliant on a keyboard.

- Updating the means used for collecting motion data with new released Leap Motion SDK in the future

  Even though the Leap Motion Controller has launched in the past few years as a new technology, the company is always concentrating on better ways to track movements of hands and fingers. The Leap Motion's newer motion tracking technology used in this program, called Leap Motion V2, allows the device to track subjects that are not directly seen by its sensor, which is a defect in the original version, called Leap Motion V1 [56]. With different updates of Leap Motion coming out in the future, there will hopefully be a more accurate way to track motion data, and achieve better performance.

The development of technology keeps changing people's lives. If motion controlled interfaces become the new trend of Human-Computer Interaction, there is a possibility that the mature version of the program discussed in this paper will enter people's daily lives and become an essential way of communications between human and computers.

Bibliography

1. Myers, B.A., *A brief history of human-computer interaction technology.* interactions, 1998. **5**(2): p. 44-54.
2. Sutherland, I.E., *Sketch pad a man-machine graphical communication system*, in *Proceedings of the SHARE design automation workshop*. 1964, ACM. p. 6.329-6.346.
3. English, W.K., D.C. Engelbart, and M.L. Berman, *Display-Selection Techniques for Text Manipulation.* IEEE Transactions on Human Factors in Electronics, 1967. **HFE-8**(1): p. 5-15.
4. Marchand, A. and T. Hennig-Thurau, *Value creation in the video game industry: Industry economics, consumer benefits, and research opportunities.* Journal of Interactive Marketing, 2013. **27**(3): p. 141-157.
5. *Xbox One*. [cited 2016; Available from: https://en.wikipedia.org/wiki/Xbox_One.
6. Metz, R. *Look Before You Leap Motion*. July 22, 2013; Available from: https://www.technologyreview.com/s/517331/look-before-you-leap-motion/.
7. Srilatha, P. and T. Saranya, *Advancements in Gesture Recognition Technology.* IOSR Journal of VLSI and Signal Processing (IOSR-JVSP), 2014. **4**(4): p. 1-7.
8. Oka, K., Y. Sato, and H. Koike, *Real-time fingertip tracking and gesture recognition.* IEEE Computer Graphics and Applications, 2002. **22**(6): p. 64-71.
9. Carmona, J.M. and J. Climent, *A Performance Evaluation of HMM and DTW for Gesture Recognition*, in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 17th Iberoamerican Congress, CIARP 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings*, L. Alvarez, et al., Editors. 2012, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 236-243.
10. Zhichao, Y., et al. *Finger-writing-in-the-air system using Kinect sensor*. in *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. 2013.
11. Qu, C., Tian, J., Wang, S., & Xu, W., *KinWrite: Handwriting-Based Authentication Using Kinect.* 2013.
12. Jambusaria, U., Katwala, N., Kadam, M., & Narula, H., *Finger Writing in Air using Kinect.* Utsav Jambusaria et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, 2014. **5**(6): p. 8119-8121.
13. Rajesh Dilip Savatekar , A.A.D., *Implementation of Articulated Robot for Object Dimension Measurement.* International Journal of Current Trends in

Engineering & Research (IJCTER), 26/04/2016. **2**(4): p. 112-126.
14. *Leap Motion developer API Overview*. Available from: https://developer.leapmotion.com/documentation/v2/cpp/devguide/Leap_Overview.html#sensor-images.
15. SHIVANGI S NAYAK, V.H.N., *Designing a Gesture Based Device to Recognize Sign Language Using Leap Motion Controller.* International Journal of Innovative Research in Computer

and Communication Engineering, 2016. **4**(3): p. 3837-3842.
16. Weichert, F., et al., *Analysis of the accuracy and robustness of the leap motion controller.* Sensors (Basel), 2013. **13**(5): p. 6380-93.
17. *System Architecture — Leap Motion C++ SDK v2.3 documentation*. Available from: https://developer.leapmotion.com/documentation/v2/cpp/devguide/Leap_Architecture.html.
18. Ager, S. *Shorthand*. Available from: http://www.omniglot.com/writing/shorthand.htm.
19. Books, L. and S. Wikipedia, *Shorthand Systems: Shorthand, Pitman Shorthand, Gregg Shorthand, Bezenaek Shorthand, Tironian Notes, Stiefografie, Dutton Speedwords*. 2010: General Books LLC.
20. *Shorthand*. Available from: http://www.statemaster.com/encyclopedia/Shorthand.
21. *Shorthand*. [cited 2016; Available from: https://www.tititudorancea.net/z/shorthand.htm.
22. Hill, J., *Teeline : a method of fast writing*. 1968, London: Heinemann Educational.
23. Quintero, D., et al., *Workload Optimized Systems: Tuning POWER7 for Analytics*. 2013: IBM Redbooks.
24. Du Li, K.Y., *Research and development on food nutrition statistical analysis software system.* Advance Journal of Food Science and Technology, 2013. **5**(12): p. 1637-1640.
25. *Benefit of SPSS*. 2012 [cited 2016; Available from: http://benefitof.net/benefits-of-spss/.
26. Prof Kamal K Vyas, A.P., Dr. Sandhya Vyas, *Gesture Recognition and Control Part 1 - Basics, Literature Review & Different Techniques.* 2013. **1**(7): p. 575-581.
27. Müller, M., *Information retrieval for music and motion*. Vol. 2. 2007: Springer.
28. Rabiner, L. and B.-H. Juang, *Fundamentals of speech recognition*. 1993: Prentice-Hall, Inc. 507.
29. Sakoe, H. and S. Chiba, *Dynamic programming algorithm optimization for spoken word recognition.* IEEE transactions on acoustics, speech, and signal

processing, 1978. **26**(1): p. 43-49.
30. Mitoma, H., S. Uchida, and H. Sakoe. *Online character recognition using eigen-deformations*. in *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*. 2004.
31. Alon, J., V. Athitsos, and S. Sclaroff. *Online and offline character recognition using alignment to prototypes*. in *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*. 2005. IEEE.
32. Bahlmann, C. and H. Burkhardt, *The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004. **26**(3): p. 299-310.
33. Rath, T.M. and R. Manmatha. *Word image matching using dynamic time warping*. in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. 2003. IEEE.
34. Deza, M.M. and E. Deza, *Encyclopedia of distances*, in *Encyclopedia of Distances*. 2009, Springer. p. 1-583.
35. Qiao, Y. and M. Yasuhara, *Affine Invariant Dynamic Time Warping and its Application to Online Rotated Handwriting Recognition*, in *Proceedings of the 18th International Conference on Pattern Recognition - Volume 02*. 2006, IEEE Computer Society. p. 905-908.
36. zouxy09. *语音信号处理之(一)动态时间规整(DTW)*. 2013 [cited 2016; Available from: http://blog.csdn.net/zouxy09/article/details/9140207.
37. Manjunatha, V. and A. Subramanian, *Hand Gesture Recognition from a Single Training Sample.*
38. Bret. *App Review: Short Hand by LizzardWerks*. 2009 [cited 2016; Available from: http://www.appchatter.com/2009/06/app-review-short-hand-by-lizzardwerks/.
39. Nielsen, J., *Usability Engineering*. 1993: Morgan Kaufmann Publishers Inc. 358.
40. Swain, L., *Basic principles of questionnaire design.* Survey Methodology, 1985. **11**(2): p. 161-170.
41. Crawford, I.M., *Marketing research and information systems*. 1997: Food & Agriculture Org.
42. International, B.B. *The key principles of effective questionnaire design*. 2006 [cited 2016; Available from: https://www.b2binternational.com/b2b-blog/2006/05/12/the-key-principles-of-effective-questionnaire-design/.
43. Garcia-Ruiz, M.A., *Cases on usability engineering : design and development of digital products.* 2013: p. 65.

44. Gardner, D. *Five helpful principles for questionnaire design*. 2012 [cited 2016; Available from: https://www.visioncritical.com/five-helpful-principles-questionnaire-design/.
45. Anderson, C.A., et al., *Violent video game effects on aggression, empathy, and prosocial behavior in eastern and western countries: a meta-analytic review.* Psychological bulletin, 2010. **136**(2): p. 151.
46. Ferguson, C.J., *Violent video games and the Supreme Court: lessons for the scientific community in the wake of Brown v. Entertainment Merchants Association.* American Psychologist, 2013. **68**(2): p. 57.
47. Lemola, S., et al., *Habitual computer game playing at night is related to depressive symptoms.* Personality and Individual Differences, 2011. **51**(2): p. 117-122.
48. Granic, I., A. Lobel, and R.C. Engels, *The benefits of playing video games.* American Psychologist, 2014. **69**(1): p. 66.
49. Gerling, K., et al., *Ageing Playfully: Advancing Research on Games for Older Adults Beyond Accessibility and Health Benefits*, in *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. 2015, ACM: London, United Kingdom. p. 817-820.
50. Hulley, S.B., et al., *Designing clinical research*. 2013: Lippincott Williams & Wilkins.
51. SCHADE, A. *Pilot Testing: Getting It Right (Before) the First Time*. April 5, 2015 [cited 2016; Available from: https://www.nngroup.com/articles/pilot-testing/.
52. Gray, C.D. and P.R. Kinnear, *IBM SPSS statistics 19 made simple*. 2012: Psychology Press.
53. Kranzler, G., J. Moursund, and J.H. Kranzler, *Statistics for the Terrified*. 2006: Prentice-Hall, Inc.
54. Einspruch, E.L., *An Introductory Guide to SPSS? for Windows?* 2005: Sage.
55. Premaratne, P., Q. Nguyen, and M. Premaratne. *Human computer interaction using hand gestures*. in *International Conference on Intelligent Computing*. 2010. Springer.
56. Buckley, S. *Leap Motion's latest motion tracking tech can seee your joints*. 2014 [cited 2016; Available from: https://www.engadget.com/2014/05/28/leap-motions-beta-tracking-tech-can-see-your-joints/.

Appendix

Appendix A Program source code

LeapMotion_CAS.cpp

```
// LeapMotion_CAS.cpp: Define the entry of the program for the console
//

#include "stdafx.h"
#include "MyHeader.h"
#include "MyListener.h"
#include "MyKNN.h"
#include <time.h>

static bool isFileExist(string file_name)
{
	string folder = "Database\\";
	string path = folder.append(file_name);
	fstream stream(path);
	if(!stream)
	{
		return false;
	}
	else
	{
		return true;
	}
}


static void FindAllFile(string path,vector<string> &files)
{
	_finddata_t c_file;
	intptr_t hFile;
	hFile=_findfirst(path.append("\\*.txt").c_str(), &c_file);
	if( hFile	== -1)
		return;
```

```
    do
    {
        if(c_file.attrib&_A_SUBDIR)
        {
            //Ignore it if it is a subfolder
        }
        else
        {
            files.push_back(c_file.name);
        }
    }
    while( _findnext( hFile, &c_file ) == 0);
    _findclose(hFile);
}

bool model;



int _tmain(int argc, _TCHAR* argv[])
{
    MyListener m_listener;
    Controller m_controller;
    MyKNN knn;
    SetWindowPos(GetConsoleWindow(), HWND_TOPMOST, 0, 0, 770, 500,
SWP_SHOWWINDOW);
    vector<Mat> records;
    vector<string> result;
    vector<string> filesPathVector;



    // Get a standard I/O handle
    HANDLE hOut = GetStdHandle(STD_OUTPUT_HANDLE);
    HANDLE hIn = GetStdHandle(STD_INPUT_HANDLE);

    DWORD               dwRes, dwState=0;
    INPUT_RECORD         keyRec;

    char c ;
```

```
        bool is_reset;
        is_reset = true;


#pragma region Main_Console
        Main_Console:
            {
                if(filesPathVector.size()==0)
                {
                        FindAllFile("Database",filesPathVector);
                        if(filesPathVector.size() != 0)
                        {
                                for(int i=0;i<filesPathVector.size();i++)
                                {
                                        string path_name="Database\\";
                                        string lable;
                                        for(int j=1;j<filesPathVector[i].length();j++)
                                        {
                                                if( filesPathVector[i].substr(j,1)>="0"&
filesPathVector[i].substr(j,1)<="9")
                                                {
                                                        lable = filesPathVector[i].substr(0,j);
                                                        if(lable=="&")
                                                        {
                                                                lable=" ";
                                                        }
                                                }
                                        }

knn.AddSampleFromFile(path_name.append(filesPathVector[i]),lable);
                                }
                                cout<<endl<<filesPathVector.size()<<" samples trained!"<<endl<<endl;
                        }// end if
                }

                if(is_reset)
                {cout<<"Do you want to change to Edit Mode? Y/N    "<<std::endl;}
            while(1)
            {
                    is_reset = false;
```

```
ReadConsoleInput(hIn, &keyRec, 1, &dwRes);
if (keyRec.EventType == KEY_EVENT)
{
    if(keyRec.Event.KeyEvent.bKeyDown)
    {
        c = keyRec.Event.KeyEvent.uChar.AsciiChar;
        switch(c)
        {
            case    'y'|'Y':
            {
                model = true;
                cout<<endl<<"Please press ENTER key to start and then place one hand above the Leap Motion"<<endl;
                break;
            }
            case     'n'|'N':
            {
                model = false;
                cout<<endl<<"Please press ENTER key to start and then place one hand above the Leap Motion"<<endl;
                break;
            }
            case     13://ASCII code for Enter key
            {
                m_controller.addListener(m_listener);

                cout<<endl<<"Please only extend your INDEX FINGER and then press SPACE to record"<<endl;
                break;
            }
            case    ' ':
            {
                m_listener.start();
                cout<<endl<<"Please press ESC to stop"<<endl;
                break;
            }
            case    27://ESC key
            {
                m_controller.removeListener(m_listener);
                if(model)
```

```
                goto Edit;
            else
                goto Record;
        }
        case    9://TAB key
        {
            if(records.size() >0)
            {
                for(int i=0;i<records.size();i++)
                {
                        string f="AllRecords\\";
                        char number[100];
                        sprintf(number,"%02d",i);
                        string recordsname="file.txt";

recordsname=recordsname.substr(0,4)+number+recordsname.substr(4,4);
                        string path=f.append(recordsname);
                        ofstream outrecords(path,ios::app);
                        for (int m=0;m<records[i].rows;m++)
                        {
                            for (int n=0;n<2;n++)
                            {

outrecords<<records[i].at<double>(m,n)<<' ';
                            }
                            outrecords<<endl;
                        }
                        outrecords.close();
                }


                goto Recognise;
            }
            break;
        }
        case 52://4
        {
            if(model)
            {
                goto Save;
```

```
                        }
                    }
                    case 53://5
                    {
                        cout<<endl<<"Please input the letter you just recorded (e.g.
A)"<<endl;
                        string console_lable;
                        cin>>console_lable;
                        Mat mat_coordinate =
knn.CoordinateToMatrix(m_listener.vector_fingerlocation);
                        knn.InitMatrix(mat_coordinate);
                        knn.Train(mat_coordinate, console_lable);
                        cout<<endl<<"Please press 4 to save this record"<<endl;
                        break;
                    }
                    case 49://1
                    {
                        Mat tmp
=knn.CoordinateToMatrix(m_listener.vector_fingerlocation);
                        knn.InitMatrix(tmp);
                        records.push_back(tmp);
                        cout<<endl<<"Record successfully saved!"<<endl;
                        cout<<endl<<"Please press ENTER key to start new record, or
press TAB key to recognize"<<endl;
                        is_reset = false;
                    }

                }//end switch
            }

        }

    }// end Model
  }
#pragma endregion

#pragma region Edit
Edit:
  {
      if(model)
```

```
		{
			cout<<endl<<"The size of this record is
"<<m_listener.vector_fingerlocation.size()<<endl;
			if(m_listener.vector_fingerlocation.size() >0)
			{
				cout<<endl<<"Please press 4 to save this record, or press 5 to train this record,
or press ENTER key to start new record"<<endl;
				{cout<<"Do you want to change to Edit Mode? Y/N    "<<std::endl;}

			}//end if(m_listener.vector_fingerlocation.size() >0)
		}// end if(model)
		goto Main_Console;
	}//end Edit
#pragma endregion

#pragma region Save
Save:
	{
		cout<<endl<<"Please name this record as the following format: Letter.txt"<<endl;
		cout<<endl<<"e.g. I just recorded an 'A',and the file name should be 'A.txt' "<<endl;
		bool isSaveCompleted = false;
		while(!isSaveCompleted)
		{
			string file_name;
			cin>>file_name;
			if(isFileExist(file_name))
			{
				cout<<endl<<"Sorry, the file name already exists, please input a different
one"<<endl;
			}
			else
			{
				m_listener.SaveCoordination(file_name);
				isSaveCompleted = true;
				cout<<endl<<endl;

			}
		}//end while
		cout<<"Do you want to stay in Edit Mode? Y/N    "<<endl;
		if (c=='y'|'Y')
```

```
		{
			c=13;
			goto Main_Console;
		}
		if(c=='n'|'N')
		{
			is_reset=true;
			goto Main_Console;
		}//end if
	}
#pragma endregion

#pragma region Record
Record:
	{
		if(!model)
		{
			cout<<endl<<"The size of this record is
"<<m_listener.vector_fingerlocation.size()<<endl;
			if(m_listener.vector_fingerlocation.size() >0)
			{
				cout<<endl<<"Please press 1 to keep this record or press ENTER key to start
new record"<<endl;
			}

			goto Main_Console;
		}
	}

#pragma endregion

#pragma region Recognise
Recognise:
	{
		int s=0;
		char c2,c3;
		time_t start,stop;
		if(!model)
		{
			cout<<endl<<"The total number of records is    "<<records.size()<<endl;
```

```
		if(records.size() >0)
		{
			cout<<endl<<"Under recognizing....."<<endl;
			start = time(NULL);
			for(int i=0;i<records.size();i++)
			{
				s++;
				string result_tmp = knn.FindNearst(records[i],5);
				result.push_back(result_tmp);

			}

	cout<<"********************************************************************
************"<<endl;
			cout<<endl<<"The recognition result is:"<<endl<<endl;
			for(int j=0;j<records.size();j++)
			{

				if (j==0)
				{
					string t=result[j];
					if (t[0]>=97&t[0]<=122)	//Transfer it to capital if the first letter is
in lower case
					{
						t[0]-=32;
					}
					result[j]=t;

				}//end if
				cout<<result[j];
			}
			cout<<endl<<endl;

	cout<<"********************************************************************
************"<<endl;
			stop = time(NULL);
			printf("Use Time: %ldseconds\n\n\n",(stop-start));//Display the time used for
recognizing
		}//end if(records.size() >0)
```

```
                cout<<"Do you want to compare using another database? Y/N"<<std::endl;
                c2=getchar();
                c3=getchar();
                if(c2=='y'|c2=='Y')
                {
                    if (s==records.size())
                    {
                        knn.ClearTrainedSample(s);
                    }
                    result.clear();
                    filesPathVector.clear();
                    goto Compare;
                }
                else if(c2=='n'|c2=='N')
                {
                    result.clear();
                    records.clear();
                    is_reset=true;
                    goto Main_Console;
                }
            }//end if(!model)

    }


#pragma endregion
#pragma region Compare
Compare:
    {
        int s=0;
        time_t start,stop;
        if(!model)
        {
        FindAllFile("Zrecord",filesPathVector);
        if(filesPathVector.size() != 0)
        {
            for(int i=0;i<filesPathVector.size();i++)
            {
                string path_name="Zrecord\\";
```

```
                        string lable;
                        for(int j=1;j<filesPathVector[i].length();j++)
                        {
                              if( filesPathVector[i].substr(j,1)>="0"&
filesPathVector[i].substr(j,1)<="9")
                              {
                                    lable = filesPathVector[i].substr(0,j);
                                    if(lable=="&")
                                    {
                                          lable=" ";
                                    }
                              }
                        }
                        knn.AddSampleFromFile(path_name.append(filesPathVector[i]),lable);
                  }
            }// end if
            cout<<endl<<"Under recognizing....."<<endl;
            start = time(NULL);
            for(int i=0;i<records.size();i++)
            {
                  s++;
                  string result_tmp = knn.FindNearst(records[i],5);
                  result.push_back(result_tmp);
            }

      cout<<"============================================================
===================="<<endl;
            cout<<endl<<"The recognition result is:"<<endl<<endl;
            for(int j=0;j<records.size();j++)
            {
                  if (j==0)
                  {
                        string t=result[j];
                        if (t[0]>=97&t[0]<=122)
                        {
                              t[0]-=32;
                        }
                        result[j]=t;

                  }//end if
```

```
                cout<<result[j];
            }
            cout<<endl<<endl;

        cout<<"============================================================
===================="<<endl;
            stop = time(NULL);
            printf("Use Time: %ldseconds\n\n\n",(stop-start));
            }

            if(s==records.size())
            {
                knn.ClearTrainedSample(s);
            }

            filesPathVector.clear();
            result.clear();
            records.clear();
            is_reset=true;
            goto Main_Console;
        }
#pragma endregion


return 0;
}
```

MyKNN.cpp

```
#include "StdAfx.h"
#include "MyKNN.h"


MyKNN::MyKNN(void)
{
}


MyKNN::~MyKNN(void)
{
}


Mat MyKNN::CoordinateToMatrix(vector<FingerLocation>    vector_fingerlocation)
{
    // Converts the set of input coordinates to a matrix
    Mat input_mat(0,2,DataType<double>::type);//Result matrix used to return
    Mat mat(1,2,DataType<double>::type);    //Temporary matrix used to save one pair of input
coordinates
    for(int i=0;i< vector_fingerlocation.size();i++)
    {
        mat.at<double>(0,0)=vector_fingerlocation[i].x;
        mat.at<double>(0,1)=vector_fingerlocation[i].y;
        input_mat.push_back(mat);
    }
    mat.release();//Empty the temporary matrix
    return input_mat;
}

void MyKNN::Train(Mat sample,string lable)
{
    //Set a label to each matrix and add it to the trained sample set
    TrainedSample temp;
    sample.copyTo(temp.TrainedSample_Mat);
    temp.TrainedSample_lable=lable;
    vector_trained_sample.push_back(temp);
}
```

```
string MyKNN::FindNearst(Mat sample,int K)
{
	//Input a matrix and the value for K, then output a string as label
	Mat sample_dist(0,2,DataType<double>::type);// Result matrix used to return after
recognizing, the first element in each row is the number of the template used for comparing, and
the second elements in each row is the result of DTW algorithm
	Mat temp(1,2,DataType<double>::type);	//Temporary matrix
	for(int i=0;i<vector_trained_sample.size();i++)
	{
		temp.at<double>(0,0)=i;
		double dist=dtw_OK(vector_trained_sample[i].TrainedSample_Mat,sample);
		temp.at<double>(0,1)=dist;
		sample_dist.push_back(temp);
	}

	if(K>sample_dist.rows)
		cout<<"K比À¨¨样¨´本À?集¡¥合?的Ì?个?数ºy还1大ä¨®，ê?请?修T改?K的Ì?值
¦Ì"<<endl;

	BubleSort(sample_dist,K);//Bubble sort for K times

	temp.release();

	vector<VoteVector> vector_lable; //Save the voting result
	vector_lable.clear();			//Empty the voting result before begin a new vote
	if(vector_lable.size() != 0)
	{
		cout<<"vector_lable	is	not empty"<<endl<<endl;
	}
	else
	{
		for(int j=0;j<K;j++)
		{
			bool isFinded=false; //A flag used to identify a label is already existed in the set of
voting result
			int ID=sample_dist.at<double>(j,0);
			for(int count=0;count<vector_lable.size();count++)
			{
				if(vector_lable[count].lable ==
vector_trained_sample[ID].TrainedSample_lable)
```

```
                    {
                        vector_lable[count].vote++;
                        isFinded = true;//Set flag to true is there is the same label in the voting
result
                    }
                }//end for int count
                if(!isFinded)
                {           //Add the label to the voting result if there is no same label in the result
                    VoteVector tempVote;
                    tempVote.lable=vector_trained_sample[ID].TrainedSample_lable;
                    tempVote.vote=1;
                    vector_lable.push_back(tempVote);
                }
            }//end for int j
        }

        //Voting process is ended, return the vote label which has the maximum number of votes
        int result=0;    //Return the first label in case of any exceptions
        for(int i=1;i<vector_lable.size();i++)
        {
            if(vector_lable[i].vote>vector_lable[result].vote)
                result=i;
        }

        return vector_lable[result].lable;
}


void MyKNN::InitMatrix(Mat mat)
{
        //Initial the matrix, subtract each column by the minimum value of this column
        double max_value1,max_value2;
        double min_value1,min_value2;
        cv::minMaxIdx(mat.col(0),&min_value1, &max_value1);
        cv::minMaxIdx(mat.col(1),&min_value2, &max_value2);
        Mat min_mat1(mat.rows,1,DataType<double>::type,min_value1);
        cv::subtract(mat.col(0),min_mat1,mat.col(0));
        Mat min_mat2(mat.rows,1,DataType<double>::type,min_value2);
        cv::subtract(mat.col(1),min_mat2,mat.col(1));
        min_mat1.release();
```

```
	min_mat2.release();

}


void MyKNN::AddSampleFromFile(string file_name,string lable)
{
	// Read coordinates in the files and train them
	Mat mat=ReadMatrixFromFile(file_name);

	InitMatrix(mat);

	Train(mat, lable);
}

Mat MyKNN::ReadMatrixFromFile(string filename)
{
	//Convert the coordinates in a file to a matrix
	Mat mat(0,2, DataType<double>::type);
	Mat mat_temp(1,2, DataType<double>::type);
	ifstream stream_in;	//Input file
	stream_in.open(filename,ios::in);
	if(stream_in.is_open())
	{
		while(!stream_in.eof())
		{
			double x,y;
			stream_in>>x;
			stream_in>>y;
			if(x != NULL	&& y != NULL	)
			{
				mat_temp.at<double>(0,0)=x;
				mat_temp.at<double>(0,1)=y;
				mat.push_back(mat_temp);
			}
		}
	}
	else
		cout<<endl<<endl<<"文?件t不?存ä?在¨²，ê?请?检¨¬查¨¦"<<endl<<endl;
	stream_in.close();
```

```
    mat_temp.release();
    return mat;
}

void MyKNN::BubleSort(Mat mat,int K)
{
    //Bubble sort for K times
    if(K > mat.rows)
        return;
    Mat temp_row(1,2,DataType<double>::type);
    for(int i=0;i<K;i++)
    {

        mat.row(i).copyTo(temp_row);//Record the start position of each Bubble sort
        for(int ptr=i;ptr < mat.rows;ptr++)
        {
            if(mat.at<double>(ptr,1) < temp_row.at<double>(0,1))
            {
                mat.row(ptr).copyTo(temp_row);
                mat.row(i).copyTo(mat.row(ptr));
                temp_row.copyTo(mat.row(i));
            }
        }
    }
    temp_row.release();
}

double MyKNN::dtw_OK(Mat A,Mat B)
{
    //Compute the similarity of two matrixes using Euclidean distance
    Mat d(A.rows,B.rows,DataType<double>::type);//Save the Euclidean distance of each pair of
coordinates
    for(int i=0;i<d.rows;i++)
    {
        for(int j=0;j<d.cols;j++)
        {

    *d.ptr<double>(i,j)=dist(*A.ptr<double>(i,0),*A.ptr<double>(i,1),*B.ptr<double>(j,0),*B.pt
r<double>(j,1));
        }
```

```
    }

    Mat D=Mat::zeros(d.rows,d.cols,DataType<double>::type);//Save the DTW distance between each pair of points
    *D.ptr<double>(0,0)=*d.ptr<double>(0,0);

    for(int i=1;i<D.rows;i++)
    {
        *D.ptr<double>(i,0)=*d.ptr<double>(i,0)+*D.ptr<double>(i-1,0);
    }
    for(int j=1;j<D.cols;j++)
    {
        *D.ptr<double>(0,j)=*d.ptr<double>(0,j)+*D.ptr<double>(0,j-1);
    }

    for(int m=1;m<D.rows;m++)
    {
        for(int n=1;n<D.cols;n++)
        {
            double temp_min = *D.ptr<double>(m-1,n-1);
            if(*D.ptr<double>(m,n-1)< temp_min)
            {
                temp_min=*(D.ptr<double>(m,n-1));
            }
            if(*D.ptr<double>(m-1,n) < temp_min)
            {
                temp_min =*D.ptr<double>(m-1,n);
            }
            *D.ptr<double>(m,n) = *d.ptr<double>(m,n)+temp_min;
        }
    }
    double Dist=D.at<double>(D.rows-1,D.cols-1);
    return Dist;//Dist is used to represent the similarity of two samples, the smaller the value, the more similar the samples
}


double MyKNN::dist(double x1,double y1,double x2,double y2)
{
    //Compute the Euclidean distance between two coordinates
```

```
    return sqrt(pow(x1-x2,2) + pow(y1-y2,2));
}


void MyKNN::ClearTrainedSample(int s)
{
    vector_trained_sample.clear();
}
```

MyListener.cpp

```
#include "StdAfx.h"
#include "MyListener.h"


MyListener::MyListener(void)
{
	img=Mat(400,600,CV_8UC1,cv::Scalar(255));//Set the size of the image to 400 by 600
}


MyListener::~MyListener(void)
{
	Listener::~Listener();
}

void MyListener::start()
{
	flag=true;
}

void MyListener::onInit(const Controller& controller)
{
	this->vector_fingerlocation.clear();//Clear the Display Window
	img.setTo(cv::Scalar(255));		//Set the Display Window to white color
	flag=false;
}

void MyListener::onFrame(const Controller& controller)
{
	const Frame frame = controller.frame();
	FingerList fingers = frame.fingers();

	if(!fingers.isEmpty())
	{
		Finger finger;
		bool isFind_extendedFinger=false;
		if(!isFind_extendedFinger)
		{for(int i=0;i<fingers.count();i++)
		{
```

```
            if(fingers[i].isExtended() & fingers[i].type() == Finger::TYPE_INDEX)
            {//Tracking the movements of the fingertip of Index finger
                finger=fingers[i];
                i=fingers.count()+4;//End the loop
                isFind_extendedFinger=true;
            }

        }
        }
        if(isFind_extendedFinger)
        {
            FingerLocation finger_location;
            finger_location.x = finger.tipPosition().x;
            finger_location.y = finger.tipPosition().y;
            int i=0;
            int j=0;
            i=finger.tipPosition().x+300;//Modify the value of x to fit the X-axis of Display
Window
            j=finger.tipPosition().y;
            if(j >= 400)
                j=399;
            if(i >=600)
                i=599;
            cv::circle(img,Point(i,400-j-1),2.5,cv::Scalar(0,0,0),2,8,0);

            cv::namedWindow("LeapMotion");
            cv::moveWindow("LeapMotion",800,50);
            imshow("LeapMotion",img);

            waitKey(1);
            if(flag == true)
            {
                vector_fingerlocation.push_back(finger_location);
            }else
            {
                img.setTo(cv::Scalar(255));
                cv::line(img,Point(0,200),Point(599,200),cv::Scalar(0,0,0),2,8,0);

            }
        }
```

```
        else
        {
            cout<<endl<<"There is no extended finger."<<endl;
        }

    }
    else
    {
        cout<<endl<<"No finger found."<<endl;
    }

}

void MyListener::SaveCoordination(string fileName)
{
    // Save each coordinates to a file
    string folder = "Database\\";
    string path = folder.append(fileName);
    ofstream stream(path,ios::app);
    for(int i=0;i<vector_fingerlocation.size();i++)
    {
        stream<<vector_fingerlocation[i].x<<"      "
              <<vector_fingerlocation[i].y<<"\n";
    }
}
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// LeapMotion_CAS.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in STADFA.H
// and not in this file
```

VoteVector.cpp

```
#include "stdafx.h"
#include "MyHeader.h"

VoteVector::VoteVector()
{
	vote=0; // Default value of vote is zero
}
```

MyHeader.h

```
#pragma once
#include "stdafx.h"
#include <opencv2\core\core.hpp>
#include <opencv.hpp>
#include <fstream>
#include "Leap.h"
#include <io.h>
#include <Windows.h>
#include <WinUser.h>
#include <math.h>

using namespace Leap;
using namespace cv;
using namespace std;

//Trained sample structure
struct TrainedSample
{
	cv::Mat   TrainedSample_Mat;    //Coordinate matrix for each template
	string    TrainedSample_lable; //Label for each template
};

//Test sample structure
typedef struct FingerLocation
{
	float   x;
	float    y;
};

//Vote result structure
typedef struct VoteVector
{
	VoteVector(); //Constructor, the default value of vote is set to 0 in VoteVectoe.cpp
	string   lable;//Label
	int     vote;    //Votes
};
```

MyKNN.h

```
#pragma once
#include "MyHeader.h"
class MyKNN
{
public:
	MyKNN(void);
	~MyKNN(void);
	vector<TrainedSample> vector_trained_sample; //Store the set of the trained sample

	Mat CoordinateToMatrix(vector<FingerLocation>   vector_fingerlocation); //Convert a set of coordinates to a matrix
	void Train(Mat sample,string lable); //Store Initial matrix and its label into vector_trained_sample
	string FindNearst(Mat sample,int K); //Find similar template and return its label


	void InitMatrix(Mat mat); //Initial matrix

	void AddSampleFromFile(string filename,string lable); //Read coordinates from files and add them to the set of samples
	Mat   ReadMatrixFromFile(string filename);//Read coordinates from files

	void BubleSort(Mat mat,int K);//Using bubble sort for two matrixes for K times

	double dtw_OK(Mat A,Mat B);    //Compute the similarity of two matrixes
	double dist(double x1,double y1,double x2,double y2);//Calculate the Euclidean distance of a pair of coordinates
	void ClearTrainedSample(int s);
};
```

MyListener.h

```
#pragma once
#include "MyHeader.h"

class MyListener :  public Listener
{
public:
	MyListener(void);
	~MyListener(void);

	bool flag;//Flag to identify whether start to record or not
	void start();//Start to record
	Mat img;    // Display the movements on Display Window
	vector<FingerLocation>    vector_fingerlocation;//Save the coordinates of each record

	void SaveCoordination(string fileName); // Save coordinates to a file
	virtual void onInit(const Controller&);
	virtual void onFrame(const Controller&);

};
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently,
// but are changed infrequently

#pragma once

#include "targetver.h"

#include <stdio.h>
#include <tchar.h>



// TODO: reference additional headers your program requires here
```

targetver.h

```
#pragma once

// Including SDKDDKVer.h defines the highest available Windows platform.


// If you wish to build your application for a previous Windows platform, include WinSDKVer.h and

// set the _WIN32_WINNT macro to the platform you wish to support before including SDKDDKVer.h.

#include <SDKDDKVer.h>
```

Appendix B Approval letter of research application

LaurentianUniversity
UniversitéLaurentienne

APPROVAL FOR CONDUCTING RESEARCH INVOLVING HUMAN SUBJECTS
Research Ethics Board – Laurentian University

This letter confirms that the research project identified below has successfully passed the ethics review by the Laurentian University Research Ethics Board (REB). Your ethics approval date, other milestone dates, and any special conditions for your project are indicated below.

TYPE OF APPROVAL / New X / Modifications to project / Time extension	
Name of Principal Investigator and school/department	Weikai Zang, M.Sc. Candidate, Mathematics and Computer Science, supervisor, Ratvinder Singh Grewal
Title of Project	Testing the accuracy of the application of Teeline Shorthand on Leap Motion Controller
REB file number	2016-06-14
Date of original approval of project	Sept. 8th, 2016
Date of approval of project modifications or extension *(if applicable)*	
Final/Interim report due on: *(You may request an extension)*	Sept. 8th, 2017
Conditions placed on project	

During the course of your research, no deviations from, or changes to, the protocol, recruitment or consent forms may be initiated without prior written approval from the REB. If you wish to modify your research project, please refer to the Research Ethics website to complete the appropriate REB form.

All projects must submit a report to REB at least once per year. If involvement with human participants continues for longer than one year (e.g. you have not completed the objectives of the study and have not yet terminated contact with the participants, except for feedback of final results to participants), you must request an extension using the appropriate LU REB form. In all cases, please ensure that your research complies with Tri-Council Policy Statement (TCPS). Also please quote your REB file number on all future correspondence with the REB office.

Congratulations and best wishes in conducting your research.

Rosanna Langer

Rosanna Langer, PHD, Chair, *Laurentian University Research Ethics Board*

Appendix C Questionnaire template in experiments

LaurentianUniversity
UniversitéLaurentienne

Department of Mathematics and Computer Science
Département de mathématiques et d'informatique
Tel/Tél.: 705-675-1151, 2310
Fax/Téléc.: 705-673-6591

Questionnaire

With this questionnaire, I would like to get to know some background information about you. The information will be helpful for my analyses after you complete the experiment. This questionnaire consists of 13 questions. None of them will involve your privacy or be used to identify you. All the information that I collect from this questionnaire will be kept securely in a password protected USB flash drive, and will be digitally shredded once the research is finished. It is important that you answer these questions truthfully. If you don't understand a certain question, please do not hesitate to ask me.

Thank you very much for your cooperation!

- - For administrative use only --

Participant's ID： _______________ Date: _________________

---

1. Can you read in English? ☐ Yes ☐ No

2. Can you understand English? ☐ Yes ☐ No

3. What is your program?

   ________________________________ (☐ undergraduate/☐ graduate)

4. What is your age range?

   ☐ 16-20 ☐ 21-25 ☐ 26-30 ☐ 31-35

   ☐ 36-40 ☐ 41-45 ☐ 46-50 ☐ 51+

5. What is your gender:

   ☐ Male ☐ Female ☐ Prefer not to disclose

6. What is your mother tongue?

   ☐ English ☐ French ☐ Arabic ☐ Hindi ☐ Japanese

   ☐ Mandarin/Cantonese ☐ Korean ☐ Other, please specify:___________

7. Which hand do you use more frequently in writing?

   ☐ Right hand ☐ Left hand

8. Have you used shorthand before? ☐ No ☐ Yes

   If "Yes", please specify which form you have used:

   ☐ Pitman ☐ Gregg ☐ Teeline ☐ Other:____________

9. Did you know about Teeline shorthand before participating this experiment?

   ☐ No ☐ Yes

   If "Yes", how much did you know about Teeline shorthand?

   ☐ A little bit ("I heard of it before")

☐ Very well ("I can read and write Teeline shorthand")

10. Do you play video games?

☐ Yes ☐ No

If "Yes", how many hours do you play a day on average?

☐ Less than 1 hour ☐ 1 hr.-3 hr. ☐ 3 hr.-5 hr. ☐ More than 5 hours

11. Are you familiar with any touchscreen gesture-based user interface (e.g. screen pattern lock, zooming in or out using gestures)?

☐ Yes ☐ No

12. Have you tried any motion control products (e.g. Nintendo Wii, Kinect)?

☐ Yes ☐No

If "Yes", how often do you interact with your motion control product?

☐ Daily ☐ Weekly ☐ Monthly ☐ Few times a year

13. Have you heard about Leap Motion controller?

☐ No ☐ Yes

If "Yes", do you have experience in interacting with Leap Motion controller?

☐ Not at all ☐Only one or two times ☐ Several times ☐ Many times

This is the end of this questionnaire.

Thank you again for your cooperation!

Appendix D Statistical analysis results

Table 1 Correlations Between Recognition Accuracy and Sample Size using Database 1

Correlations

		SampleSize	A	B	C	D	E	F
SampleSize	Pearson Correlation	1	.906**	.609	.837**	.867**	.822**	.899**
	Sig. (2-tailed)		.000	.062	.003	.001	.004	.000
	N	10	10	10	10	10	10	10
		SampleSize	G	H	I	J	K	L
SampleSize	Pearson Correlation	1	.696*	.111	.609	.736*	-.130	.058
	Sig. (2-tailed)		.025	.759	.062	.015	.720	.873
	N	10	10	10	10	10	10	10
		SampleSize	M	N	O	P	Q	R
SampleSize	Pearson Correlation	1	.901**	.930**	.696*	-.290	.[c]	.[c]
	Sig. (2-tailed)		.000	.000	.025	.416	.	.
	N	10	10	10	10	10	10	10
		SampleSize	S	T	U	V	W	X
SampleSize	Pearson Correlation	1	.785**	.[c]	.000	.846**	.[c]	.[c]
	Sig. (2-tailed)		.007	.	1.000	.002	.	.
	N	10	10	10	10	10	10	10
		SampleSize	Y	Z	Space			
SampleSize	Pearson Correlation	1	-.078	.878**	.570			
	Sig. (2-tailed)		.831	.001	.086			
	N	10	10	10	10			

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

c. Cannot be computed because at least one of the variables is constant.

Table 2 Correlations between Recognition Accuracy and Sample Size using Database 2

Correlations

		SampleSize	A	B	C	D	E	F
SampleSize	Pearson Correlation	1	.765**	.478	.962**	.578	.892**	.696*
	Sig. (2-tailed)		.000	.062	.006	.158	.905	.031
	N	10	10	10	10	10	10	10

		SampleSize	G	H	I	J	K	L
SampleSize	Pearson Correlation	1	.659*	.870**	.621	.522	.743*	.684*
	Sig. (2-tailed)		.028	.034	.401	.416	.137	.002
	N	10	10	10	10	10	10	10

		SampleSize	M	N	O	P	Q	R
SampleSize	Pearson Correlation	1	.824**	.720**	.807**	.354	.000	.c
	Sig. (2-tailed)		.090	.007	.052	.469	.631	.
	N	10	10	10	10	10	10	10

		SampleSize	S	T	U	V	W	X
SampleSize	Pearson Correlation	1	.897**	.c	.c	.798**	.c	.c
	Sig. (2-tailed)		.009	.	.122	.086	.	.
	N	10	10	10	10	10	10	10

		SampleSize	Y	Z	Space
SampleSize	Pearson Correlation	1	.872**	.962**	-.420
	Sig. (2-tailed)		.001	.000	.554
	N	10	10	10	10

**. Correlation is significant at the 0.01 level (2-tailed).

*. Correlation is significant at the 0.05 level (2-tailed).

c. Cannot be computed because at least one of the variables is constant.

Table 3 Paired-Samples T test for Optimal Sample Size

Paired Samples Correlations

	N	Correlation	Sig.
Pair 1 Database1 & Database2	27	.229	.251

Paired Samples Test

	Paired Differences			
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference
				Lower
Pair 1 Database1 - Database2	.074	3.761	.724	-1.414

Paired Samples Test

	Paired Differences	t	df	Sig. (2-tailed)
	95% Confidence Interval of the Difference			
	Upper			
Pair 1 Database1 - Database2	1.562	0.102	26	.919

Table 4 Paired-Samples T test for Overall Recognition Accuracies in Two Databases

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	Overall recognition accuracies using Database 1	83.9535%	30	9.98592%	1.82317%
	Overall recognition accuracies using Database 2	78.9922%	30	10.37309%	1.89386%

Paired Samples Correlations

		N	Correlation	Sig.
Pair 1	Overall recognition accuracies using Database 1 & Overall recognition accuracies using Database 2	30	.668	.000

Paired Samples Test

		Paired Differences			
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference
					Lower
Pair 1	Overall recognition accuracies using Database 1 - Overall recognition accuracies using Database 2	4.96129%	8.30084%	1.51552%	1.86170%

Paired Samples Test

		Paired Differences	t	df	Sig. (2-tailed)
		95% Confidence Interval of the Difference			
		Upper			
Pair 1	Overall recognition accuracies using Database 1 - Overall recognition accuracies using Database 2	8.06087%	3.274	29	.003

Table 5 One-Way ANOVA for Different Age Groups Users using Database 1 and Database 2

ANOVA

Overall recognition accuracies using Database 1

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	108.145	3	36.048	.337	.799
Within Groups	2783.693	26	107.065		
Total	2891.838	29			

ANOVA

Overall recognition accuracies using Database 2

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	261.568	3	87.189	.793	.509
Within Groups	2858.859	26	109.956		
Total	3120.426	29			

Table 6 Independent-Sample T test for males' and females' test samples using Database 1

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means
		F	Sig.	t
Overall recognition accuracies using Database 1	Equal variances assumed	2.419	.131	-.890
	Equal variances not assumed			-.890

Independent Samples Test

		t-test for Equality of Means		
		df	Sig. (2-tailed)	Mean Difference
Overall recognition accuracies using Database 1	Equal variances assumed	28	.381	-3.25582%
	Equal variances not assumed	24.788	.382	-3.25582%

Independent Samples Test

		t-test for Equality of Means	
		Std. Error Difference	95% Confidence Interval of the Difference
			Lower
Overall recognition accuracies using Database 1	Equal variances assumed	3.65952%	-10.75200%
	Equal variances not assumed	3.65952%	-10.79602%

Independent Samples Test

		t-test for Equality of Means
		95% Confidence Interval of the Difference
		Upper
Overall recognition accuracies using Database 1	Equal variances assumed	4.24036%
	Equal variances not assumed	4.28438%

Table 7 Independent-Sample T test for males' and females' test samples using Database 2

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means
		F	Sig.	t
Overall recognition accuracies using Database 2	Equal variances assumed	1.229	.277	-.525
	Equal variances not assumed			-.525

Independent Samples Test

		t-test for Equality of Means		
		df	Sig. (2-tailed)	Mean Difference
Overall recognition accuracies using Database 2	Equal variances assumed	28	.603	-2.01539%
	Equal variances not assumed	27.693	.603	-2.01539%

Independent Samples Test

		t-test for Equality of Means	
		Std. Error Difference	95% Confidence Interval of the Difference
			Lower
Overall recognition accuracies using Database 2	Equal variances assumed	3.83590%	-9.87287%
	Equal variances not assumed	3.83590%	-9.87680%

Independent Samples Test

		t-test for Equality of Means
		95% Confidence Interval of the Difference
		Upper
Overall recognition accuracies using Database 2	Equal variances assumed	5.84209%
	Equal variances not assumed	5.84601%

Table 8 Independent-Samples T test for right-handed and left-handed participants' test samples using Database 1

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means
		F	Sig.	t
Overall recognition accuracies using Database 1	Equal variances assumed	.848	.365	.461
	Equal variances not assumed			.331

Independent Samples Test

		t-test for Equality of Means		
		df	Sig. (2-tailed)	Mean Difference
Overall recognition accuracies using Database 1	Equal variances assumed	28	.648	2.84239%
	Equal variances not assumed	2.203	.770	2.84239%

Independent Samples Test

		t-test for Equality of Means	
		Std. Error Difference	95% Confidence Interval of the Difference
			Lower
Overall recognition accuracies using Database 1	Equal variances assumed	6.16144%	-9.77874%
	Equal variances not assumed	8.59044%	-31.04101%

Independent Samples Test

		t-test for Equality of Means
		95% Confidence Interval of the Difference
		Upper
Overall recognition accuracies using Database 1	Equal variances assumed	15.46352%
	Equal variances not assumed	36.72578%

Table 9 Independent-Samples T test for right-handed and left-handed participants' test samples using Database 2

Independent Samples Test

		Levene's Test for Equality of Variances		t-test for Equality of Means
		F	Sig.	t
Overall recognition accuracies using Database 2	Equal variances assumed	1.656	.209	.390
	Equal variances not assumed			.260

Independent Samples Test

		t-test for Equality of Means		
		df	Sig. (2-tailed)	Mean Difference
Overall recognition accuracies using Database 2	Equal variances assumed	28	.700	2.49781%
	Equal variances not assumed	2.169	.817	2.49781%

Independent Samples Test

		t-test for Equality of Means	
		Std. Error Difference	95% Confidence Interval of the Difference
			Lower
Overall recognition accuracies using Database 2	Equal variances assumed	6.40723%	-10.62681%
	Equal variances not assumed	9.59176%	-35.83636%

Independent Samples Test

		t-test for Equality of Means
		95% Confidence Interval of the Difference
		Upper
Overall recognition accuracies using Database 2	Equal variances assumed	15.62244%
	Equal variances not assumed	40.83198%

Appendix E Additional findings in program's performance for different user groups

Difference in recognition accuracies based on users' experience with video games

In order to discover if the participants with different experience playing video games obtained similar overall recognition accuracies, an ANOVA was conducted on each of the two databases. In this case, the participants' experience playing video games was selected as the independent variable. The analysis results using Database 1 and Database 2 are illustrated in Table 10.

Table 10 One-Way ANOVA for User's Video Game Experience

Descriptives

		N	Mean	Std. Deviation	Std. Error
Overall recognition accuracies using Database 1	0 hour	12	84.8837%	8.67325%	2.50375%
	Less than 1 hour	9	86.0465%	8.22218%	2.74073%
	1-3hrs.	6	82.5581%	11.08011%	4.52344%
	2-5hrs.	2	86.0465%	13.15544%	9.30230%
	More than 5 hours	1	58.1395%	.	.
	Total	30	83.9535%	9.98592%	1.82317%
Overall recognition accuracies using Database 2	0 hour	12	77.7131%	10.88349%	3.14179%
	Less than 1 hour	9	80.1033%	10.97659%	3.65886%
	1-3hrs.	6	80.6202%	9.71936%	3.96791%
	2-5hrs.	2	77.9070%	18.08878%	12.79070%
	More than 5 hours	1	76.7442%	.	.
	Total	30	78.9922%	10.37309%	1.89386%

Descriptives

	95% Confidence Interval for Mean	Minimum

		Lower Bound	Upper Bound	
Overall recognition accuracies using Database 1	0 hour	79.3730%	90.3944%	72.09%
	Less than 1 hour	79.7264%	92.3666%	69.77%
	1-3hrs.	70.9303%	94.1860%	67.44%
	2-5hrs.	-32.1504%	204.2434%	76.74%
	More than 5 hours	.	.	58.14%
	Total	80.2247%	87.6823%	58.14%
Overall recognition accuracies using Database 2	0 hour	70.7981%	84.6282%	60.47%
	Less than 1 hour	71.6659%	88.5406%	67.44%
	1-3hrs.	70.4203%	90.8200%	69.77%
	2-5hrs.	-84.6143%	240.4283%	65.12%
	More than 5 hours	.	.	76.74%
	Total	75.1188%	82.8656%	60.47%

Descriptives

		Maximum
Overall recognition accuracies using Database 1	0 hour	97.67%
	Less than 1 hour	97.67%
	1-3hrs.	95.35%
	2-5hrs.	95.35%
	More than 5 hours	58.14%
	Total	97.67%
Overall recognition accuracies using Database 2	0 hour	90.70%
	Less than 1 hour	95.35%
	1-3hrs.	93.02%
	2-5hrs.	90.70%
	More than 5 hours	76.74%
	Total	95.35%

ANOVA

		Sum of Squares	df	Mean Square	F
Overall recognition accuracies using Database 1	Between Groups	736.616	4	184.154	2.136
	Within Groups	2155.222	25	86.209	
	Total	2891.838	29		
Overall recognition accuracies using Database 2	Between Groups	54.053	4	13.513	.110
	Within Groups	3066.373	25	122.655	

Total	3120.426	29		

ANOVA

		Sig.
Overall recognition accuracies using Database 1	Between Groups	.106
	Within Groups	
	Total	
Overall recognition accuracies using Database 2	Between Groups	.978
	Within Groups	
	Total	

The ANOVA table above shows the results of the overall analysis of variance, including between groups, within groups, as well as the total sum of squares, degrees of freedom and mean squares. The F-ratios for the analysis using the two databases are 2.136 and 0.110, respectively, with the probabilities of 0.106 and 0.978 when using Database 1 and Database 2. Both of these probabilities are greater than 0.05; therefore, the participants with various experience playing video games obtained similar mean accuracies. In conclusion, the program has consistent performance for users who have various experience levels playing video games.

Difference in recognition accuracies based on users' experience with motion control devices

The ANOVA applied in this part took participants' experience using motion control devices as the independent variable. The analysis results using Database 1 and Database 2 are presented in Table 11.

Table 11 One-Way ANOVA for User's Motion Control Devices Experience

Descriptives

		N	Mean	Std. Deviation	Std. Error
Overall recognition accuracies using Database 1	No	8	79.0698%	12.24289%	4.32851%
	Weekly	1	76.7442%	.	.
	Monthly	4	88.9535%	5.15664%	2.57832%
	Few times a year	17	85.4993%	9.28522%	2.25200%
	Total	30	83.9535%	9.98592%	1.82317%
Overall recognition accuracies using Database 2	No	8	78.4884%	9.60875%	3.39721%
	Weekly	1	65.1163%	.	.
	Monthly	4	86.0463%	5.02424%	2.51212%
	Few times a year	17	78.3857%	11.17624%	2.71064%
	Total	30	78.9922%	10.37309%	1.89386%

Descriptives

		95% Confidence Interval for Mean		Minimum
		Lower Bound	Upper Bound	
Overall recognition accuracies using Database 1	No	68.8345%	89.3051%	58.14%
	Weekly	.	.	76.74%
	Monthly	80.7481%	97.1588%	83.72%
	Few times a year	80.7253%	90.2733%	69.77%
	Total	80.2247%	87.6823%	58.14%
Overall recognition accuracies using Database 2	No	70.4553%	86.5215%	67.44%
	Weekly	.	.	65.12%
	Monthly	78.0516%	94.0410%	79.07%
	Few times a year	72.6394%	84.1320%	60.47%
	Total	75.1188%	82.8656%	60.47%

Descriptives

		Maximum
Overall recognition accuracies using Database 1	No	90.70%
	Weekly	76.74%
	Monthly	95.35%
	Few times a year	97.67%
	Total	97.67%
Overall recognition accuracies using Database 2	No	93.02%
	Weekly	65.12%

	Monthly	90.70%
	Few times a year	95.35%
	Total	95.35%

ANOVA

		Sum of Squares	df	Mean Square	F
Overall recognition accuracies using Database 1	Between Groups	383.401	3	127.800	1.325
	Within Groups	2508.437	26	96.478	
	Total	2891.838	29		
Overall recognition accuracies using Database 2	Between Groups	399.865	3	133.288	1.274
	Within Groups	2720.561	26	104.637	
	Total	3120.426	29		

ANOVA

		Sig.
Overall recognition accuracies using Database 1	Between Groups	.288
	Within Groups	
	Total	
Overall recognition accuracies using Database 2	Between Groups	.304
	Within Groups	
	Total	

The ANOVA table above shows the results of the overall analysis of variance. According to the tables, the F-ratios for the analysis using the two databases are 1.325 and 1.274, respectively, with the probabilities of 0.288 and 0.304 using Database 1 and Database 2. Similar to the above results, both of the two probabilities in this analysis are greater than 0.05; therefore, the participants, who have different levels of experience using motion control devices obtained similar overall mean accuracies. It can be concluded that the program consistently performed no matter if a user has experience in using motion control products or not.