Department of Computer Science and Engineering

The Chinese University of Hong Kong

2007 – 2008 Final Year Project Semester 1 Report

Legendary of 18 Weapons – Motion Capture Analysis for Wii Remote

Group: LYU0702

Supervised by

Prof. Michael R. Lyu

Students:

Ng Kwok Ho (05669353)

Tsoi Chi Hung (05619693)

ABSTRACT

Motion capture refers to the technique of recording movements in a digital manner for different purpose in entertainment, sports or medical treatments. At present, most researches regarding motion capture as a use of accelerometers. Those accelerometers are normally attached to certain parts of the body; say the limbs or the hands. However, there are not much researches relating to traditional Chinese martial Art.

Wii Remote also has built-in accelerometers. A player can connect Wii Remote through Bluetooth to the computer, and capture acceleration values through Wii Remote API, i.e. capture player’s motion. It is interesting to investigate on motion capturing and classification with Wii Remote.

In this report, we are going to describe the motivation, background information, what we have done in this semester, as well as problems and limitations we encountered so far. Our objective is to first build a motion classifier to classify motions done with the use of Wii Remote, so that we can further make use of our classifier, either using in games, or for educational purpose.

In the following sections, we would first give some background information as well as the idea of this final year project. Following is different classification algorithms and representations of motion data we have studied. Then we will take about the details of our implementation, and finally we will mention problems and limitations we encountered, as well as our future work.

TABLE OF CONTENTS

CHAPTER *1* INTRODUCTION

In this chapter we are going to describe some background information about current game industry, the players, the revolution brought by Wii, as well as our objectives and motivation. The development environment of our project will be discussed here as well.

This chapter comprises the following sections:

1.1 INTRODUCTION TO THE GAME INDUSTRY

The game industry is mainly composed of two parts – video games, and computer games. Since the birth of game industry in 1972, the industry evolved rapidly, and many new features were introduced with the advancement of hardware and technology from time to time.

However, since the introduction of new consoles using 32-bit technology in 1994, the development of the game industry started to stabilize and tend to be mature. There are several reasons for this:

1. With the invention of 32-bit technology, games could make use of 3D graphics rather than 2D graphics previously. Many of the games since then are some remake of the previous ones, or are originated from them. New ideas are not common. This lacks creativity.

2. The video game market is dominated by proprietary standards from big manufacturers, such as Sony PlayStation® and Microsoft Xbox 360®. In the meantime, to prevent obsolescence of old consoles, the owners of the standards are not willing to release new upgrades frequently, or introduce new features which requires hardware support. This slows down the development.

3. In the computer game market, undoubtedly it is dominated by a single hardware standard – the personal computer [1]. Both players and the manufacturers are limited by the input device – keyboard. This limitation also exists in video game market – the game pad limits everything. This narrows the range of games and limits creativity.

1.2 WHAT DO GAME PLAYERS WANT?

There are 2 big aspects that the player wants – variety of inputs, as well as reality. Players want to have different types of inputs, rather than the traditional keyboards / gamepads to control, to play the game in a more realistic way, or have a more realistic feeling, so that they can enjoy the game more.

One common example is the steering wheels for playing vehicle games. Players can play the control the vehicle in a realistic way, say by rotating the steering wheel to control the car movement, or applying brakes with their legs to stop the car.

Fig 1.1 A typical steering wheel specially designed for games

Konami's Dance Revolution is another example making use of different inputs. Players can dance on a specially designed game pad and this is considered as the actual movement inside the game. Undoubtedly players can enjoy this very well.

Fig. 1.2 Dance Pad specially designed for Konami's Dance Revolution

1.3 THE REVOLUTION OF WII

Nintendo Wii®, first available in November 19 2006, is considered as a revolution in Video Games.

Nintendo noticed that classical game controllers haven't been changed for more than two decades, and thought this limited creativity and ranges of games. "There are examples of controllers that were made for specific games such as Konami's Dance Revolution. And for a long time, we thought that changing the interface would broaden game design and loosen creative constraints on programmers." said Shigeru Miyamoto from Wii development team [2].

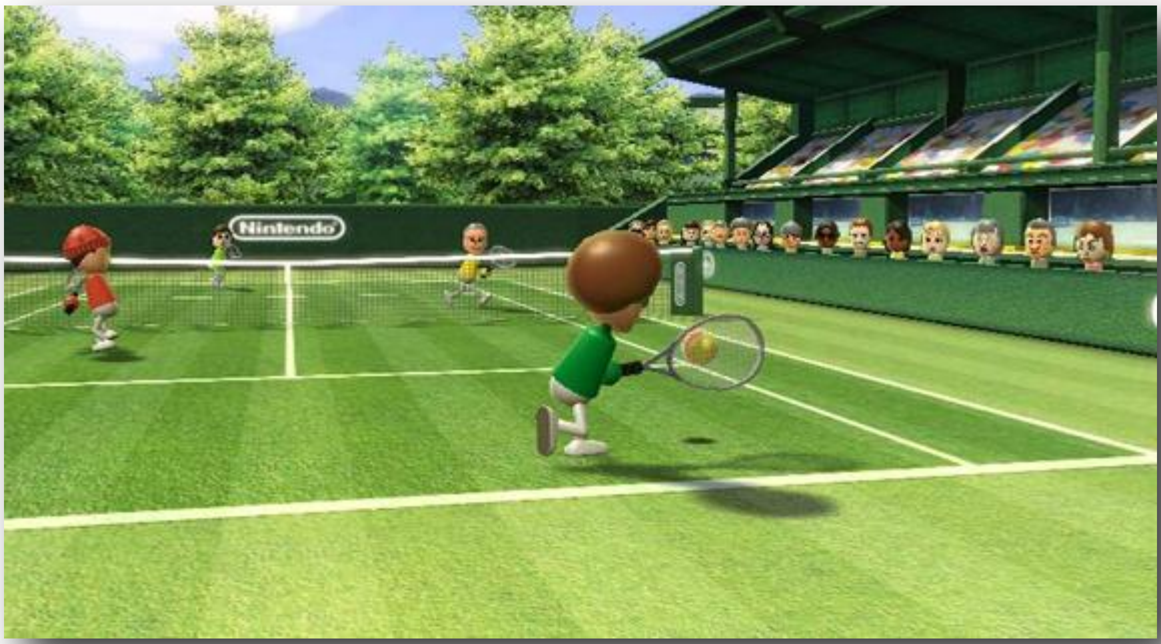


Fig. 1.3 Screen shot of Wii Sports – the tennis

So Nintendo started developing a new console with codename "revolution", Wii is the final name for this console. One of the most distinctive features is the design of the game controller. Unlike controllers of other consoles, Wii controller is a remote – *Wii Remote*. This remote has several important features:

1. Motion sensors are installed inside the remote to sense movements from players, which is treated as the input to the Wii machine. Instead of holding the gamepads and press buttons to control, Wii Remote allows user to control by just waving the remote, which is more realistic.

2. The remote is designed in such a way that motion detection is more accurate and intuitive [4].
3. The remote communicates with the console through Bluetooth. This allows maximum flexibility to the player when motion is performed.

With the Wii Remote, players can play the game in a more realistic way; say you can play tennis or golf games in Wii by performing actual tennis or golf-like motions. This was never been possible before the invention of Wii.

1.4 What is lacking in Wii?

Although Wii Remote has built-in motion sensors, still in most of the games, they are used to detect movements only. Only a few games make use of Wii Remote to perform motion classification. This does not make full use of Wii Remote. In the mean time, even for those games with motion classification, the performances are not very good still.

In other words, Wii is still lacking games with accurate motion classification.

Fig. 1.4 Victorious Boxers: Revolution, one of the Wii game which requires motion recognition

On the other hand, the majority of Wii games are manufactured by game companies from Japan and Western countries, in which the ideas are originated from their home countries. The game market is still lacking games originated from our country – the mainland China; there are rooms to develop games in which the idea comes from our traditional cultures.

Furthermore, being inspired by Nintendogs, a Nintendo DS game featuring dog raising, player should actually be able to define their own motions so that they can have fun in the game using their own style of motions.

Fig 1.5 In Nintendogs, you'll be able to name your pooch using the game's voice-recognition software, repeating it a few times until your pooch "learns" his or her name.

1.5 WHAT IS IN LEGENDARY OF 18 WEAPONS?

In Chinese martial art, one of the well-known components is the "Eighteen Arms". In fact "Eighteen Arms" is the list of the eighteen major weapons used in traditional martial art. There were rumors saying that these eighteen weapons first appeared in a book eight hundred years before. Although there are inconsistencies on the correct list of eighteen weapons, still all lists contains most of the following weapons:

- Dao 刀, Qiang 槍, Jian 劍, Halberd 戟
- Axe 斧, Battle axe 鉞, Hook sword 鉤, Fork 叉
- Trident-halberd 镗, Cudge 棍, Lance 槊, Short cudgel 棒
- Chain whip 鞭, Mace 鐧, Hammer 錘, Talon 抓
- Stick 拐, Meteor Hammer 流星錘

1.6 OUR MOTIVATION & OBJECTIVE

Although Wii is a revolutionary console in the game industry, still people can only enjoy such new methods by buying a console. However, it is relatively costly to buy a console separately (around HKD$2000), while buying a remote just costs you one-tenth of the price.

In the meantime, every family is equipped with a computer in major modern societies such as Hong Kong. The computer can serve for many purposes including computer games and many people love playing computer games, too. Still at this moment there are no games for users to perform some motions as control in the game.

With the popularity of Bluetooth, one can purchase a Bluetooth USB adapter with around HKD$60 in different computer shops in Hong Kong. The adapter can pair with the Wii Remote, and connects it with the computer. That means user can use the Wii Remote to serve as an input to the computer. This is rather important as user can perform Wii-like motions in front of the computer, and the movements can be detected.

Fig 1.6 A typical Bluetooth® USB adapter

We believe that, if the data captured from the Wii Remote are analyzed with accurate motion classification algorithms, together with specially-designed games, a player can perform exactly the same thing to personal computers as what the player may do in front of the Wii console. The performance can be even better due to the high computational power of personal computers when compared to Wii console, as well as the presence of accurate motion classification algorithms.

Moreover, we also believe that this is a much bigger market since the number of personal computers over all the families must be more than the number of Wii consoles. Undoubtedly this will be a big market to all game developers and a new area of game playing, and we believe many interesting findings or creations can be done from this new area.

This is our motivation.

With this motivation, we decide to work on this project. Our objective is to first construct a motion classifier, using data from motion sensors in Wii Remote as the input. With this it is possible for players to perform complicated motions and be recognized accurately in a short time.

When the motion classifier is ready, the next step is to build an interface for players to perform motions directly in front of the computer, and be recognized correctly. The interface can be in the form of a game, or educational software.

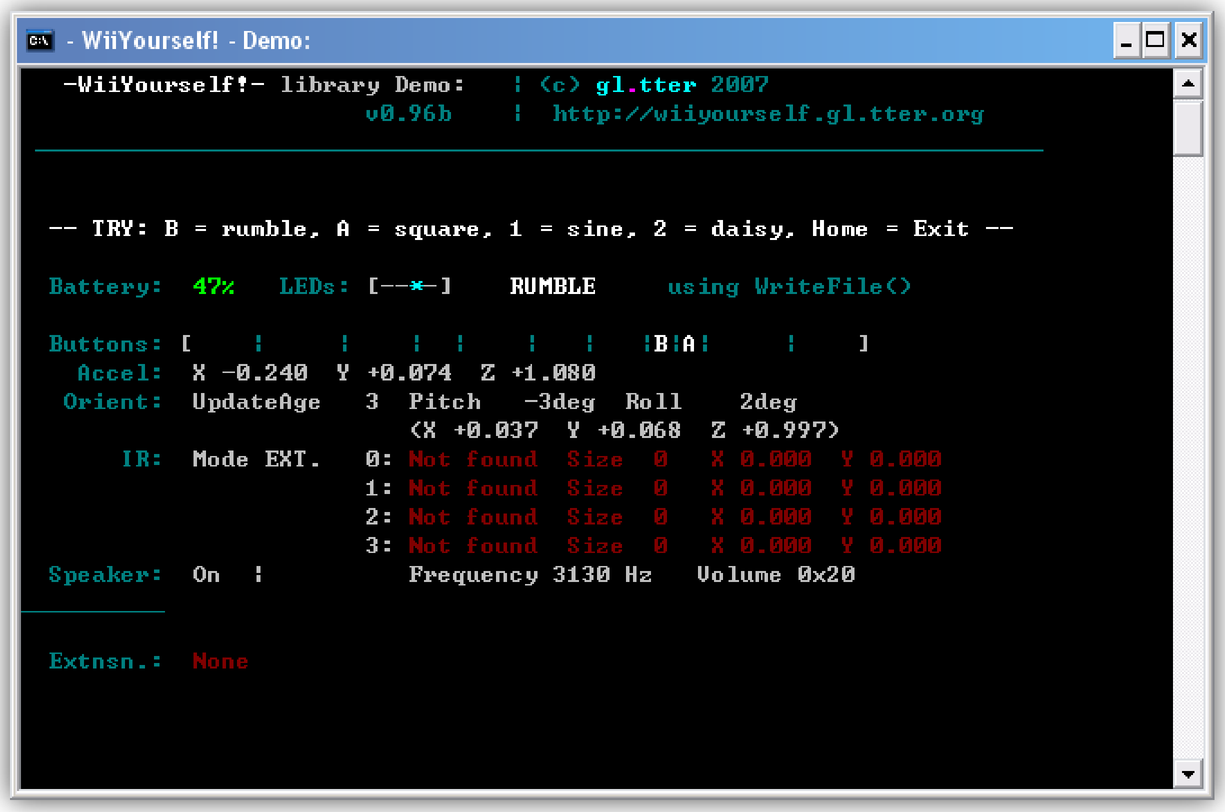
With this, players can enjoy playing with such revolutionary approaches without the presence of Wii console. Players can use any modern computers to play rather than a designated machine; this increases flexibility and players do not need to pay extra costs.

1.7 Development Environment

Our system is developed under Microsoft Windows XP Professional Edition. Windows XP is chosen as the development system since it is currently the most popular operating system in personal computers.

To retrieve sensor inputs from Wii Remote, we have to use a Wii Remote API. We have selected *WiiYourself!* [5] as the API for our project. We select *WiiYourself!* because the source is freely available over the internet, such that we can integrate it with our system easily. At the same time it can provide some other useful data to help us in computation.

In our project, we adopt C++ as our programming language, and Microsoft Visual Studio as our programming environment. We choose C++ because most Wii Remote API available, including *WiiYourself!*, are also written in C++. To facilitate system development, we decide to adopt C++ as our programming language. Microsoft Visual Studio allows a hierarchical structure among files and so we can organize our coding well.



```
 -WiiYourself!- library Demo:   | (c) gl.tter 2007
                    v0.96b      |  http://wiiyourself.gl.tter.org


-- TRY: B = rumble, A = square, 1 = sine, 2 = daisy, Home = Exit --

Battery:  47%   LEDs: [--*-]    RUMBLE     using WriteFile()

Buttons: [    |     |    |  |    |   |   |B|A|     |    ]
  Accel:  X -0.240  Y +0.074  Z +1.080
 Orient:  UpdateAge   3  Pitch   -3deg  Roll    2deg
                         (X +0.037  Y +0.068  Z +0.997)
     IR:  Mode EXT.   0: Not found  Size  0   X 0.000  Y 0.000
                      1: Not found  Size  0   X 0.000  Y 0.000
                      2: Not found  Size  0   X 0.000  Y 0.000
                      3: Not found  Size  0   X 0.000  Y 0.000
Speaker:  On  |          Frequency 3130 Hz   Volume 0x20


Extnsn.:  None
```

Fig 1.7 A screenshot of *WiiYourself!* API Demo

CHAPTER 2 – WII REMOTE OVERVIEW

In this chapter we are going to introduce the main apparatus for this project – the Wii Remote. As claimed by Nintendo, the wireless Wii Remote is a multi-functional device, and it is only limited by game designer’s imagination. Apart from its main functionalities – pointing and motion-detection, Wii remote can also serve for other purposes include controller feedback and speaker.

In this chapter, we will briefly introduce Wii Remote’s design as well as the functionality. We will see a general picture on the use of Wii Remote on Wii Games

This chapter comprises the following sections:

2.1 DESIGN

The Wii Remote, as its name specified, is a remote-like controller instead of the traditional gamepad controllers from previous gaming consoles. Unlike the traditional controllers, which require both hands to hold the controller, Wii Remote has a one-handed design such that players can handle them with either one of two hands. Its dimension – 148 mm long, 36.2 mm wide, and 30.8 mm thick, allows most of the players to hold the Wii Remote comfortably, such that they can enjoy using it more.

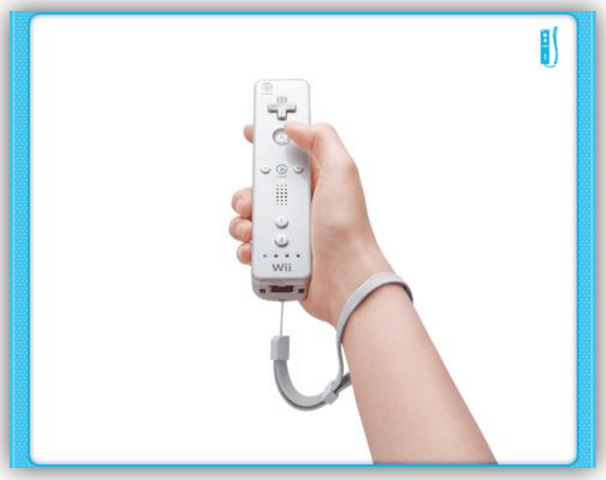


Fig 2.1 Nintendo's official picture of holding a Wii Remote

This design has 2 advantages [4] to suit for its purpose:

1. Motion sensing is more intuitive. In Wii, instead of pressing buttons to control everything inside the game, the majority of controls are done based on player's motion. To let players control everything easily, motion sensing has to be done in an easy, straight forward manner. The remote-like design of Wii Remote facilitates motion sensing a lot. It's not hard to image how difficult it is when players are asked to hold the traditional controllers to perform some motions.

2. Fit perfectly for pointing. Some types of games, for example, shooting games, require players to focus on a particular target to perform some actions, say shooting. As Wii Remote serves as the controller, undoubtedly player has to point to the correct position to specify the target. With Wii Remote, such pointing action is just as natural as using a TV remote and point to TV's infra-red sensor for operation.

2.2 FUNCTIONALITY

In this section we will focus on 2 main functionalities of Wii Remote – motion sensing and pointing.

2.2.1 MOTION DETECTION [12]

The motion sensing is done by accelerometers inside the Wii Remote. In fact accelerometers are some cantilevers hewn from silicon and teetering between two electrodes. A 1-volt voltage is constantly applied so that cantilever's beam will vibrate.

When the Wii Remote accelerates – either by rotation or movement of Wii Remote, the beam's tip traces an ellipse. The value of acceleration can then be measured based on the eccentricity of the ellipse.

Each accelerometer is responsible for tracking accelerations in 1 direction only. So to trace accelerations in the space, 3 accelerometers are placed at right angles with each other. The acceleration values are then reported to the console through Bluetooth.

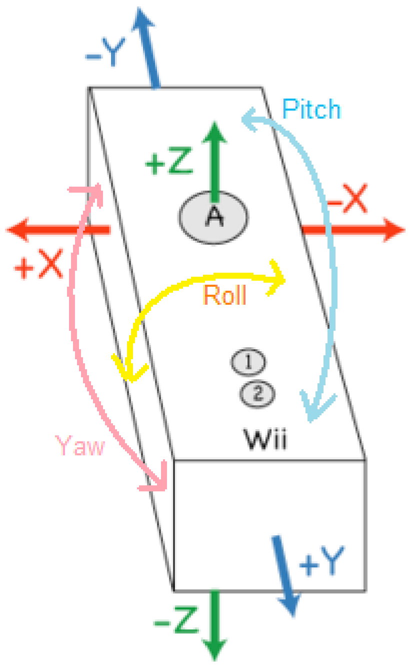
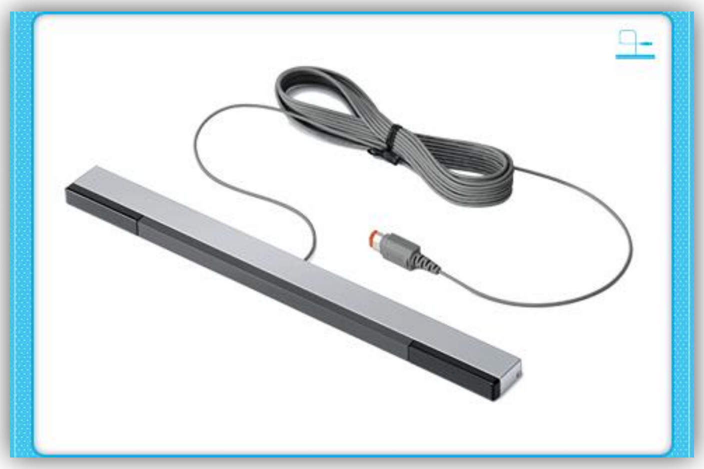


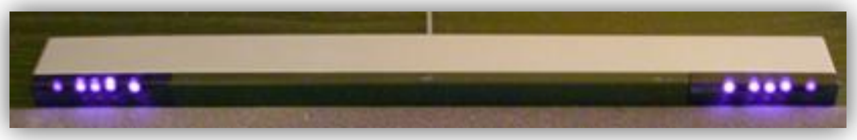
Fig 2.2 The 6 degrees of freedom of Wii Remote

2.2.2 POINTING [4]

Another functionality of Wii Remote is the pointing. Pointing in Wii Remote is achieved by the Wii Remote and Wii sensor bar.



(a)



(b)

Fig 2.3 The Wii infra-red sensor bar

< (a) – the overview; (b) – when the infra-red LEDs are highlighting>

The sensor bar consists of 10 infra-red LEDs, 5 on each end. In each side, the LED farthest away from the centre points slightly away from the centre; while the LED closest to the centre points slightly towards the centre; the remaining LEDs just group together and point straight forward.

Generally the sensor bar is placed at the centre, either above or below the television. The Wii Remote includes a 1024x768 monochrome camera, with an IR-pass filter in front of it. The camera includes a built-in processor capable of tracking up to 4 moving objects. [12] It is used to sense the infra-red emitted from each LED. It senses the light as 2 bright dots and the distance between the Wii Remote and the sensor bar is computed using triangulation. Thus the pointing position can then be known.

2.2.3 Expansion Port – The Nunchuk

There is an expansion port for players to connect other components with the Wii Remote. One of the components that can be connected is the Nunchuk.

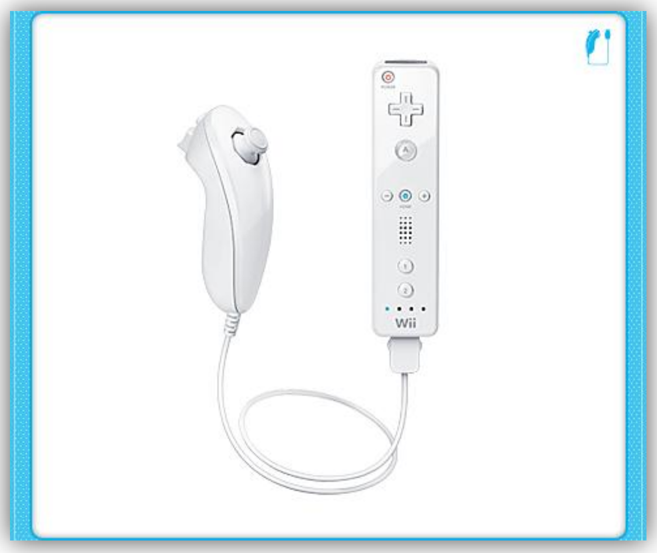


Fig 2.4 How nunchuk (Left) is connected to a Wii Remote

Generally Nunchuk contains the same motion-sensing technology used in the Wii Remote, i.e. it provides accelerometers for motion detection in 3 axes. On the other hand, it also includes an analog stick to assist characters movement, and two buttons for quick access.

Nunchuk works together with the main controller in many games, and this enables the ambidextrous nature of the Wii Remote – making use of both hands in playing games. Such nature is not found in other game controllers.

2.3 WII REMOTE ON WII GAMES

Although Wii Remote is a revolutionary design, still the game developers can design games making use of this remote. Some of the uses are as follows:

1. Adventure and First-person shooting Games. For these types of games, a player can treat the Wii Remote as a weapon, points and attacks an enemy in the game. Two outstanding examples are *Resident Evil 4 : Wii Edition* and *Medal of Honor Heroes 2*.

Fig 2.5 Screenshot of *Resident Evil 4* (Left) and *Medal of Honor Heroes 2* (Right)

2. Role-playing Games. One of the well known games in Hong Kong is *Cooking mama*.

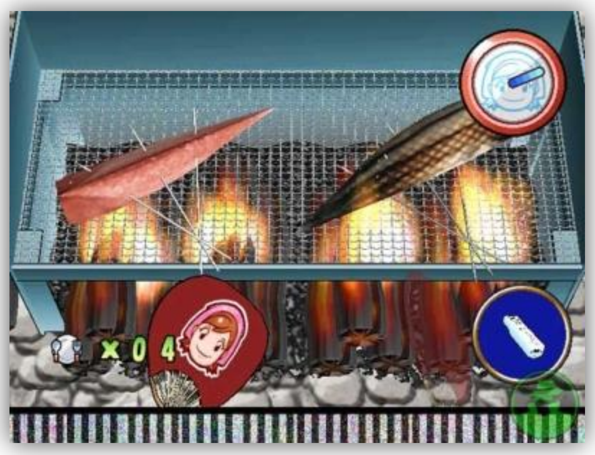


Fig 2.6 Screenshot of *Cooking mama*

The Wii Remote can serve for simulation purpose at this case. A player can imagine himself as holding the fishes on the hand; and rotate the Wii Remote if he wants to cook the other side of the fish.

3. Racing and driving games. Although it does not seem possible to drive cars using Wii Remote, still a number of manufacturers made it possible. One example is the *Mario Dart*.

   It is easy to turn left or right – just rotate the Wii Remote in clockwise or anti-clockwise direction. Some Wii lovers simply made a wheel themselves, with Wii Remote installed at the middle, so that they can enjoy more.

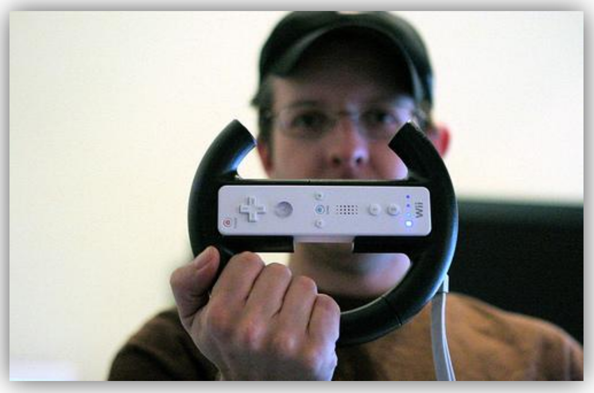
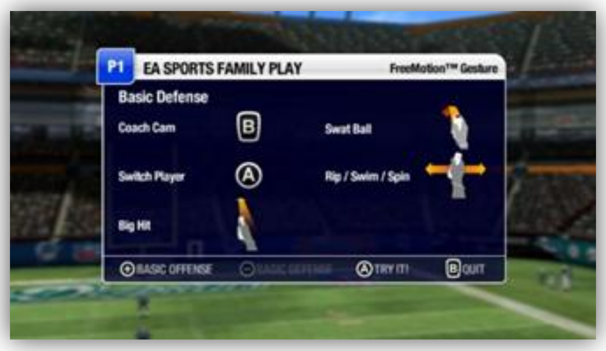


Fig. 2.7 (a) Fig. 2.7 (b)

(a) A screenshot of Mario Dart

(b) The man-made steering wheel

4. Sports game. In some sports, the Wii Remote can be served as the racket that players swing with their arm. Some examples include the tennis and golf. While in some other sports, the Wii Remote can be treated as a ball that the player is holding. Some examples are bowling and America football.



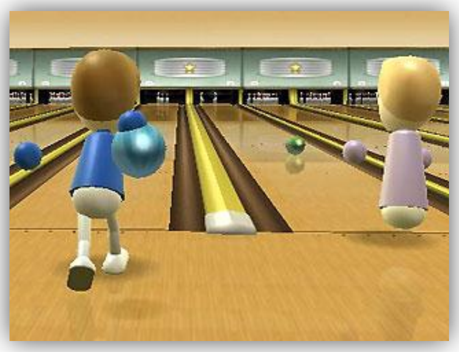


Fig 2.8 (a) Fig 2.8 (b)

(a) A screenshot for *Madden NFL 08* – the America football game; it teaches players on using Wii Remote to defense.

(b) A screenshot from *Wii Sports* – Bowling. Players can play just as if they are holding the ball

5. Fighting games. Traditionally players can fight with the computer by pressing buttons in particular sequences to have special skills be performed. However, with Wii Remote, players can simply move the Wii Remote to achieve the same goal. One outstanding example is *Mortal Kombat: Armageddon*.

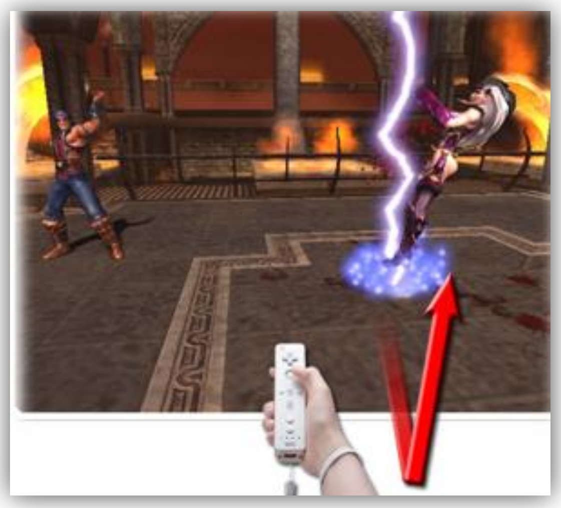


Fig 2.9 Screenshot of *Mortal Kombat: Armageddon*. The arrow indicates how players should move to perform special skills.

On the other hand, by making use of Nunchuk, players can hold both remotes, and attack the enemy as if it is in front of the players. One of such example is *Wii Sports* – Boxing.

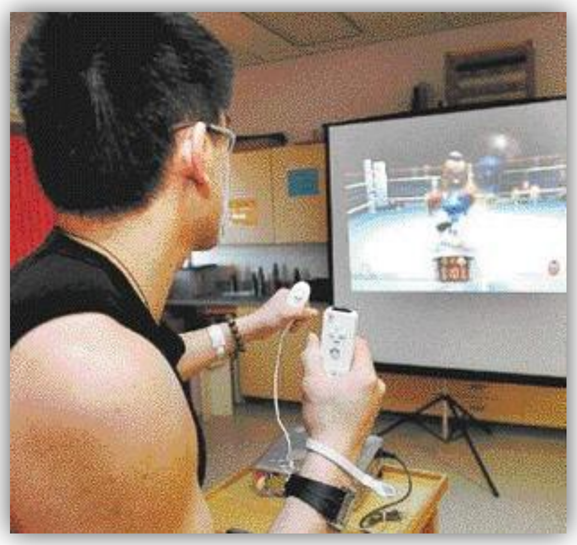


Fig 2.10 A man is playing *Wii Sports* –Boxing with the use of Nunchuk together with a Wii Remote.

CHAPTER 3 – MOTION CLASSIFICATION

In this chapter we are going to describe our studies on different classification methods. One of them is specially designed for Wii Remote on Wii, while others are well known methods used in different kinds of projects or area. We would also discuss how motion data are represented in the computer. In the meantime, we will have some investigation on the inter-relationships between motion classification, motion design and user dependency / independency.

This chapter comprises the following sections:

3.1 EXISTING CLASSIFICATION METHODS FOR WII - AILIVE LIVEMOVE

LiveMove, and recently has its new version renamed as LiveMove *Pro*, is invented by AiLive in 2006. LiveMove is generally powered by its context learning technology.

With LiveMove, developer can build motion classifiers by showing examples instead of coding – one can hold a Wii Remote, performs a number of motions repeatedly, and then a classifier is built. At runtime, the constructed motion classifiers can determine the correct motion that the player is performing.

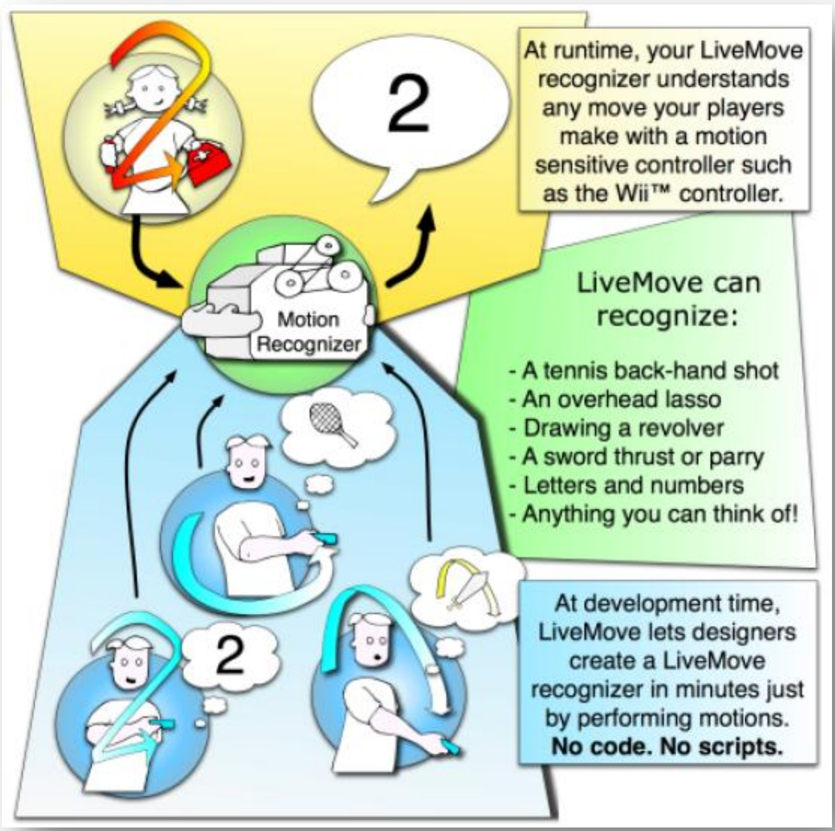

2
At runtime, your LiveMove recognizer understands any move your players make with a motion sensitive controller such as the Wii™ controller.
Motion Recognizer
LiveMove can recognize:
- A tennis back-hand shot
- An overhead lasso
- Drawing a revolver
- A sword thrust or parry
- Letters and numbers
- Anything you can think of!
2
At development time, LiveMove lets designers create a LiveMove recognizer in minutes just by performing motions.
No code. No scripts.

Fig. 3.1 A diagram summarizes the work flow of LiveMove [6]

There are several features in LiveMove *Pro* [6] :

1. Zero-lag recognition. In its newest *Pro* version, motion classification can be done within a few frames after the motion ended. This minimizes the delay between the game responses and player's actual motion.

2. Player Synchronization. In the *Pro* version, several tools are provided to help n synchronizing player's physical motion with the motion on the screen, so that the motion on the screen actually reflects what the player does – in terms of speed, degree of rotation, etc.

Although LiveMove seems to be excellent, but still there are some constraints and limitations with LiveMove:

1. LiveMove is not open source. From its LiveMove Pro Director's Cut Manual[6], it is specified that license for one PC costs about USD$2500. It also has a runtime license to be used in the game, and it costs USD$10000 per game as well. That means a normal human, except those working in wealthy game developers, does not have any chance to use this software.

2. LiveMove is designated for Wii only. From AiLive's site on ordering LiveMove *Pro*, one ordering requirement is the possession of the official Wii NDEV development hardware. This implies that LiveMove can only be used for developing games on Wii platform but not the others.

3. Animations and moves must be pre-defined by the developer / author. Obviously this limits creativity in the player side.

Since there are a number of inconveniences, that's why we would like to construct a new classifier ourselves.

3.2 REPRESENTATION OF DATA

Data representation is an important and yet fundamental step towards any programming. In the following, we will describe the format of data obtained from Wii Remote, and followed by some possible representations.

3.2.1 FORMAT OF DATA

With the use of *WiiYourself!*, the Wii Remote API, accelerations from 3 different and perpendicular axes can be obtained. The values are real number. If the acceleration is 0, this means that the Wii Remote is at rest along a particular axis. If the value is greater than 0, this means that the Wii Remote is moving towards the positive side of that axis, and vice versa. The direction of positive / negative axes can be found in Fig. 2.2. Note that the gravity is also measured by the Wii Remote. Therefore even the Wii Remote is resting on a table, the value of the axis parallel to the gravity is not 0.

By obtaining the three accelerations continuously in a regular interval of time, we will have a set of time series data, which can be represented as three curves in 2 dimensional space with the time axis.

Generally speaking, the accelerations obtained from Wii Remote can have variations and the curve representing them may not be smooth. This may increase the difficulty in comparing the similarities between the inputs and the sample motion in the classifier.

As the data can be represented as curves in 2 dimensional space, in order to have a better result, representation such as Piecewise Linear Approximation, Discrete Fourier Transform and Polynomial Functions can be used for representation. Moreover, since human motions are continuous but not discrete, continuous representation will be best to fill up the gap between two consecutive intervals where no data present.

Apart from accelerations, Wii Remote can also obtain Infra-red information when the Infra-red sensor bar is available. The information obtained is useful for determining the position of the Wii Remote in which it is pointing.

3.2.2 PIECEWISE LINEAR APPROXIMATION (PLA) [13]

Given a curve with n data points, there exist algorithms to approximate the curve into segments of discontinuous straight lines. Such representation is called Piecewise Linear Approximation (PLA) of the curve.

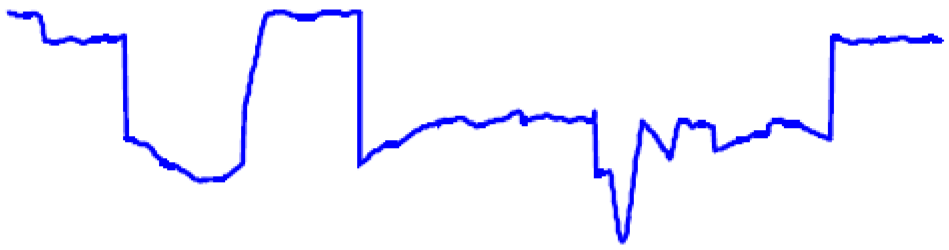


Fig 3.2 (a) A sample data curve

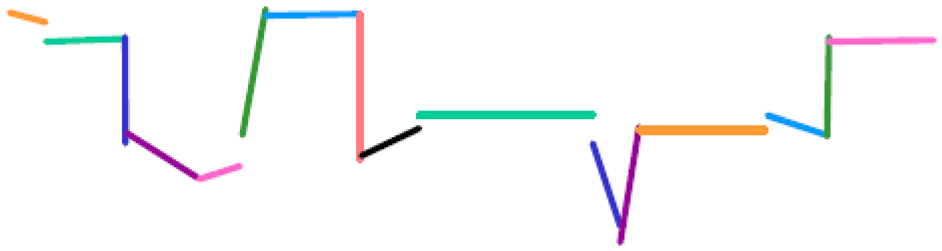


Fig 3.2 (b) The PLA of (a)

We investigated on PLA because there are several advantages:

1. PLA allows similarity search to be performed faster.
2. The change point is sharp so that similarity search can be more accurate.
3. Classification can be done faster and more accurate

Theoretically PLA can be obtained using Dynamic Programming, but it is too slow to be obtained, and so there are a number of alternatives available on the internet, which includes:

1. Top-down. Sequences of data-points are recursively partitioned until the partition meets some stopping conditions.
2. Bottom-up. The algorithm begins from the finest possible approximation; segments are formed by repeated merging until meeting stopping criteria.
3. Sliding Window. Segment keeps on growing until error bound is exceeded.

We have chosen SWAB – Sliding Windows And Bottom-up to implement, a mixture of sliding windows and bottom-up approaches. This is proven to be more accurate and efficient than some existing algorithms in [13].

The algorithm begins by first initializing the buffer size such that 5 to 6 segments can be created. Data points are taken into the buffer continuously, and bottom-up is applied in the buffer. In an iteration, the leftmost segment is “reported” and will not be included in the buffer anymore. The process is repeated until the whole curve is approximated.

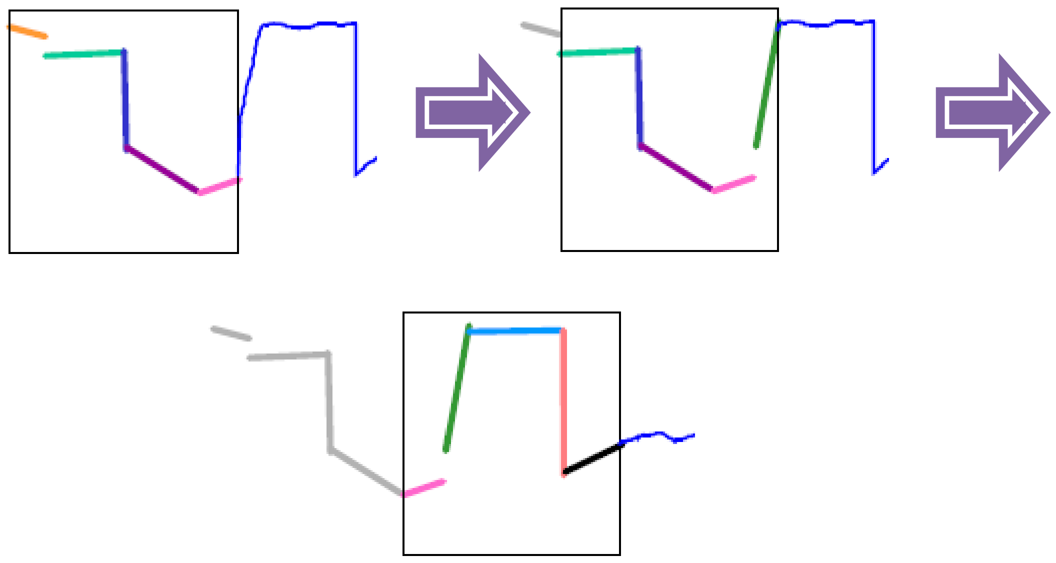


Fig 3.3 Flow chart showing how PLA is applied on the curve of Fig 3.2(a)

3.2.3 Discrete Fourier Transform (DFT) [15]

Mathematically, Discrete Fourier Transform (DFT) is one of the specific forms of Fourier analysis. With DFT, it transforms a periodic sequence $x_0, x_1, \ldots., x_{N-1}$ (non-zero values have a finite duration *N*) into another discrete sequence $X_0, X_1, \ldots, X_{N-1}$, the frequency domain representation of the input sequence, according to the following formula:

$$x_n = \frac{1}{N}\sum_{k=0}^{N-1} X_k e^{\frac{2\pi i}{N}kn} \qquad 0 \le n \le N-1$$

$$X_k = \sum_{k=0}^{N-1} x_n e^{-\frac{2\pi i}{N}kn} \qquad 0 \le k \le N-1$$

where *e* is the base of natural logarithm, *i* is the imaginary unit of complex number.

The equation can be interpreted in the following manner. At point *k*, the sequence value x_k is a linear combination of values of N sinusoids:

$$e^0, e^1, \ldots, e^{\left(\frac{2\pi}{N}\right)k(N-1)}$$

The coefficients of the sinusoids are $X_0, X_1, \ldots, X_{N-1}$ respectively, and their frequencies are $\frac{k}{N}$ cycles per sample. By writing the equations in this form, sinusoids are expressed in the form of complex exponentials making use of Euler's formula.

If DFT is computed directly based on the formulas above, it would take $O(N^2)$ arithmetic operations, which is too slow for classification. So there are a number of algorithms used to compute DFT in only $O(N \log N)$ operations, and they are generally named as Fast Fourier Transform (FFT).

In our investigation, we have chosen FFTW, which stands for the somewhat whimsical title of "Fastest Fourier Transform in the West", to compute DFT. FFTW is a subroutine library for computing DFT in one or more directions. In FFTW, it mainly uses *Cooley-Tukey FFT algorithm* for computation.

Cooley-Tukey FFT algorithm [16] speeds up the computation by re-expressing the DFT of an arbitrary composite size *N* into smaller DFTs of sizes N_1 and N_2, where $N = N_1N_2$, recursively and making use of radix-2 decimation-in-time (Radix-2 DIT).

In Radix-2 DIT, it begins by dividing a DFT of size N into two interleaved DFTs of size N/2 with each recursive stage. The next step is to first compute the Fourier Transforms of the even-indexed numbers $e_m = x_{2m}(x_0, x_2, \dots, x_{N-2})$, and followed by the computation of odd-indexed numbers $o_m = x_{2m+1}(x_1, x_3, \dots, x_{N-1})$. The 2 results are then combined to produce Fourier Transforms of the whole sequence. The decomposition can be summarized as follows:

$$\begin{aligned} X_k &= \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k} \\ &= \sum_{m=0}^{M-1} e_m e^{-\frac{2\pi i}{M}mk} + e^{-\frac{2\pi i}{N}k} \sum_{m=0}^{M-1} o_m e^{-\frac{2\pi i}{M}mk} \\ &= \begin{cases} E_k + e^{-\frac{2\pi i}{N}k} O_k & \text{if } k < M \\ E_{k-M} - e^{-\frac{2\pi i}{N}(k-M)} O_{k-M} & \text{if } k \geq M \end{cases} \end{aligned}$$

where E_j and O_j denotes the DFT of the even-indexed numbers e_m and the odd-indexed numbers o_m, respectively.

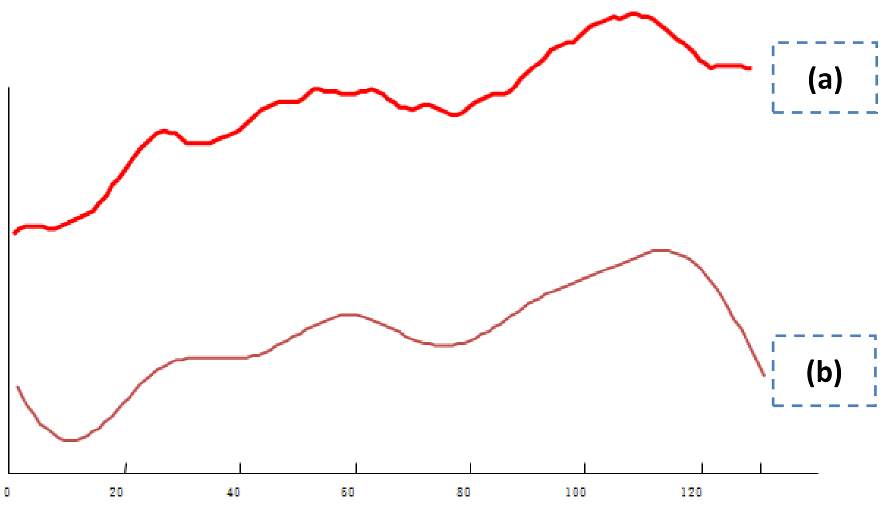


Fig 3.4

Fig 3.4 above illustrates the effect of DFT. In (a) it is the input curve. After performing DFT, the curve becomes smoother as shown in (b).

3.2.4 POLYNOMIAL FUNCTION

Apart from the complicated algorithms above, in fact a curve can also be approximated by using simple regression.

Regression tries to find the "best" curve to fit all the data points. The approximated curve may not pass through all the data points, but the error is guaranteed to be the minimum. Regression is useful when the data has certain amount of noises.

To find the best-fit curve using regression, "Linear Least Square" method can be used. Depend on the complexity of the motions; the degree of the curve that is used to fit the data points can be adjusted. The following shows how a curve with degree 2 can be calculated:

Suppose there are N data points. As the curve is in degree 2, so it must be represented in the following forms:

$$\boldsymbol{y_i = a_0 + a_1 x_i + a_2 {x_i}^2 + e_i}$$
$$\boldsymbol{for\ i = 1, 2, \dots, N, where\ e_i\ represents\ the\ error}$$
$$\boldsymbol{between\ the\ best - fit\ curve\ at\ i\ and\ the\ i^{th}\ data\ point.}$$

With N data points, the above equation can be written in matrix form as follows:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & {x_1}^2 \\ 1 & x_2 & {x_2}^2 \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ 1 & x_n & {x_n}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ \vdots \\ e_n \end{bmatrix}$$

To find the best-fit line, a possible strategy is to minimize the sum of the squares of the residuals between the measured *y* value and the actual data point, i.e.

$$\boldsymbol{S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i - a_2 {x_i}^2)^2}$$

To minimize *S_r*, a possible approach is to set the partial derivatives of *S_r* to be zero and solve the resulting systems of equations such that a_0, a_1 and a_2 can be found.

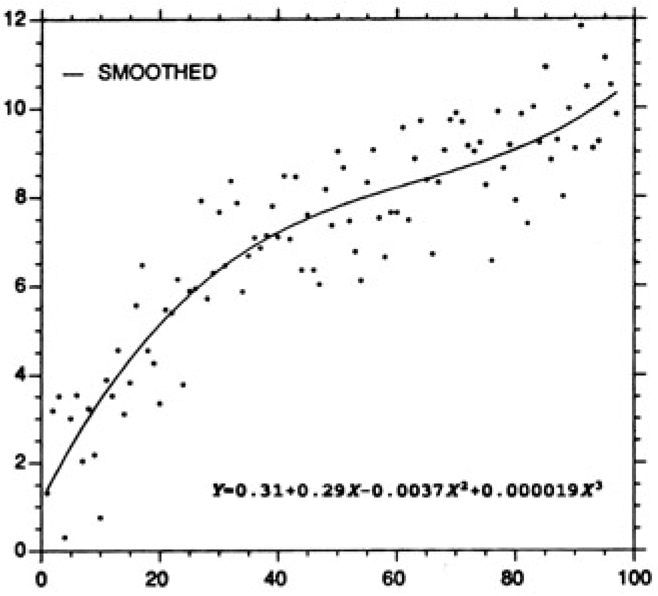


Fig 3.5 A graph showing the best-fit curve after applying linear regression.

3.2.5 RAW DATA

Apart from using different approximation algorithms, we have also used the most basic raw data format for computations. Raw data here refers to the actual acceleration values collected from the Wii Remote. They are used directly without any treatments.

3.3 "USER DEPENDENT" AND "USER INDEPENDENT"

Motion classification is largely depending on how user performs the motions. Different users perform the same action in different styles. For example, in writing the number "2", some people may include a small circle at the bottom left part of the number while some may just write it in the same way as the one appear in this report.

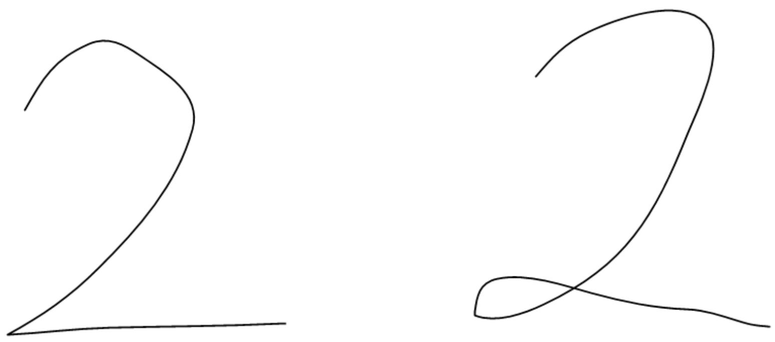


Fig 3.6 Two different styles of writing the number "2"

So here comes the question: How to recognize the motion if different players try to perform the same motion in their own ways? In another words, how to achieve user independent motion classification?

Intuitively, we can include all different styles of the same motions in the database, and compare each motion one by one. This method, however, is time and space inefficient since the number of comparison is large, and storing all the motions is space consuming. Therefore typically, we will observe the pattern of the data, and extract some useful features from it. We then build our rule for classification according to these features extracted from different styles of the same motion.

Achieving user dependent, on the other hand, is a lot easier to be done. The main problem to achieve user dependent is that it is impossible for a user to do the same motion in an identically every time. Besides extracting features from the data, since the motions performed by the same user are more or less the same, therefore we can perform a similarity search on the previously obtained data from that user with a predefined error threshold for classification.

3.4 MOTION DESIGN

Another factor that affects motion classification is the motion itself. For example, having to classify two similar motions, big circle and small circle, will be difficult since the motions are draw in the same way except the size are different.

Designing motions thus becomes a challenge, especially when the number of motions increases. Sometimes, motions have to be redesigned in order to achieve a higher classification rate, or relationships of the motions have to be set up. For instance, a big circle drawing is always followed by small circle drawing.

The main factors that need to be considered in designing motions:

1. What kinds of motions are performing? Are they sword motions? Or they are just 2D numbers drawing in 3D space? What are their features?
2. How the motions are used? Is early recognition required?
3. Are the motions too similar? Are they distinctive enough for classification?
4. Are the motions too complicated for the user to memorize and perform?

3.5 RESEARCHED CLASSIFICATION METHODS

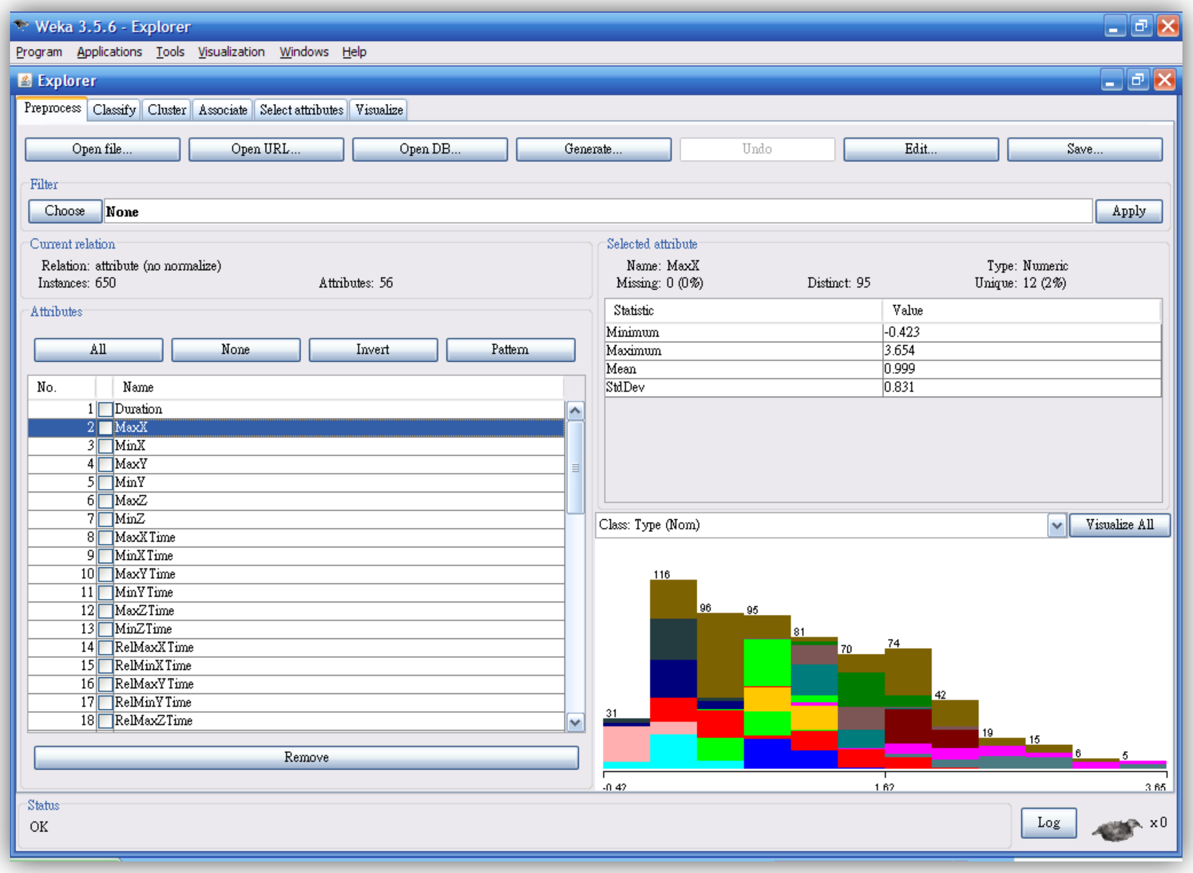
Before we decided to use a particular algorithm for motion classification and recognition, we have studied a number of methods before our final decision. In the 3 sections below we will introduce each of them, the principles behind, and the strength and weaknesses of each of them.

3.5.1 WEKA – A DATA MINING SOFTWARE

3.5.1.1 INTRODUCTION

Weka [7] is an open source software implemented by a research group in The University of Waikato. It is a collection of machine learning algorithms for data mining classes. The software contains different tools for data-preprocessing, classification, regression, clustering, association rules and visualization.

In the book "*Data Mining: Practical machine learning tools and techniques*" [7], it is said there are 3 different ways of using Weka – "Apply a learning method to dataset and analyze the output", "use learned models to generate predictions on new instances", and "apply several different learners and compare their performance in order to choose one for prediction". In our project, we used Weka mainly in the 2nd and 3rd way – we tried several learners to try on a given data set, and we would like to see which learner performs better based on the predictions on new instances, and eventually we can implement classifiers based on the classification results.


Weka 3.5.6 - Explorer
Program Applications Tools Visualization Windows Help
Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Open file... Open URL... Open DB... Generate... Undo Edit... Save...
Filter
Choose
Apply
All None Invert Pattern
Remove
Visualize All
Log
x 0
116
96
95
81
70
74
42
31
19
15
6
5
-0.42
1.62
3.65

Filter: None

Current relation
Relation: attribute (no normalize)
Instances: 650 Attributes: 56

Selected attribute
Name: MaxX Type: Numeric
Missing: 0 (0%) Distinct: 95 Unique: 12 (2%)

Statistic	Value
Minimum	-0.423
Maximum	3.654
Mean	0.999
StdDev	0.831

Attributes

No.	Name
1	Duration
2	MaxX
3	MinX
4	MaxY
5	MinY
6	MaxZ
7	MinZ
8	MaxXTime
9	MinXTime
10	MaxYTime
11	MinYTime
12	MaxZTime
13	MinZTime
14	RelMaxXTime
15	RelMinXTime
16	RelMaxYTime
17	RelMinYTime
18	RelMaxZTime

Class: Type (Nom)

Status
OK

Fig. 3.7 A screenshot of Weka data-preprocessing section

3.5.1.2 TESTED CLASSIFIERS

Weka provides different types of classifier algorithms, which includes – Bayes, Trees, Rules, Functions and Lazy. We have selected 5 classifiers, 3 from Trees and 2 from Rules, to investigate and compare their performances.

	Name	Function [7]
Trees	J48	C4.5 decision tree learner (implements C4.5 revision 8)
	RandomTree	Construct a tree that considers a given number of random features at each node
	REPTree	Fast tree learner that uses reduced-error pruning
Rules	PART	Obtain rules from partial decision trees built using J4.8
	JRip	RIPPER algorithm for fast, effective rule induction
Functions	Multilayer Perception	A Classifier that uses back propagation to classify instances.

Tabel 3.1 Lists of classifiers selected for investigation and their function

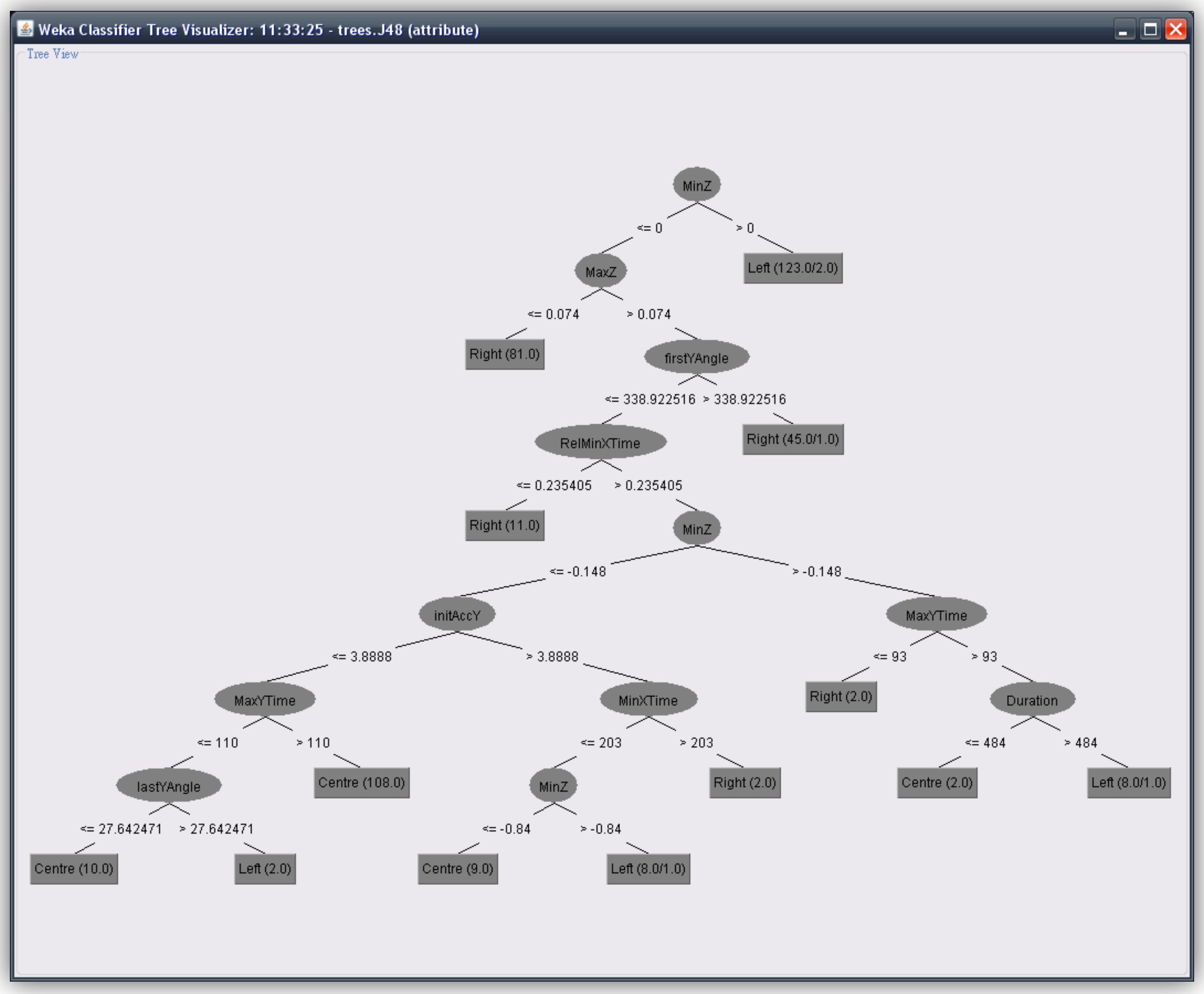


Fig. 3.8 A decision tree generated by Weka using J48 implementation

3.5.1.3 ATTRIBUTES USED FOR CLASSIFIERS

Classifiers will not work just based on the motion data itself, which is just a sequence of numbers without any meaningful information; so we have computed a number of attributes based on the capture motion data, i.e. the time frame number, and the accelerations of the 3 axis, to serve as the inputs for classifiers.

Name of Attribute	Meaning of the attribute
RelMaxXTime / RelMaxYTime / RelMaxZTime	The time relative to the whole motion (**relative time**) when the acceleration is maximum in **positive** X / Y / Z axis direction.
RelMinXTime / RelMinYTime / RelMinZTime	The relative time when the acceleration is maximum in **negative** X / Y / Z axis direction.
relIntersectXY / relIntersectXZ / relIntersectYZ	The relative time when the acceleration of X and Y axis / X and Z axis / Y and Z axis is the same **for the first time**
relZeroAccX / relZeroAccY / relZeroAccZ	The relative time when the acceleration of X / Y / Z axis is zero **for the first time.**
initAccX / initAccY / initAccZ	The average acceleration along X / Y / Z axis in the first 75 milliseconds
endAccX / endAccY / endAccZ	The average acceleration along X / Y /Z axis in the last 75 milliseconds before the motion ends
firstCombinedAcc	The average combined acceleration of X, Y and Z axis in the first 75 milliseconds
lastCombinedAcc	The average combined acceleration of X, Y and Z axis in the first 75 milliseconds
firstXAngle / firstYAngle / firstZAngle	The initial angle of deflection on YX / ZY / XZ plane
endXAngle / endYAngle / endZAngle	The angle of deflection on YX / ZY / XZ plane before the motion ends.

Tabel 3.2 List of attributes we have computed

(Cont'd Table 3.2)

Name of Attribute	Meaning of the attribute
relMaxAccTime	The relative time when the combined acceleration of X, Y and Z axis is maximum.
relMinAccTime	The relative time when the combined acceleration of X, Y and Z axis is minimum.
averageXAcc / averageYAcc / averageZAcc	The average acceleration of X / Y / Z axis throughout the motion
averageCombAcc	The average combined acceleration of X, Y, Z axis throughout the motion

3.5.1.4 DATA PRE-PROCESSING

To further improve classification accuracy, we have done some pre-processing on the computed attributes before it is used for classification.

3.5.1.4.1 Normalization of amplitude [14]

Normalization is any process that makes something more normal, which typically means conforming to some regularity or rule, or returning from some state of abnormality.

In our case, since our input is a set of data points. Normalization is hardly to be done except on the amplitude of the points. In our normalization, we normalize the amplitude between -10 and 10.

3.5.1.4.2 Discretization of attributes

Discretization refers to the separation of numeric attributes into a smaller number of distinct ranges. This idea exists because some classification algorithms can only deal with nominal attributes only and cannot handle attributes on a numeric scale [7]. So as to increase the reliability of classification, attributes are discretized before actual classification starts.

3.5.1.5 PROS AND CONS AMONG CLASSIFIERS

Among the classifiers we have tested, we got the following conclusion about each of them:

1. When compared to neural networks, tree and rule-based classifiers are easy to be coded. Weka visualizes the classification tree or rules to the user after learning is completed. Obviously this makes coding of the decision trees or rules much easier by just using a number of if statements. However, for neural networks, it is not easy to have prototype within a second and thus we cannot use this to construct classifiers easily.

2. From our trials, we found that neural network can always guarantees accuracy greater than 90%. However, the training time is longer when compare to other algorithms.

3. For tree or rule-based algorithm, as well as neural networks, in order to have an accurate result, a large number of data samples must be ready in the learning step.

3.5.2 EUCLIDEAN DISTANCE ALGORITHM

3.5.2.1 INTRODUCTION

Apart from computing attributes based on the acceleration values, another possible approach for motion classification is to perform similarity search among the input signals and the sample data stored in the system, by computing the Euclidean Distance between the 2 different set of points.

3.5.2.2 WORKING PRINCIPLE

Euclidean Distance is defined mathematically as the "ordinary" distance between two points that one would measure with a ruler [8]. It is computed as follows:

$$For\ 2\ set\ of\ points\ P = (p_1, p_2, \dots, p_n)\ and\ Q = (q_1, q_2, \dots, q_n),$$
$$the\ Euclidean\ Distance\ between\ points\ P\ and\ Q, is\ defined\ as:$$

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

To perform similarity search, sequences of acceleration values from both the input signals and the sample data on the same axis, are aligned sequentially. Euclidean Distance is then computed.

The total error of the input motion is defined as the sum of Euclidean Distances for the 3 axis.

To correctly classify the motions, similarity search is repeated for all the samples in the classifier. The classifier will base on the error values, and classify the input motion as a particular motion if the error value is the minimum.

3.5.2.3 STRENGTHS AND WEAKNESS

Euclidean Distance algorithm is straight forward to implement, and classifiers making use of this algorithm can allow user-defined motions to be added to the classifiers. From this aspect, this is better than the tree-based and rule-based classification algorithms mentioned in section 3.5.1.2.

However, the error rate for classification using Euclidean Distance algorithm is high due to its linear one to one mapping. In particular, if the duration for the input motion is longer than the sample motion, then data points near the end is truncated and thus increasing the error. This algorithm assumes there are no variations between 2 sequences in terms of time or speed, which is undoubtedly not suitable for motion classification since different players may do the same motion differently in different trials.

3.5.3 DYNAMIC TIME WARPING (DTW)

3.5.3.1 INTRODUCTION

Dynamic Time Warping (DTW) is yet another algorithm for measuring similarity between two sequences, even if there are variations in time or speed. It can be used to analyze any data which can be turned into a linear representation finally.

One example of the restrictions imposed on the matching of the sequences is on the monotonicity of the mapping in the time dimension. Continuity is less important in DTW than in other pattern matching algorithms; DTW is an algorithm particularly suited to matching sequences with missing information, provided there are long enough segments for matching to occur.[9]

Similar to Euclidean Distance algorithm mentioned 3.5.2, the motion is classified based on the errors with the sample motions in classifiers.

3.5.3.2 Working Principle

In general, Dynamic Time Warping tries to find the best match between two given sequences under some constraints. The sequences are mapped non-linearly within the time dimension so that similarity can still be detected regardless of some non-linear variations in time and speed.

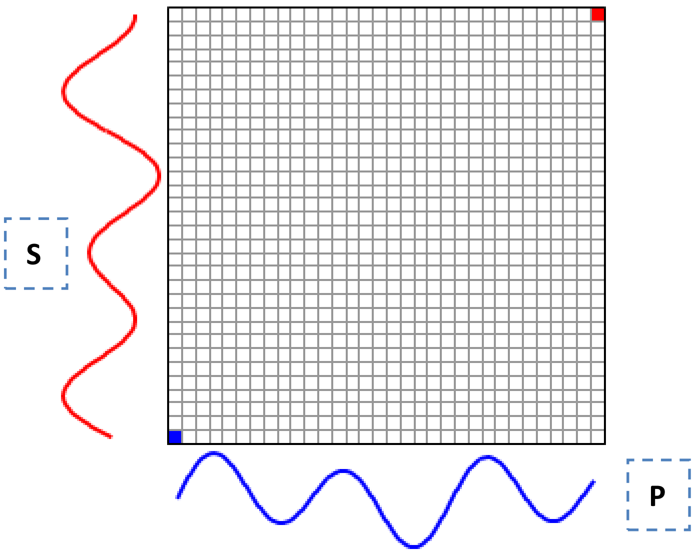


Fig. 3.9 An example grid for DTW [11]

Fig. 3.9 shows a grid for DTW. Generally speaking, red curve represents the stored curve, and blue curve represents the input curve. In our project, the red curve resembles the sequence of acceleration values from a motion sample in the classifier; while the blue curve resembles the sequence of values captured from the Wii Remote by user's motion.

In order to find the best match between the two sequences, we can find a path through the grid which minimizes the total distance to travel from the blue square at the bottom left corner, i.e. the beginning of 2 curves, to the red square at the top right corner, i.e. the ending of 2 curves.

The equation for computing values at each square is defined as follows [11]:

$$\boldsymbol{\gamma(i,j) = d(s_i, p_j) + \min[\gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1)]}$$

where $d(s_i, p_j)$ refers to the difference in values between ith point of stored curve and jth point of the input curve.

After computing the all the values, the warping path, which is the shortest path from the blue square to the red square can then be found, which is shown in the figure below:

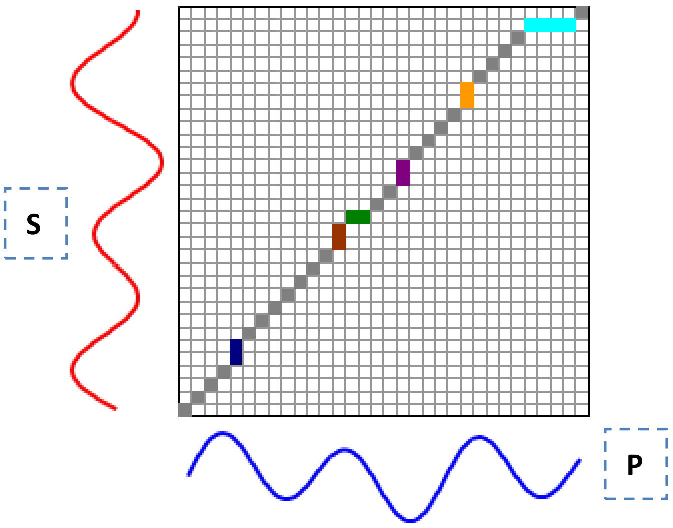


Fig. 3.10 DTW grid with warping path found [11]

With this warping path, a non-linear mapping between 2 curves is formed, and the error is then computed as well.

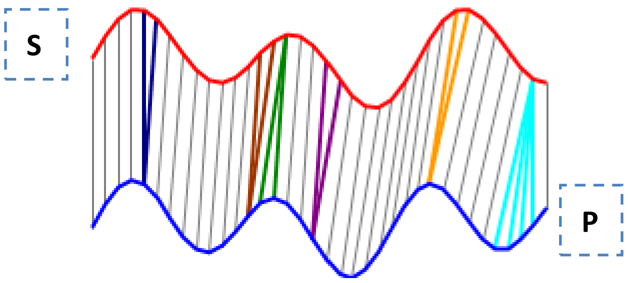


Fig. 3.11 A non-linear mapping between 2 different curves [11]

Fig. 3.11 shows a non-linear mapping between curves S and P mentioned previously. Attention can be drawn to the coloured (and bolded) lines, in which some points map to multiple points, and some other points are mapped with multiple points. The mapping is based on the result from the grid with warping path found.

Although DTW can allow non-linear mapping, however its native version spends too long time to compute values that will not be used. Generally the mapping has some properties [10]:

1. Monotonic condition – the indexes either stay or increase, and will never go back.
2. Continuity condition – only one step advancement at a time
3. Adjustment window condition – the curve generally does not vary much from the diagonal
4. Slope constraint condition – the path is not too sheep or shallow so that short sequences is not matched with long sequences.

By considering these conditions, optimizations can be made. One example is to impose global constraints of the warping path such that boxes outside a particular area will not be computed [11].

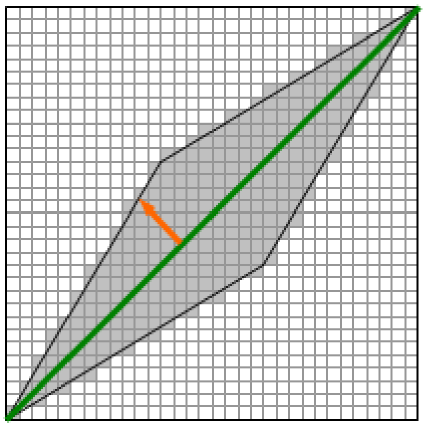


Fig. 3.12 A DWT grid with global constraints imposed.

Fig. 3.12 shows the DWT grid with global constraints imposed. Since the warping path must be obeying the 4 properties stated above, so the values outside the grey area will not be considered and computed, and thus saving computation time.

3.5.3.3 WHY DYNAMIC TIME WARPING?

When compared to Euclidean Distance algorithm, Dynamic Time Warping allows non-linear mapping, thus non-linear variations on time or speed can be overcome. We illustrate the advantage of Dynamic Time Warping with an example below.

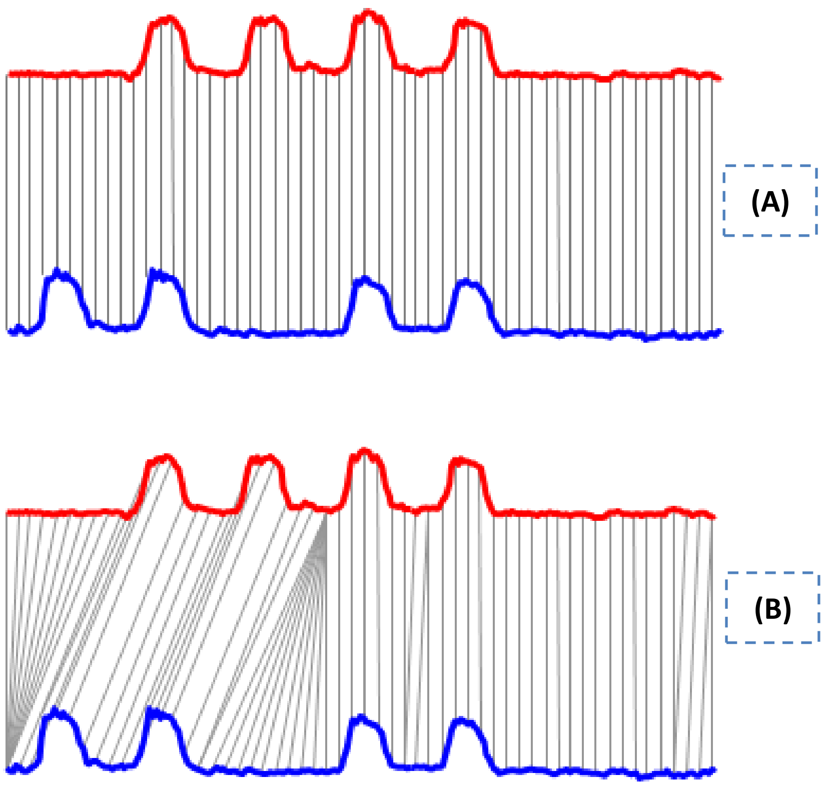


Fig. 3.13 Comparisons between Euclidean Distance algorithm (graph A) and Dynamic Time Warping (graph B) [11]

The figure above shows how a stored curve (the curve on top) is compared with the input curve. Obviously 2 curves are doing the same thing. However, in the stored curve, the 4 crests occur consecutively; while in the input curve, 2 crests occur first, and then followed by a short period of rest, and another 2 crests occur.

As shown in graph A, if Euclidean Distance algorithm is used, the errors will be large due to appearance of crests at different time. On the other hand, if Dynamic Time Warping is used, the time difference in the occurrence of crests can be overcome since one point can map to multiple points. That means crests at the one curve are mapped to corresponding crests at the other curve. The computed error will then be much smaller, and thus the input can then be classified at a higher accuracy.

From the example, we can see why Dynamic Time Warping is more preferable to Euclidean Distance algorithm.

3.5.3.4 STRENGTH AND WEAKNESS

One obvious advantage of Dynamic Time Warping is the high accuracy. No matter user performs the motion with different speed, still it can be classified correctly. On the other hand, it still allows players to define their own motion.

However, as the motion is recognized based on similarity search, so if there exists 2 or more similar motions, the classifier is unable to classify them accurately. On the other hand, it also assumes players to always hold the Wii Remote in the same manner; otherwise it cannot be classified accurately as well.

CHAPTER 4 – OUR RESEARCH

In this semester, we studied different representation of the data, different classification methods as well as different methods to increase the classification rate. At this stage, we have the following configuration:

- Weapon to study: Dao (Sabre)
- Representation of Data: Raw Format
- Data Preprocessing: None
- Classification Algorithm: Dynamic Time Warping on 3D Space

This chapter comprises the following sections:

4.1 TERMINOLOGY

For the sake of easy understandings, we define some terms here which will be used in the following sections.

Terminologies	Meaning
Frame Data	Accelerations data of three axes obtained from Wii Remote in a particular time
Motion Data	A collections of time frame data, representing the whole motion
Motion Database	A collection of different motion data of different motions.
DTW	Dynamic Time Warping
DTW Error	The result generated by Dynamic Time Warping
Model Answer	The sample motion used in Dynamic Time Warping
Average Motion Data	By finding the mapping of the points using DTW, we can find the average of each acceleration values for each point and generate a new average motion data.

Table 4.1 Terminologies which will be used in the later chapters

4.2 THE DESIGN STRUCTURE

Fig 4.1 in the next page illustrates our design structure for the classifier.

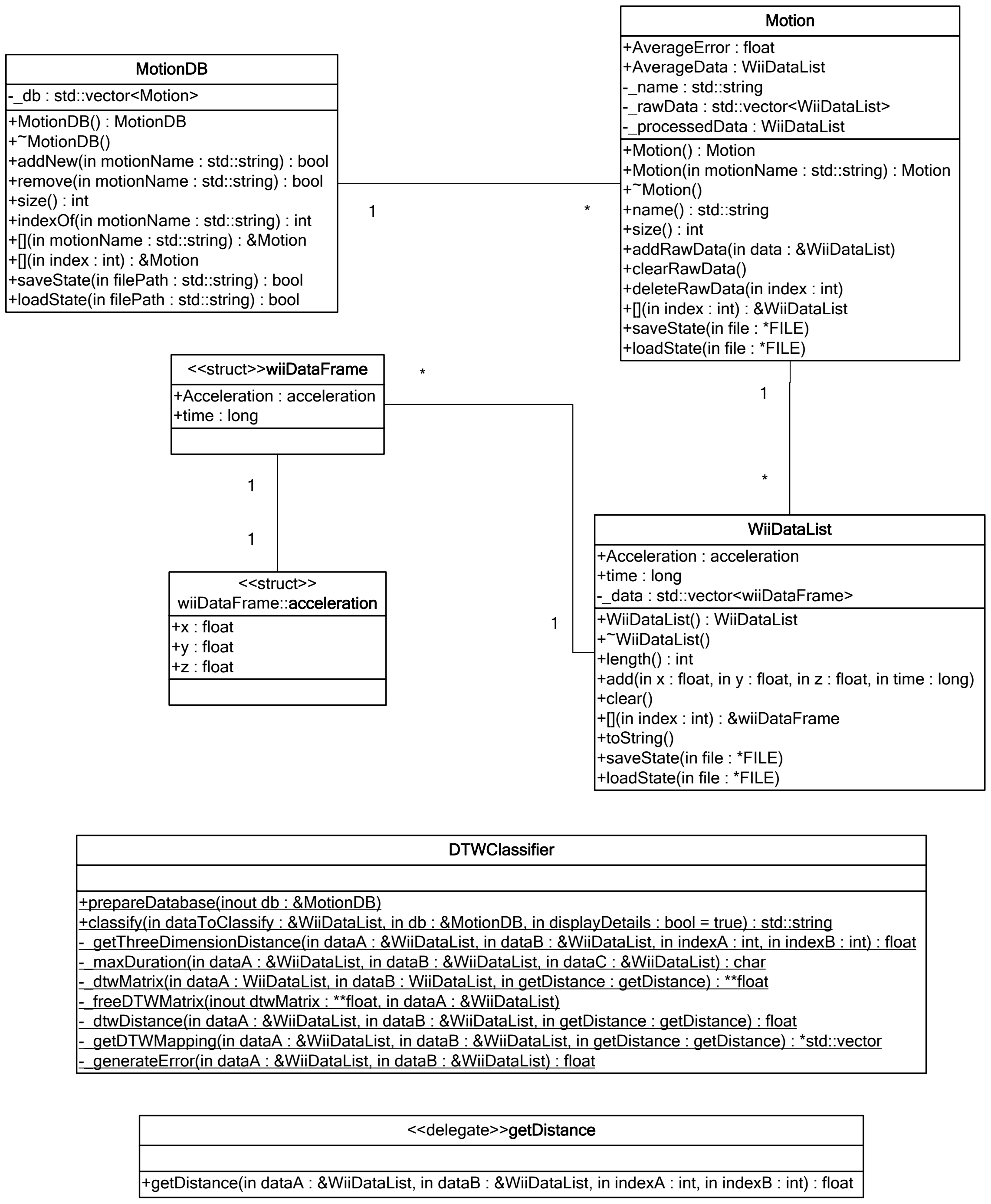

MotionDB
-_db : std::vector<Motion>
+MotionDB() : MotionDB
+~MotionDB()
+addNew(in motionName : std::string) : bool
+remove(in motionName : std::string) : bool
+size() : int
+indexOf(in motionName : std::string) : int
+[](in motionName : std::string) : &Motion
+[](in index : int) : &Motion
+saveState(in filePath : std::string) : bool
+loadState(in filePath : std::string) : bool
1
*
Motion
+AverageError : float
+AverageData : WiiDataList
-_name : std::string
-_rawData : std::vector<WiiDataList>
-_processedData : WiiDataList
+Motion() : Motion
+Motion(in motionName : std::string) : Motion
+~Motion()
+name() : std::string
+size() : int
+addRawData(in data : &WiiDataList)
+clearRawData()
+deleteRawData(in index : int)
+[](in index : int) : &WiiDataList
+saveState(in file : *FILE)
+loadState(in file : *FILE)
1
*
<<struct>>wiiDataFrame
+Acceleration : acceleration
+time : long
*
1
1
1
<<struct>>
wiiDataFrame::acceleration
+x : float
+y : float
+z : float
WiiDataList
+Acceleration : acceleration
+time : long
-_data : std::vector<wiiDataFrame>
+WiiDataList() : WiiDataList
+~WiiDataList()
+length() : int
+add(in x : float, in y : float, in z : float, in time : long)
+clear()
+[](in index : int) : &wiiDataFrame
+toString()
+saveState(in file : *FILE)
+loadState(in file : *FILE)
DTWClassifier
+prepareDatabase(inout db : &MotionDB)
+classify(in dataToClassify : &WiiDataList, in db : &MotionDB, in displayDetails : bool = true) : std::string
-_getThreeDimensionDistance(in dataA : &WiiDataList, in dataB : &WiiDataList, in indexA : int, in indexB : int) : float
-_maxDuration(in dataA : &WiiDataList, in dataB : &WiiDataList, in dataC : &WiiDataList) : char
-_dtwMatrix(in dataA : WiiDataList, in dataB : WiiDataList, in getDistance : getDistance) : **float
-_freeDTWMatrix(inout dtwMatrix : **float, in dataA : &WiiDataList)
-_dtwDistance(in dataA : &WiiDataList, in dataB : &WiiDataList, in getDistance : getDistance) : float
-_getDTWMapping(in dataA : &WiiDataList, in dataB : &WiiDataList, in getDistance : getDistance) : *std::vector
-_generateError(in dataA : &WiiDataList, in dataB : &WiiDataList) : float
<<delegate>>getDistance
+getDistance(in dataA : &WiiDataList, in dataB : &WiiDataList, in indexA : int, in indexB : int) : float

Fig 4.1 The design structure of our data representation structure and classifier

4.3 CLASSIFICATION WORKING FLOW

The classification working flow is divided into 3 main steps – Obtaining Data, Data Classification, and Classification.

1. Obtaining Data
   Before anything can be done, we have to connect to the Wii Remote first, so that we are able to obtain data. By using the API *WiiYourself!*, we can obtain different kinds of information such as accelerations of the x, y, z axes at a particular time. We call this the frame data. Currently, frame data are reported at an interval of 15 milliseconds.

2. Data Preparation
   For DTW to execute, "model answer" from different motions must exist. Therefore, the first step for the classification is data preparation. By obtaining many frame data for a period of time, we obtain the motion data. For each motion, one to several motion data are recorded as "model answer" and stored in the database.

3. Classification
   Finally it is the time to do the classification. We first obtain a motion data that has to be classified, and then put it together with the motion database into the classifier. Depend on what the programmer needs, the classifier can return either the name of the best matched motion, or a list of DTW errors of each motion exist in the database.

4.4 Chosen Weapon – Dao

In the Legendary of Eighteen Weapons, Dao (Sabre), Qiang (Spear), Jian (Sword) and Gun (Staff) are referred as "The General of All Weapons".[17] As a beginning, we chose to use Dao motions since it involves one controller - Wii Remote only. On the other hand, we can have the assumption on how the user hold the Wii Remote, since dao itself is single-edged-bladed weapon. We did not study other weapons in this semester.

Fig 4.2 Dao

4.5 Chosen Data Representation – Raw Format

In the beginning, the first thing we did was to study the representation of the data. Different kinds of representations were studied and were mentioned in section 3.2. However, since there are no immediate available codes for Piecewise Linear Representation, and also we have difficulties in understanding and applying Discrete Fourier Transform, therefore we choose raw format for the data representation. The original idea for introducing polynomial representation is to carry out noise reduction. However, polynomial representation shows insignificant improvement on classification rate, it make us to decide to stick to the raw format.

4.6 Chosen Data Preprocessing – None

So far, no data preprocessing is done, since there are hardly any preprocessing can be done with the raw data.

4.7 Chosen Algorithm – Dynamic Time Warping

4.7.1 Why Dynamic Time Warping is selected?

After studied different algorithms mentioned before, we decided to choose Dynamic Time Warping as our classification algorithm. There are a number of reasons for that:

1. Only one motion data is needed - Since DTW itself is a similarity search algorithm, one perfect motion data will be enough to be used for classification. This is especially useful to achieve user-defined motion function so those users are not required to train the classifier a lot.
2. Similarity value output - Unlike typical data mining method, the result of DTW can be used as an indicator on how similar the motion is performed.
3. Fast Learning - Unlike Neural Network and other data mining methods, to add new motions, we just record the motion into the database.
4. Extensible - DTW is based on the distance between two points, without any limitation on the number of dimensions. Therefore we can apply DTW in 3 dimensional spaces by using Euclidean Distance.
5. Optimization is possible - There are several existing methods to optimize the time and space complexity of this algorithm.

4.7.2 Initial coding – Used Dynamic Time Warping wrongly

When we first applied the DTW algorithm, we did not consider the fact that the accelerations of three axes should not be considered separately. Therefore, in the beginning we applied DTW on each axis and executed DTW three times on the same motion. This is wrong because DTW is talking about the mapping of the points. If DTW is applied separately, most likely the mappings are not consistent among each other, even though the classification rate are quite high.

4.7.3 EXTENSION OF DYNAMIC TIME WARPING

We discovered that we applied DTW wrongly when we tried to implement the algorithm for finding the warping path. As DTW finds the mapping based on the distance, instead of using the distance between two points in one axis, we extended DTW to calculate the Euclidean Distance between two points in 3 dimensional spaces, by using the acceleration value in one time frame as a 3 dimensional coordinate. On average, the extended version of DTW performs much better than the initial coding one.

Below is an example showing the details of the extension:

Time	X Acceleration	Y Acceleration	Z Acceleration
0	1	3	7
15	4	5	2

Table 4.2 Example illustrating the details of extension

Distance between Two Points in X Axis $= |1-4| = 3$
Distance between Two Points in Y Axis $= |3-5| = 2$
Distance between Two Points in Z Axis $= |7-2| = 5$
Distance between Two Points in 3 Dimensional Space
$= \sqrt{(1-4)^2 + (3-5)^2 + (7-2)^2} = 6.16$

The advantage of this algorithm is that it can be easily extensible when more data are available. For example, if Wii Remote expansion port is connected with Nunchuk, three more axes of accelerations will be available. If the two set of accelerations on the two remotes are correlated, based on the principle, since Euclidean Distance is generalized to n-Dimensional Space, we can calculate the Euclidean Distance in 6 Dimensional Space.

4.7.4 TUNING

In order to achieve higher classification rate, different methods are used to tune the Dynamic Time Warping algorithm.

4.7.4.1 FILTERING INPUT

Before we discovered the bug on applying DTW, two methods are used for filtering the input motion data in order to remove noise. The result seems to have no significant improvement. As the DTW algorithms we are using now are in three dimensional spaces, these two filtering methods can no longer be used.

4.7.4.2 DATA TRANSFORMATION

Besides treating the three axes of acceleration values as a Cartesian coordinate, we can transform it into other coordinate system. The below shows the coordinate systems we tried before we discovered the bug on applying DTW. The coordinate is transformed into three angles, and the length of the vector. However, the result isn't very good. Possibly it is because there is "jumping" of values in the angle when some of the coordinate values change their signs. As the DTW algorithms we are using now are in three dimensional spaces, data transformation can no longer be applied.

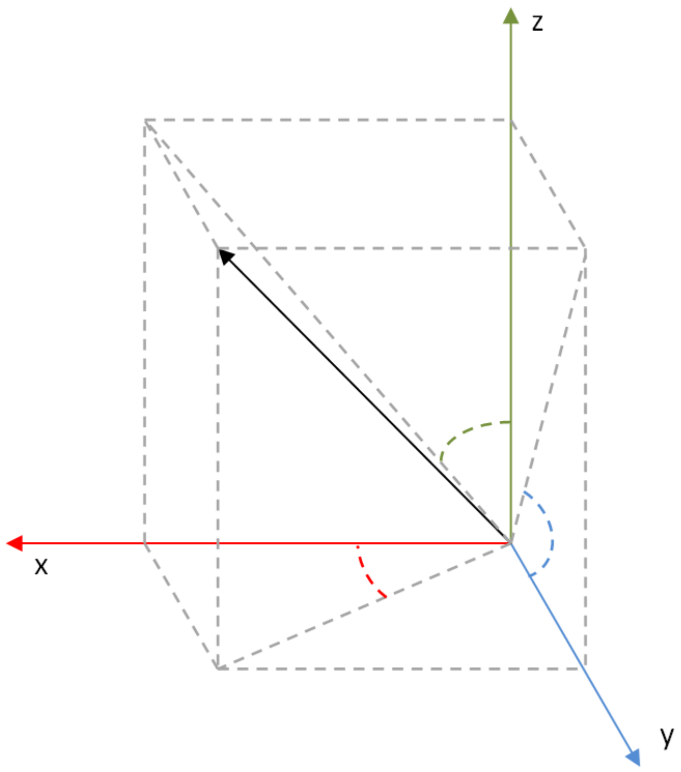


Fig 4.3 The coordinate system we tried before

4.7.4.3 "Model Answer" Tuning

DTW is performed between the to-be-classify motion data and the "model answer". As a result, the quality of the "model answer" will largely affect the classification rate. Therefore, we tried the following in order to minimize this problem. The result of the analysis is shown after this section.

4.7.4.3.1 Number of samples Matching

Typically the to-be-classify motion data is matched with each "model answer" once only. However, we can allow the user to input some more motion data for the same motion as "model answers". Therefore the to-be-classify motion data can be used to compare with other slightly varied "model answers".

4.7.4.3.2 Error Compensation

Besides having more samples for comparisons, we can, on the other hand, take some more motion data from the user, and calculate the average DTW error among those motion data. When doing the classification, the newly computed DTW error can be subtracted by the average DTW errors obtained previously.

4.7.4.3.3 Average of Motion Data

To increase the quality of the "model answer", one of the methods is to take the average of the motion data from the user as "model answer". Several numbers of motion data are obtained from the user. By finding the warping path, we obtain the mapping of each point between different pairs of motion data. For each set of mapped points, we calculate the average coordinate value of that point, and make it as an average motion data.

4.7.5 Analysis Result of Tuning

4.7.5.1 Motion used

Several categories of motions are used. Below is the list of motions:

- Five directions
  - Horizontal Left
  - Diagonal Left
  - Vertical Down
  - Diagonal Right
  - Horizontal Right

- Cursive Writing Letters
    - A, B, C, D, E, G

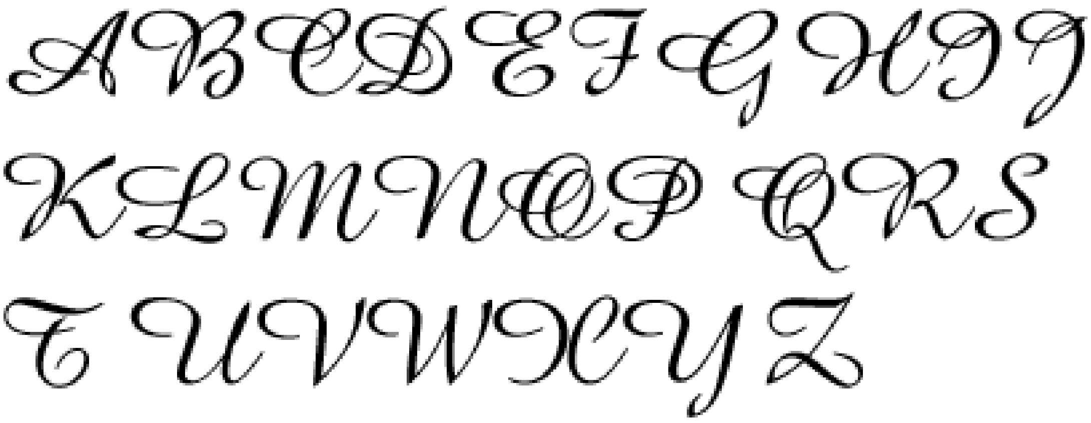


Fig 4.4 Cursive Writing Letters

- Discrete Motions
    - Forward Left Up
    - Forward Right Down
    - Forward Down Left
    - Back Right
    - Back Down Left
    - Forward Right Up
    - Forward Up Left
    - Forward Left Down
- Numbers
    - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- Dao Motions
    - Dao motions are too complicated to be described in text. Below are some photos of how the five Dao motion look like.
        - Motion 1 – Horizontal Left
        - Motion 2 – Top Right Circle
        - Motion 3 – Left Circle
        - Motion 4 – Forward
        - Motion 5 – Vertical Circle

- ***Motion 1 – Horizontal Left***

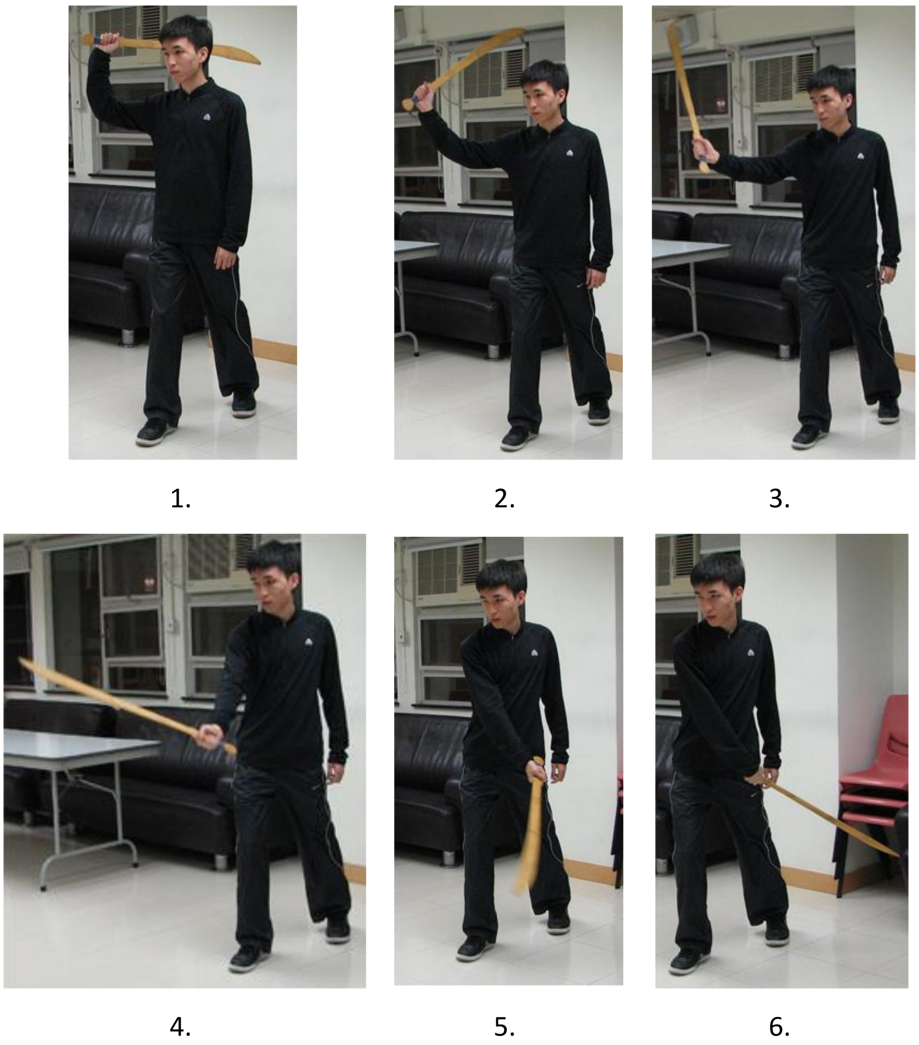


Fig 4.5 Photos illustrating Dao Motion – “Horizontal Left”

- ***Motion 2 – Top Right Circle***

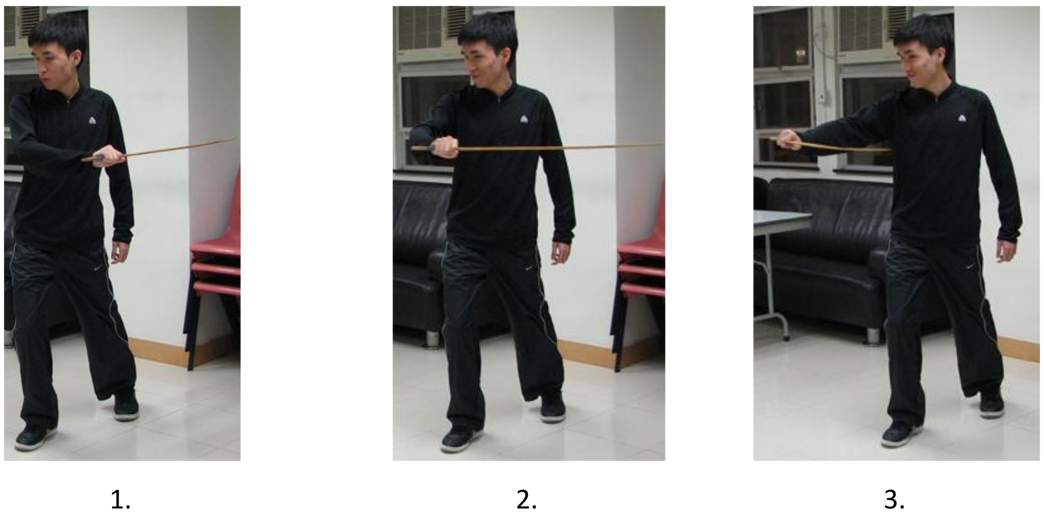
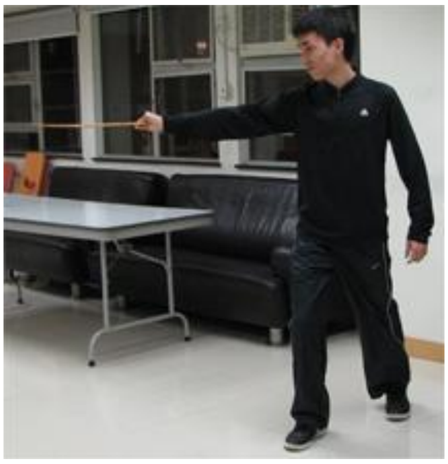
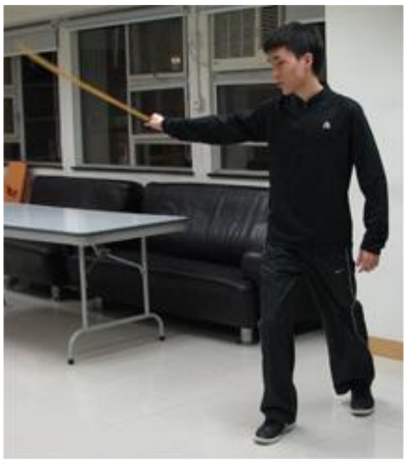
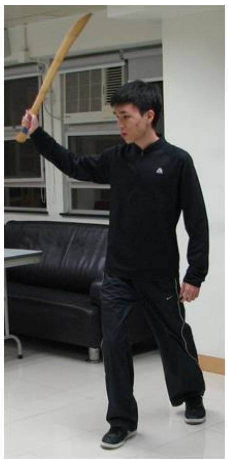
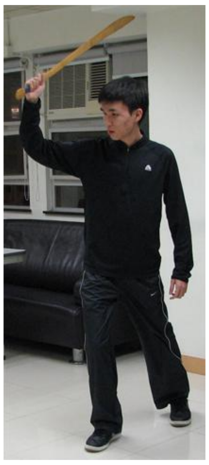


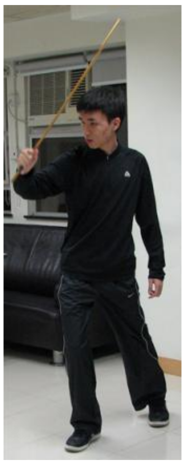
Fig 4.6 Photo illustrating Dao Motion – “Top Right Circle”

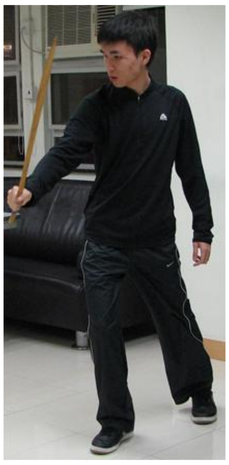

4.

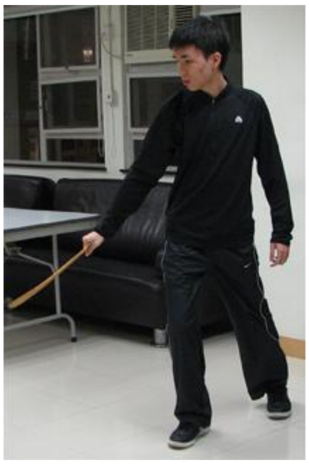

5.


6.


7.


8.


9.


10

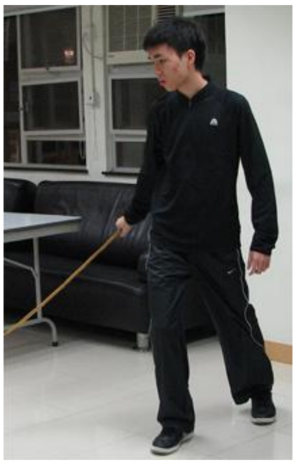
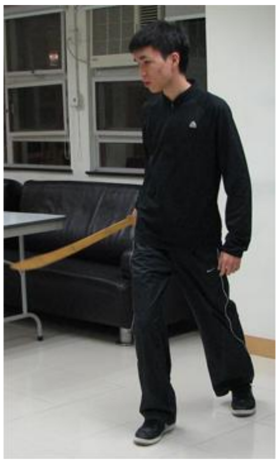
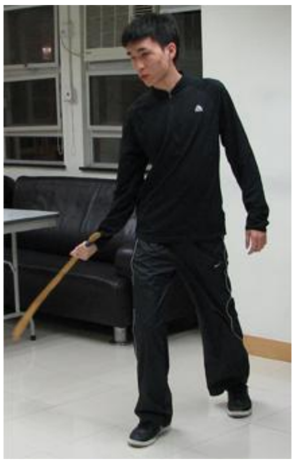
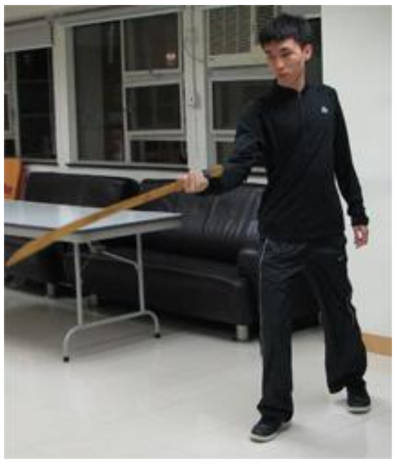

11.

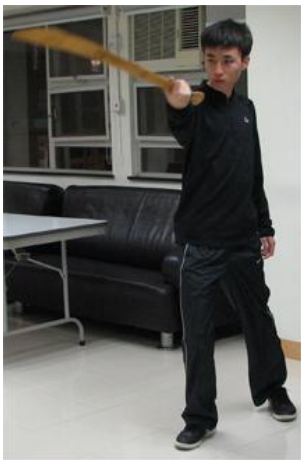
Fig 4.6 Photo illustrating Dao Motion – "Top Right Circle" (cont'd from last page)

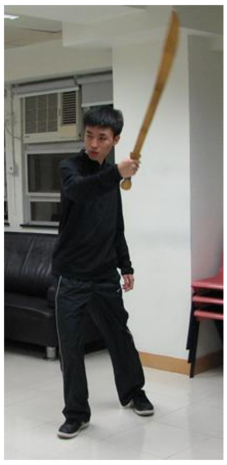
- ***Motion 3 – Left Circle***


1.


2.


3.


4.


5.

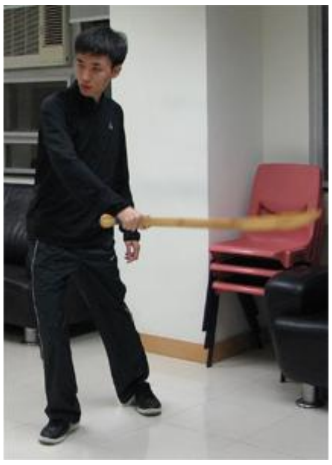
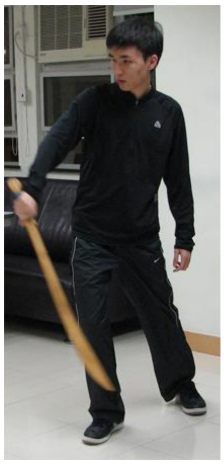
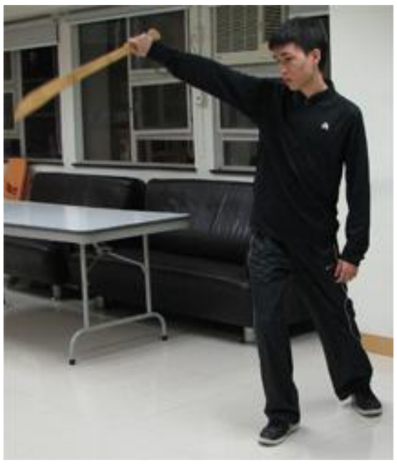
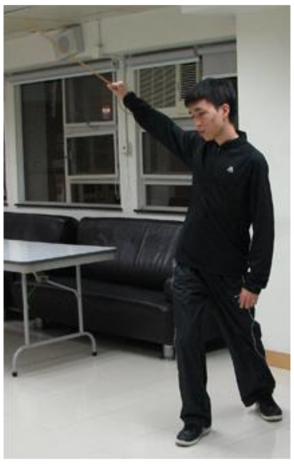

6.

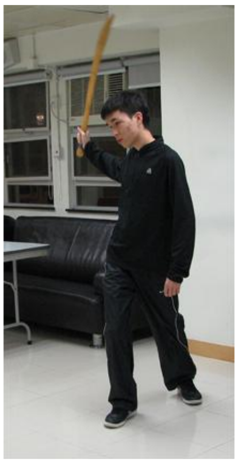
Fig 4.7 Photo illustrating Dao Motion – “Left Circle”

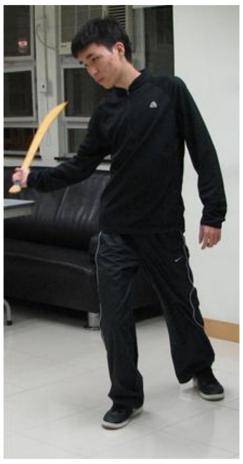

7.

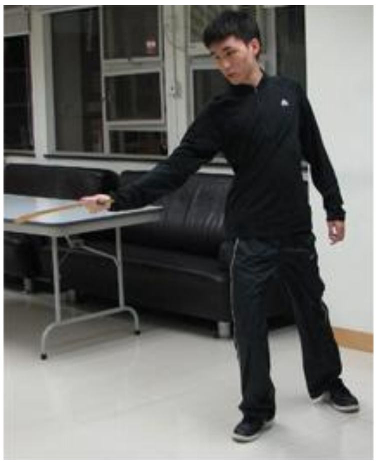

8.

- ***Motion 4 – Forward***


1.


2.


3.


4.

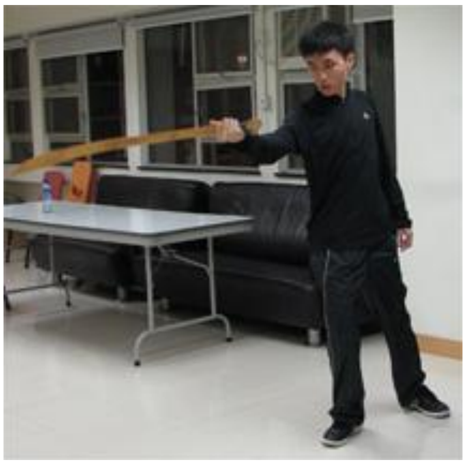
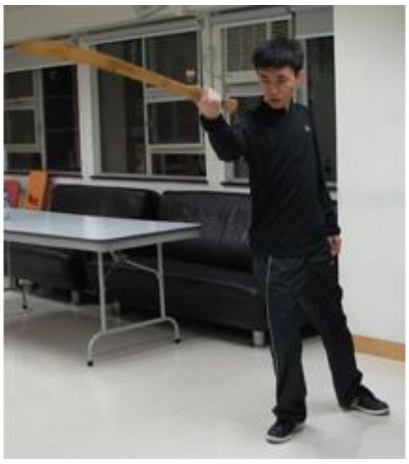
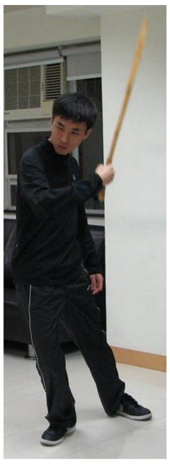
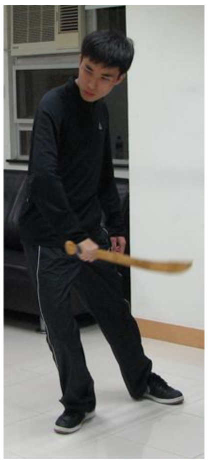

5.

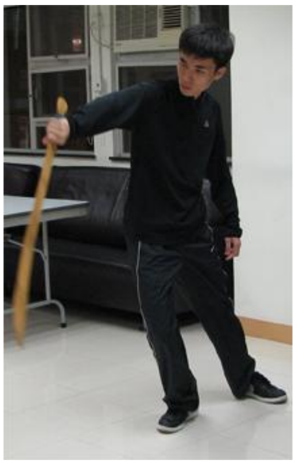
Fig 4.8 Photo illustrating Dao Motion – "Forward"

- ***Motion 5 – Vertical Circle***


1.


2.


3.


4.

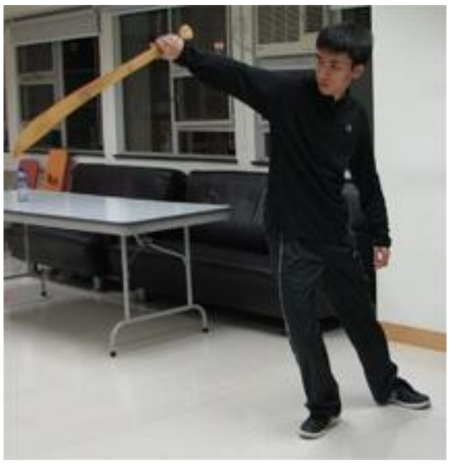

5.

Fig 4.9 Photo illustrating Dao Motion – "Vertical Circle"

4.7.5.2 CONFIGURATION USED

Eight different configurations are used in the analysis. Below is the list of configuration:

- 1 Sample Matching
- 1 Sample Matching, With Error Compensation from 3 Samples
- 3 Samples Matching
- 3 Samples Matching, With Error Compensation from 3 Samples
- 5 Samples Matching
- 5 Samples Matching, With Error Compensation from 3 Samples
- Average of 3 Motion Data
- Average of 3 Motion Data, With Error Compensation from 3 Samples

4.7.5.3 RESULT

Table 4.3 below shows the correct motion classification rate (grouped by motion categories) using different configurations as specified above. Each motion in a category contains at least 30 samples from the involved players.

Motion / Configurations	Five Directions			Letters	Discrete Dao Motion		Numbers			Average
	Player A	Player B	Player C	Player A	Player A	Player C	Player A	Player B	Player C	
1 Sample	95.33 %	91.67 %	90.71 %	100 %	99.51 %	100 %	97.2 %	78.8 %	81.07 %	92.70 %
1 Sample with Error Compensation from 3 Samples	98.67 %	96.67 %	91.43 %	100 %	98.52 %	100 %	96 %	78.8 %	96.43 %	95.17 %
3 Samples	88.67 %	91.67 %	92.86 %	100 %	99.51 %	100 %	97.6 %	88.8 %	100 %	95.46 %
3 Samples with Error Compensation from 3 Samples	92.67 %	91.67 %	91.43 %	100 %	99.51 %	100 %	97.2 %	97.2 %	100 %	96.63 %
5 Samples	92.67 %	94.17 %	95.71 %	100 %	99.51 %	100 %	96 %	95.2 %	100 %	97.03 %
5 Samples with Error Compensation from 3 Samples	94 %	95 %	95.71 %	100 %	99.51 %	100 %	97.2 %	96.8 %	100 %	97.58 %
Average of 3	91.33 %	92.5 %	90.71 %	100 %	99.51 %	100 %	96.8 %	95.6 %	100 %	96.27 %
Average of 3 with Error Compensation from 3 Samples	100 %	90.83 %	87.14 %	100 %	99.51 %	99.29 %	96 %	75.6 %	99.64 %	94.22 %

Table 4.3

4.7.5.4 Conclusion

From table 4.3, we can conclude that, on average, the more the samples, the higher the accuracy, at the expense of time complexity. But there are cases that more samples can lead to a drop in accuracy. One possible reason is that the "model answers" of the newly included samples are of poor quality.

On the other hand, error compensation on average increases the accuracy.

When comparing the results of the two different configurations on average of 3 motion data, we find that applying error compensation will make the accuracy lower. Possibly it is because the error has already been included when doing the average of the data. Therefore applying error compensation is actually repeating and therefore lowers the accuracy.

When we look at different categories of motions, we discovered that motions that are too similar will achieve a lower accuracy.

All in all, the quality of "model answer" and the design of motion largely affect the accuracy. Therefore more motion data are needed to improve the quality of "model answer". On the other hand, depending on the requirements, if accuracy is more important than speed, we can run DTW a few more times on different motion data to further improve the accuracy. Hybrid of the three methods may also be considered.

4.8 OUR DEVELOPED CONSOLE

Besides studying different kinds of algorithm, we also spent a lot time in working our development console as well as the representation of the data and working flow. A good design can largely minimize our time in coding different prototypes and testing time. Code rewrite of the console happened a lot. Below is the menu of our current console:

- Motion Database Operation
    - Register New Motion
    - Display Registered Motions
    - Edit Existing Motion
        - Display First Recorded Motion
        - Re-recorded First Motion
        - Display Second Recorded Motion
        - Re-recorded Second Motion
        - Display Third Recorded Motion
        - Re-recorded Third Motion
        - Display Other Recorded Motion
        - Re-recorded Other Motion
        - Re-recorded Other Motion
    - Unregister Motion
    - Save Database
    - Load Database
    - Back to Main Menu
- Record Motions In Batch
    - Record Motion Sequentially
    - Back to Main Menu
- Classification
    - Benchmark Using Dynamic Time Warping
    - Real Time Using Dynamic Time Warping
    - Benchmark Using Dynamic Time Warping with Specific File
    - Back to Main Menu
- Exit Program

CHAPTER 5 – PROJECT PROGRESS

In this chapter we are going to describe the progress of our project.

Period	Progress
Summer 2007	● Developed a little program to show the variation of accelerations as a hand-on practice on handling data captured from the Wii Remote. ● Got the Wii Remote connected to PC via Bluetooth and Wii Remote API. ● Read some papers about motion classification using accelerometers ● Searched for Wii Remote API for Windows.
September 2007	● Refined our project goal and schedule ● Searched for more information about Wii games and Wii Remote ● Studied different kinds of data representations ● Successfully make the data report in regular interval using threading ● Switched to a better Wii Remote API
October 2007	● First implemented a classifier based on Euclidean Distance Algorithm, ● Implemented a console to generate more than 40 types of attributes in total; served as the input for Weka ● Got familiar with the use of Weka and made use of it to generate source code for a tree-based classifier ● Tried to improve the accuracy of tree-based classifier by - including sample data with different orientations - normalizing amplitudes - discretizing attributes
November 2007	● Implemented classifiers based on Dynamic Time Warping Algorithm, and spent some effort in improving the classification rate following what we have mentioned in section 4.7.4. ● Console rewrote to suit Dynamic Time Warping ● Included benchmark mode in console, which was used to analyze the classification rate on different types of motions ● Worked on the first term report

CHAPTER *6* – DIFFICULTIES ENCOUNTERED

In this chapter we are going to describe the difficulties we encountered in this project so far. There are 2 main difficulties encountered up to this moment – User independent, and how players hold Wii Remote: the orientation.

This chapter comprises the following sections:

- 6.1 Problem on User independent
- 6.2 How player holds Wii Remote: Orientation

6.1 Problem on User independent

As mentioned previously in section 3.3, "user independent" means different players perform the same motion in their own style, for example, performing at different speed, or with different amplitude. However, this leads to the problem on classification.

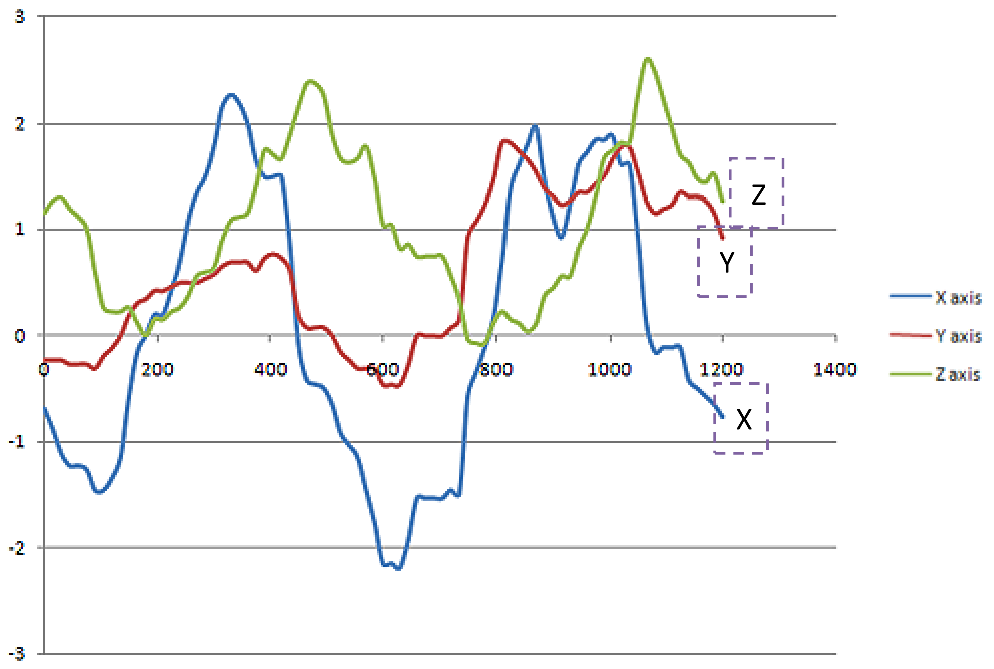


Fig 6.1 (a)

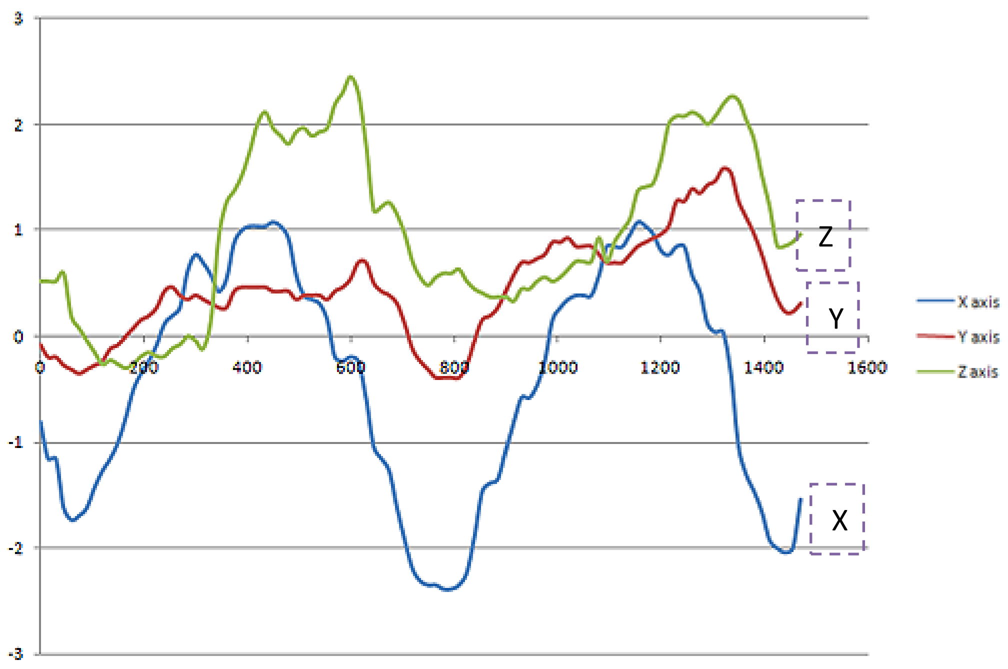


Fig 6.1 (b)

Fig 6.1 shows the acceleration-time graphs in which 2 players perform the same motion – drawing the number "3". Although they perform the same motion but still there are some differences in the changes of accelerations.

In fig 6.1 (a) and (b), it is observed that the changes of accelerations for some axes are quite different, including general shape of the graph, as well as the amplitudes of accelerations. For example:

- For Z axis, the curve shape is generally sharper in (a) than in (b).
- It is also observed that the curve representing the accelerations of x axis in (a) has greater amplitude than that in (b).

Dynamic Time Warping basically performs similarity search on 2 different curves. Although it is observed that the durations of motions, as well as the position of local maximums and minimums of X and Y axes are quite different; still this is solved by Dynamic Time Warping due to its non-linear mapping properties.

However, Dynamic Time Warping itself cannot overcome the 2 differences said above. These 2 differences are just treated as errors in classification. And the motion can either be classified wrongly, or even cannot be classified. These problems are particularly serious in complicated motions like drawing a "3" in this case

It seems to be possible of using tree-based or rule-based classifiers to classify the motions; however, using such classifiers requires a huge amount of sample data from different players. If the amount of sample data is not enough, then a player's style of performing will not be taken into considerations. Another problem is, if there are new sets of sample data added to the classifier, it has to be re-built again.

6.2 How player holds the Wii Remote: Orientation

Another difficulties facing is the orientation. Orientation refers to how Wii Remote is held, whether it is held upright, slightly rotated, or rotated greatly.

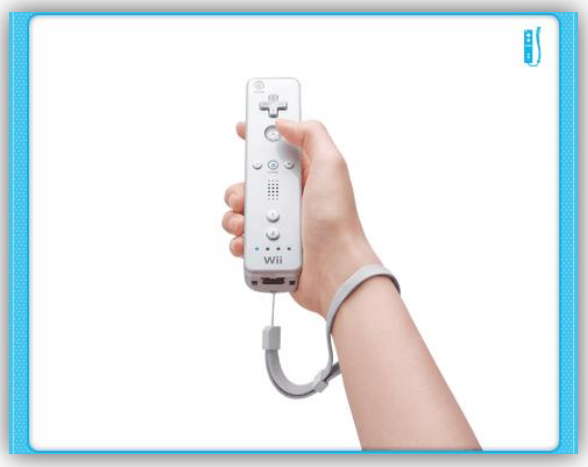


Fig 6.2 A player is holding Wii Remote upright.

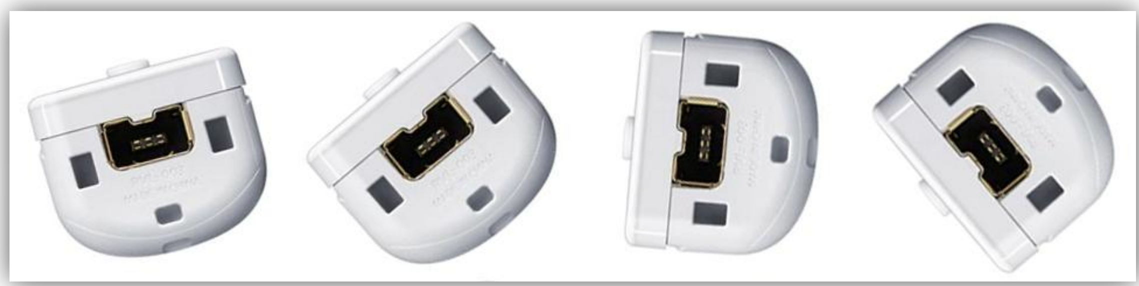


Fig 6.3 Some other possible orientations of holding a Wii Remote (back view)

We noted that if the Wii Remote is holding with different orientations, no matter if the remote is at rest or moving, the recorded accelerations are different as well.

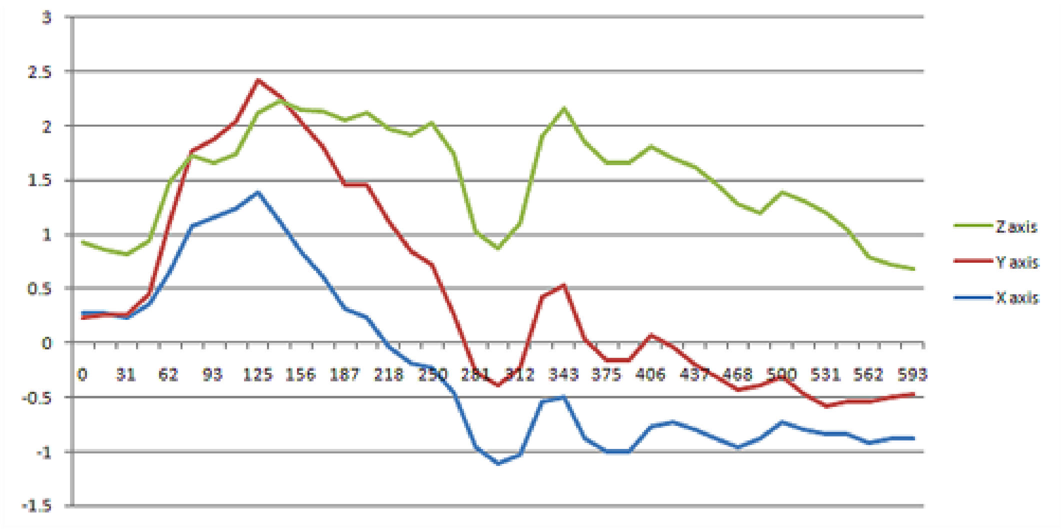


Fig. 6.4 (a)

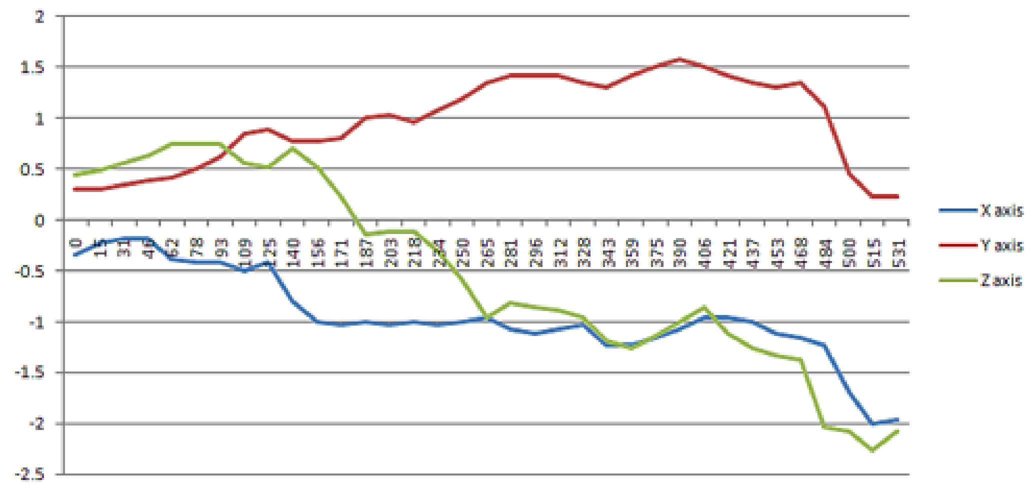


Fig 6.4 (b)

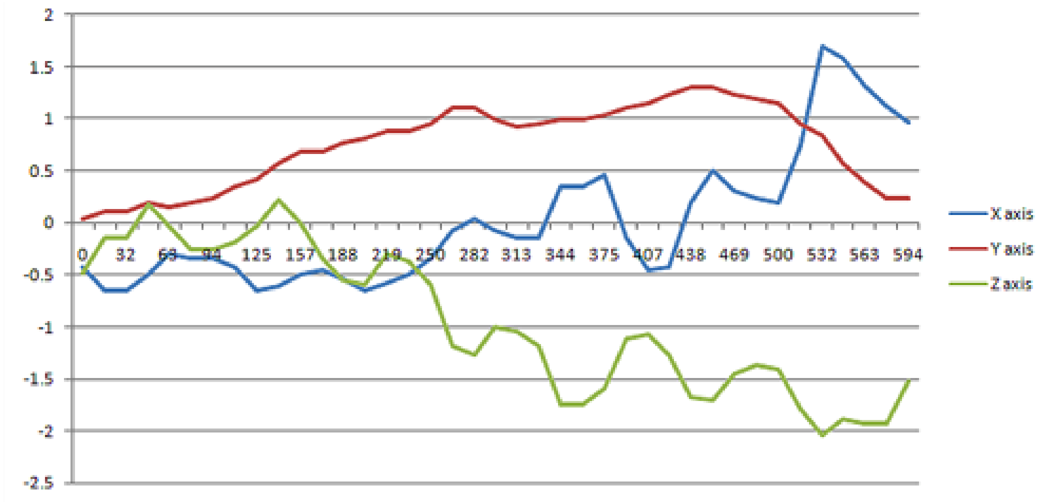


Fig 6.4 (c)

Fig 6.4 (a), (b) and (c) show the acceleration-time graph of the same motion done by the same player, with 3 different orientations of holding Wii Remote. It is obvious that there are some differences between the curves of accelerations. The differences are summarized as follows:

- Consider X axis. It is observed in graph (c) the acceleration values generally increase; whereas in graph (b), the acceleration values generally decrease; still in graph (a), the acceleration values first reach a global maximum, and then fall down. With 3 different orientations, the trend of the change in acceleration values is completely different from each other.
- Consider Y axis. Although graphs (b) and (c) show similar shape, still in graph (a), the acceleration values attain the maximum at the beginning, and then keep decreasing; in graphs (b) and (c), both of them have the acceleration values decrease only towards the end. Similar observations also exist for Z axis.

With the significant differences in accelerations, Dynamic Time Warping will just treat them as errors. Thus the errors are huge and resulting in incorrect classifications.

We have also tried including sample data of different orientations in both tree-based / rule-based classifiers, as well as Dynamic Time Warping. In tree-based / rule-based classifiers, some motions are classified wrongly still. As in Dynamic Time Warping, including those sample data can help, but it fails again once there are little variations in the orientation. Even though it is correctly classified, the error rate is still quite high.

CHAPTER 7 – CURRENT LIMITATIONS

In this chapter we are going to describe some limitations of the research that we have at this moment.

This chapter comprises the following sections:

7.1 Offline Recognition

Currently the classification is done once the motion is performed completely. When the motion duration is long, users will not receive any feedback until the end of the motion.

7.2 Button Required

Currently, in order to indicate the start and the end of motion, users are required to press a button on Wii Remote while performing the motions.

7.3 Fixed Orientations

As mentioned in section 6.2, the orientation of the Wii Remote will affect the pattern of the accelerations data. One possible solution is to apply transformation on the data. However, being not able to obtain the direction of the gravity, it is impossible to know what transformation we should apply. Therefore, in our current classification algorithm, we assumed the orientation is fixed.

CHAPTER *8* – FUTURE WORK

In this chapter we are going to give a preview on what we will try to do in the coming semester.

This chapter comprises the following sections:

8.1 GAME DEMO FEATURING OUR CLASSIFIER

In this semester, we spent most of the time in studying how to obtain data from Wii Remote, and also the classifier. However, we did not have any applications using our classifier. Therefore in the next semester we will have a game demo featuring our classifier.

8.2 USER DEPENDENT = EDUCATIONAL

Dynamic Time Warping algorithm is one of the similarity search algorithm. If two motions are identical, then the DTW error will be zero.

Having this fact, although it is a disadvantage that the motion cannot be different a lot from the "model answer", it is an advantage for the user to learn how to perform the motion exactly the same as the "model answer". In practicing Chinese Martial Art, the basic of it is to perform the motion smoothly and correctly. By using DTW algorithm, one can able to practice Martial Art by using Wii Remote. The correctness of the motion can be sent back to the user by using the DTW error calculated. Further extension of the algorithm can provide more useful information such as in which period of time the motion is perform badly. Thus, a gaming controller can be turned into an educational tool.

8.3 MOTION FINITE STATE MACHINE

The more the different types of motions, the more difficult for the classifier to classify the motions correctly. Therefore, by applying some kind of game rules, we can minimize the number of possible candidates in classification, and yet preserve the fun when playing the game. The idea is to limit the type of motions that can be followed by one type of motions. For example, at first, you are only allowed to write "A", "C", "E". When "A" is wrote, the next letters that can be write will be "D", "E", "F". In this way, a motion finite state machine is built, in which each node represent a state, and the edge represent the type of motions that accept in that state. This allows the player to perform their own style of motions in any sequences they like and yet maintain the high accuracy of classification.

8.4 EARLY RECOGNITION MATCHING

One of the limitations of our classifier is offline recognition. Luckily, it is possible to further extend the algorithm in order to achieve on-the-fly recognition. The basic idea is to execute DTW algorithm while the frame data are being recording. Thus early recognition can be achieved.

8.5 TWO HAND WEAPONS WITH NUNCHUK

So far we are only focusing on using Wii Remote only, without any research in Nunchuk. It is possible to include two hand weapons like Qiang with the help of Nunchuk.

8.6 "BUTTON-LESS"

Another limitation of our classifier is that we require the user to press the button. Yet, there is possibility to determine the start and the end of motion by looking at the overall acceleration on Wii Remote.

CHAPTER 9 – REFERENCE

[1]: “*The Games Industry: Past, Present & Future*”, http://www.gamesinvestor.com/Research/History/history.htm

[2]: “*The Big Ideas Behind Nintendo's Wii*” - BusinessWeek November 16 2006, http://www.businessweek.com/technology/content/nov2006/tc20061116_750580.htm

[3]:“*Breaking: Nintendo Announces New Revolution Name - 'Wii'.*”, Gamasutra. http://www.gamasutra.com/php-bin/news_index.php?story=9075

[4]: “*Wii Remote*”, http://en.wikipedia.org/wiki/Wii_Remote

[5]: “*- WiiYourself! - gl.tter's native C++ Wiimote library.*” http://wiiyourself.gl.tter.org/

[6]: “*LiveMove Pro Director's Cut Manual*” http://www.ailive.net/papers/directorsManual_en.pdf

[7]: Ian H. Witten and Eibe Frank (2005) "*Data Mining: Practical machine learning tools and techniques*", 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[8]: “*Euclidean Distance*”, http://en.wikipedia.org/wiki/Euclidean_distance

[9]: “*Dynamic Time Warping*”, http://en.wikipedia.org/wiki/Dynamic_time_warping

[10]: “*11.2. Dynamic Time Warping*”, http://www.ics.mq.edu.au/~cassidy/comp449/html/ch11s02.html

[11]: “*Exact Indexing of Dynamic Time Warping*”, http://www.cs.ucr.edu/~eamonn/vldb_keogh_2002.ppt

[12]: “*Wii Remote*”, http://www.cse.cuhk.edu.hk/~csc3510/notes/3510_lecture11_Wii_Controller.pdf

[13]: Eamonn Keogh*, Selina Chu, David Hart, and Michael Pazzani (2001): “*An Online Algorithm for Segmenting Time Series*”, http://www-scf.usc.edu/~selinach/segmentation-slides.pdf

[14]: “*Normalization* “, http://en.wikipedia.org/wiki/Normalization

[15]: Hagit Shatkay (1995) : “*The Fourier Transform – A Primer*”, http://www.phys.hawaii.edu/~jgl/p274/fourier_intro_Shatkay.pdf

[16]: “*Cooley-Tukey FFT algorithm* “, http://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm

[17]: “*Dao (sword)*”, http://en.wikipedia.org/wiki/Dao_(sword)