Smart Activity Monitor

Wan Luo B.Sc.

A thesis submitted to the
University of Dublin, Trinity College,
in fulfilment of the requirements for the degree of
Master of Science in Ubiquitous Computing

2006

DECLARATION

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this or any other University, and that, unless otherwise stated, it is entirely my own work.

_____

Wan Luo B.Sc.

September 2006

PERMISSION TO LEND AND/OR COPY

I, the undersigned, agree that the Trinity College Library may lend and/or copy this thesis upon request.

_____

Wan Luo B.Sc.

September 2006

ACKNOWLEDGEMENTS

"I would like to take this opportunity to thank the following people for their help and support :

Many thanks to Dr. Gavin Doherty for suggesting such an interesting subject matter, and all his help and advice throughout the course of this work.

Thanks also to Joe McKnight, who helps me on the MT9 setting up at the early stage.

I would also like to express my sincere gratitude to the 10 volunteers for their time and effort in helping me collect out data. Special thanks to Huijuan Tian for assisting me on administering data collection."

Wan Luo

University of Dublin , Trinity College

September, 2006

ABSTRACT

There are over 1 billion overweight adults globally, at least 300 million of them obese [18]. Overweight and obesity lead to adverse metabolic effects on blood pressure, cholesterol, triglycerides and insulin resistance. It is a fast growing health problem, especially in the developed countries. The most healthy method to control our weight is balancing the energy consumption and energy expenditure. The energy consumption is about the eating and drinking. Today, most food and drink we purchased from supermarket have nutrition information displayed on their packets. Therefore, it is very easy for us to monitor the energy consumption. Letting people know their energy expenditure on different activities is the motivation of this project.

The main purpose of this dissertation is designing and implementing a smart activity monitor, which can detect the user's activities as well as calculating the energy expenditure on that activity. The activity detection is the most challenging part of this project. There are many solutions have been proposed from motion tracking to motion detection. In this project I am using a single motion sensor "Inside-In" technique [1]. The sensor is tied beside the user's ankle in order to collect the motion data. The activity recognition is implemented by statistical pattern recognition with an unsupervised classifier. It can recognise 3 different leg-only activities (walking, running and cycling) with over 88% accuracy. The algorithm has been tested across 10 different people, with 80 different data sets and over 1800 different tests. The activity detection system is practical, reliable and can be adapted to many different context awareness application, which requires the user' current activity information. The whole system has been fully deployed on a handheld. The sensor is connected to a handheld, which is used to process the data as well as for the user interface via a serial interface.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

According to the data from WHO (World Health Organization) [18], there are over 1 billion overweight adults globally, at least 300 million of them obese. The number is increasing very fast. With current trends, the number will be doubled in 20 years time. Are you one of the members? This is very easy to assess using body mass index (BMI).

$$\text{BMI} = \frac{Weight(Kg)}{Height^2(m^2)}$$

A BMI over 25kg/m2 is defined as overweight, and a BMI of over 30kg/m2 as obese.

Overweight does not only affect people's shape. It will also damage our health. Overweight and obesity lead to adverse metabolic effects on blood pressure, cholesterol, triglycerides and insulin resistance. It is more likely to get Type 2 diabetes and hypertension for overweight people. Approximately 85% of people with diabetes are type 2, and of these, 90% are obese or overweight. High BMI also increases the risks of cancer of the breast, colon, prostrate, endometroium, kidney and gallbladder.

Obesity is also one of the fastest growing health problems in Ireland [19]. According to the SLÁN survey in 2003, there are 47% Irish people reported being overweight or obesity. Ireland has the fourth highest prevalence of overweight and obesity in men in the EU and the seventh highest prevalence among women. One in eight Irish people are obese and every second person is overweight. There are at least 2,500 people dies coursed by obesity in Ireland every year.

1.1 Motivation

From the facts introduced above we can see, overweight is a series problem. It should be attached important to every people in the world. People who are not overweight should prevent from obesity and keep weight. People who are overweight or obesity should begin to fight fat and fight to be fit.

How to keep weight or loss weight? Basically, we should eat and drink healthier as well as doing more physical activities. If the energy we consumed is much greater than the energy we expended, then time after time our weight will be increased. How to balance the consumption and energy expenditure is the key to control our weight.

Today most food and drink we bought from supermarket has its nutrition information on its packet, e.g. a bar of chocolate, a can of Coke. Figure 1-1 is an example of nutrition information on a food product's packet. It is very easy for us to know how much energy we have consumed. The problem is it is difficult to know how much energy we have expended on physical activities. If the energy we expend on physical activities is too little, we are still facing the overweight problem. However, if we expend too much energy on physical activities, it is also bad for our health.

Figure 1-1 Example of Nutrition Info.

The motivation of this project is designing a smart activity monitor that can monitor the user's physical activities and calculate the energy expenditure.

There are many energy expenditure monitoring equipments has been built in the physiology science area. Those equipments are designed for physiology researches. Most of them are either too big or too expensive or even both. This project is

expected to using motion tracking techniques to detect activities and using current available energy expenditure equations from exercise physiology to calculate the energy expenditure. Today the most motion tracking researches are in vision area. However, there is big disadvantage by using vision method to detect activity in this case. We are monitoring the user's physical activity here. It is not convenient to have a camera to monitor the user all the time. Another motivation of this project is researching in non-vision based human motion tracking.

Beyond the scope of exercise physiology, the activity detector can provides very important context information for varies of context awareness applications. Once the activity detector is successfully implemented, it can be embedded in any application that needs to know "WHAT physical activity is its user currently doing?"

1.2 Project Objectives

The primary objective of this project is to develop a human lower body physical activity monitor. The lower body physical activity includes walking, running and cycling. The goal for this project is accurately detecting the activity as well as correctly calculating the energy expenditure on that activity.

In order to detect the human activity, I need to do research in motion senor and human activity recognition algorithms. Once the activity detector is developed, it should be evaluated against its accuracy. This part of project should be independent from the application level, including energy expenditure calculating. In this case, the activity detector can be reused in other context awareness applications. I also need to do research in the exercise physiology in order to find out how do calculate the energy expenditure.

Considering the usability of the application, the project should be implemented on a handheld. Therefore developing an application on a handheld will also be an challenge to this project.

1.3 Report Overview

The remainder of the report shall follow the following structure:

Chapter 1

This chapter provides a brief overview of the project, stating the motivation and setting up the objectives.

Chapter 2

This chapter documents the background research which was completed, in order to understand the needs and requirements of this work. The background chapter will discuss some popular motion tracking methods as well as some projects of using the motion tracking sensors.

Chapter 3

This is the design chapter. There is a comprehensive description of design process documented in this chapter. The design process includes three main stages, activity detection stage, energy expenditure stage and application architecture stage.

Chapter 4

The chapter will introduce the technologies that will be used in this project. All the hardware including motion sensor and the handheld as well as the software and tools will be introduced.

Chapter 5

This chapter documents how the smart activity monitor is implemented. The steps of implementation follow the sequence of design. The detail of activity recognition algorithm, energy expenditure calculation as well as some problems solving will be documented in this chapter.

Chapter 6

In this chapter the accuracy of the activity recognition will be evaluated. A series of tests will be carried out on the specifically designed testing bed. The design of

testing bed as well as the tests results will be covered in this chapter. Finally I will give the comments according to the tests results.

Chapter 7

In this chapter I will give the conclusion of the entire project, and I will also talk about the future work of this project.

Appendix A

This appendix contains screenshots of user interface from the Pocket PC.

Appendix B

This appendix contains a full bibliography pertaining to this dissertation.

Chapter 2: Background

The chapter provides background information related to human movement tracking and recognition. The current available techniques will be discussed as well as evaluated against the objectives and scope of this project. The projects that I came across which is using the MT9 will also be discussed in this chapter.

2.1 A Review on Human Movement Tracking

With current technologies, there are three types of human movement tracking systems available now [1]. The first system is the "Outside-In" system. Such system uses one or multiple external sensors to exam the data source from human body. The second system is the "Inside-Out" system. The system uses one or multiple sensors fit on human body to detect the external artificial source. The last type of system is called "Inside-In" system. This system uses one or multiple sensors fit on human body to detect the data source also on the person's body. For example, Figure 2-1 illustrates the 3 types of human movement tracking system in one picture.

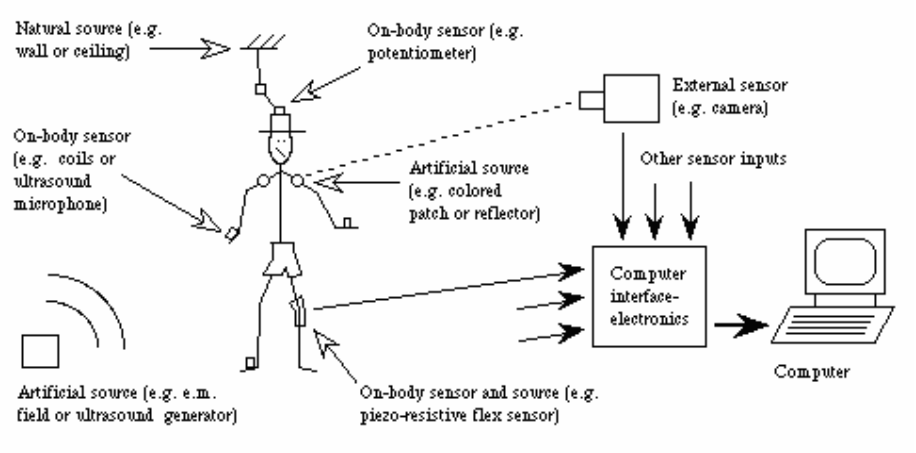


Figure 2-1 Illustration of a real human movement tracking system [1]

The “Inside-Out” system is the only system that using the external data source. Therefore the accuracy of such system for tracking the actual human activity is low. The common examples of this system would be GPS and Ultrasound systems. These systems can track the position and speed, but difficult to recognize the actual activity. Therefore, this type of system will not be reviewed in this project.

2.1.1 Vision Based Tracking

The typical “Outside-In” system is vision-based system. As it is called, vision based system relies on one or multiple optical sensors (camera). The vision based tracking system can be classified as “With-Marker” and “Without-Marker” [2]. The “With-Marker” system has been used for over 30 years. It was invented by G. Johansson in his physiological experiment. He attached small reflective markers to human body, which allow these markers to be monitored during trajectories in order to perceive human motion. This experiment is called “Moving Light Display (MLD)” [3]. Even until now, this technique is still very popular in movement tracking area. If you watch any making of latest movies or games, you can often see people wearing many small reflective balls on his/her body while making the movie. It is a milestone solution of tracking human movement.

Figure 2-2 Qualisys system [6]

There are many existing products available in market. Qualisys is one of these systems. Figure 2-2 is an example of Qualisys system. We can see there are many cameras in the scene. The Qualisys system consists of 1 to 16 cameras, each emitting a beam of infrared light. The man standing in the middle of picture above wears many white balls. These white balls are the reflective markers. These small reflective markers will reflect the infrared light emitted from the cameras back to the cameras. The camera then will measure distance in order to calculate a 2-dimensional position of the reflective target. By combining the 2-D data from several cameras a 3D position will be calculated. And then the 3-D human movement motion can be simulated by the application. The activity type can also be recognized very easily. However, if we want to use such technique to design an activity monitor, which can be used by every body, that is very difficult. Not because of the accuracy of such system will be low, but the cost and the condition of using this system are both very high. Most the vision based tracking systems with marker are implemented for in door user. Therefore we can only use such system to monitor the activity in gym. Before the system can be used, it requires the professional people to install the cameras and calibrate them. The number of cameras will affect the accuracy of the result as well as the cost. Another bad news for user is he/she has to wear many small markers on his/her body. Although we can find some system with improved usability like CODA [5], which uses pre-calibrated sensors and intrinsic marker IDs to increase the mobility and the reduce the time cost in setting up and operating, this type of system still doesn't meet the need of smart activity monitor in ubiquitous computing.

The vision based without marker system is a less restrictive motion capture technique [2]. Such system only concerns the boundaries and features of human body. This is a very active research area as there are many unsolved problems. The researchers from Sejong University, Korea, used this technique to develop a ubiquitous smart home [7]. Inside the smart home, the location and motion of human can be tracked as well as recognized in real-time. They use 3 images to analyze the human's location. The first image is the empty home image. In the second image, all the furniture and appliances are put in, but without human. The last image has everything including human. In this case, the location of human can be analyzed according to these three images as well as which furniture the people is associated.

Figure 2-3 shows the structure of camera argument. And then they use the support vector machine to predicate the human's motion.

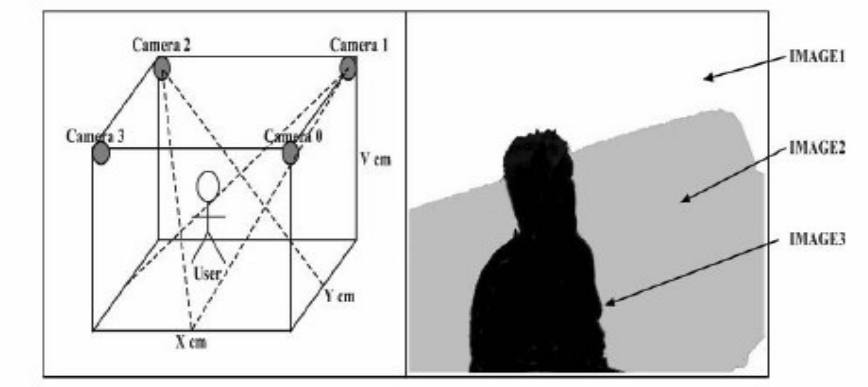
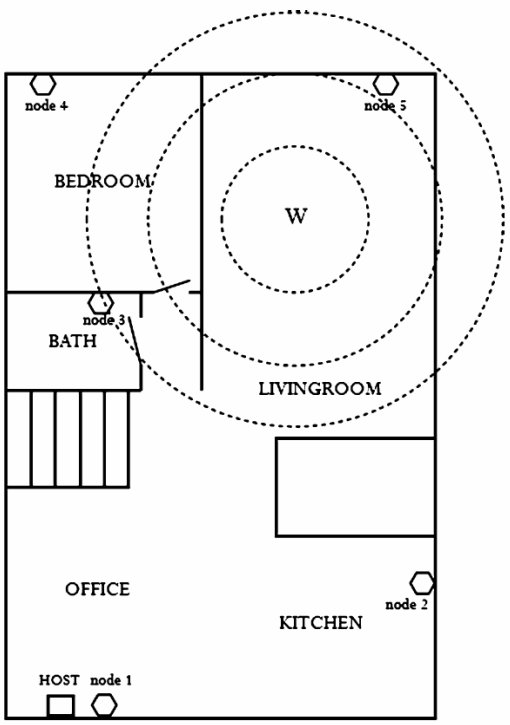


Figure 2-3 Structure of Ubiquitous Smart Home [7]

The "Without Marker" based system is not suitable to use in this project as it can only predicate the activity. The recognition accuracy of "With Marker" based system is much higher, because human skeleton is a highly articulated structure. If the markers are put in right place, it can perceive human motion precisely [3]. However, in the subject of mobility and pervasive, the vision based techniques are not quite suitable for design a daily use smart activity monitor.

2.1.2 Non-vision Based Tracking

The "Outside-In" system can also be non-vision based. Thomas Dowad [8] from Before Technology, Canada, presented a Personal Action Wireless Sensor (PAWS) system. This system is also used in the area of smart space in order to collect the important context information. Instead of using cameras, it uses motion sensors within a wireless sensor network. The user wears a watch-like sensor. The sensor has an accelerometer built-in. There are many locators are embedded in the different, which are the communication device. The sensor will use Zigbee to communicate with the locator. Figure 2-4 is an example of PAWS system. The 'W' here means the user with the sensor. The 'node' means the locator. While the sensor is broadcasting a signal, the locator will tell the strength of the signal. Base on this



information the location of the user can be estimated. According to the location and the current time, the probability of different action can be calculated. The motion sensor will collect the motion data such as velocity, duration and repetition period. Finally a host computer will recognize the action based upon motion, location and time.

Figure 2-4 PAWS System [8]

Researchers from University of California [9] designed a similar system. However, they used simple common sensors instead. They sensors they used include door close sensors, IR motion sensors and floor pressure pads. They also present a behavior model for predicting future sensor outputs and user location from previous data. In conclusion of this type of non-vision human movement tracking system, it is a future technology. If the pervasive sensors are embedded everywhere in our living environment in the future, our activity can be easily tracked. However, this is not the best approach to implement an activity monitor to track the activity of user him/herself.

The last human movement tracking technique I want to review is the “Inside-In” system. The idea of this technique is using one or multiple wearable sensors to collect the source data from the user. Nicky Kern and his colleagues from ETH Zurich, Switzerland [11] presented us a hardware platform which consists of multiple sensors and portable computer. Figure 2-5 shows the user who has the sensor tied on

his body and holding a PDA. The PDA is used for annotation. We can see he has a bag. There is a laptop inside of the bag, which is used to record and process the incoming data. The sensors they used for the project are the acceleration sensors, because of their weight, size and cost. The system can store up to 48 hours 3 dimensional acceleration data. The activity will also be recognized as well as recorded. They also point out that if we only want to recognize the leg-only activity, sensors should be placed on the leg. J. Baek from Kyungpook National University, Korea [10] designed an activity estimator that can classify the acceleration data into 8 different states. The states include standing, sitting, lying back, lying on, walking, running, upstairs, and downstairs. Figure 2-6 shows the signal process steps of motion estimation.

Figure 2-5 Multiple-sensors activity detection system [11]

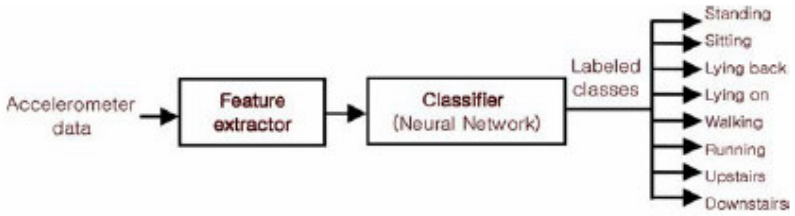


Figure 2-6 Signal processing steps for motion state estimation [10]

The "Inside-In" system can also use other different sensors. J. Lester from University of Washington [12] designed a single sensor board which contains 7 different sensors. The sensor board they designed is very small and light, which can be embedded into a mobile phone. Once the user carried the mobile phone, his/her activity will be monitored.

After the study of these activity detection techniques I found that the "Inside-In" system is the most suitable technique that can be used in my project. The smart activity monitor is designed to monitor a single user's physical activities. The "Outside-In" approach is facing mobility and installation requirement problems. The vision-base approach is also facing cost of facility and cost of computation problems. Therefore I decide to use choose a wearable sensor to collect the motion data from the user in order to monitor his/her activities. The scope and plan of this project does not allow me to design a unique sensor board. From Dr. Madz Haahr's "vision of ubiquitous computing" lecture, I meet the MT9. The MT9 is a digital measurement unit that measures 3-D rate-of-turn, acceleration and earth-magnetic field. The detailed introduction of the MT9 can be found in Chapter 4. My project will this sensor to collect the motion data from the user.

2.2 Prior MT9 Projects

In this section I will introduce two projects that are using the MT9. The purpose of studying these two projects is to make myself familiar with the sensor. The first project I know, which is using MT9, is from Dr. Madz Haahr's lecture. It is called "Smart Sword" [13]. The sword here is a Japanese bamboo sword, which is used for Kendo. The MT9 is tied with the bamboo sword, and it tracks the motion of the sword in 3-D. They created a visualization application that can simulate the movement of the sword according the MT9 reading. Figure 2-7(a) shows the user holding the smart sword. Figure 2-7(b) shows the visualization application that is simulating the sword movement.

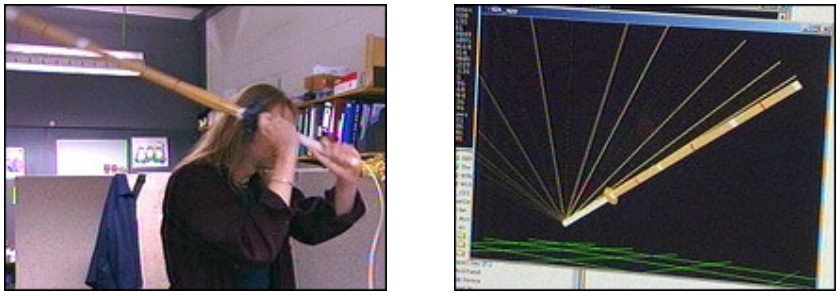


Figure 2-7 (a) User Holding the Smart Sword (b) Visualization Application [15]

They also created a training application that can record the professional kendo player's move in order to train the beginners.

Another project I came across is from my classmates Joe McKnight and Eoin Meehan's pilot project [14]. They The project is a gesture controlled TV remote, which was implemented last year. The project was using the MT9 in the area of gesture recognition. They designed the system that allows the user to hold the MT9 in his/her hand and doing the predefined gestures. The motion data that collected from the gesture will transfer to a PDA. The PDA will evaluate the gesture and using the built-in IR to control the TV.

Chapter 3: Design

The design of the project has 3 main stages. The first stage is the activity detection stage. In this stage, a solution must be designed in order to recognize different activities. The second stage is energy calculation stage. How to calculate the energy expenditure must be designed in the second stage. The final stage is the application stage. In this stage, the application architecture must be designed.

3.1 Activity Detection Stage

3.1.1 Goal

The goal of this stage is to design an appropriate and feasible solution to deal with the incoming MT9 data in order to detect different activities (Walking, Running and Cycling) in real time. This solution should use as less resource as possible but making potentially highest accuracy.

3.1.2 First Approach

MT9 Data Analysis

From the observation of people's walking, running and cycling, a truth is discovered. The rotation of human's legs and feet is only based on one axis with the human's own coordination system. From Figure 3-1 we can see, the rotation of the leg should only base on the Z axis, when the people is walking, running or cycling. In the case of rotation on Y axis, this could indicate the person is turning around. If the leg

rotates on X axis, the person might tumble. The goal of this stage is about finding out a method to detect different activities. Therefore the rest information including if the person is turning around or tumbling is not interested in this project.

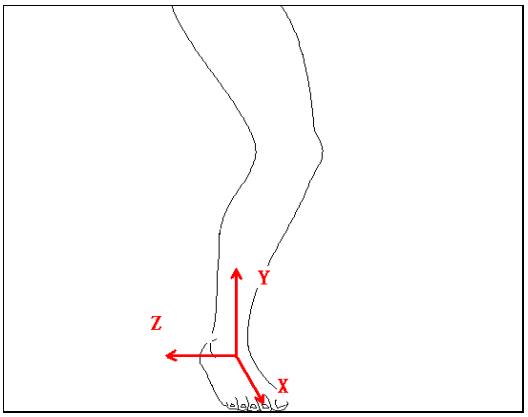


Figure 3-1 Human Leg's Coordination System

MT9 is a sensor that can calculate absolutely orientation in 3D. It is the main tool that is used in this project. The technique detail of the sensor will be introduced in the Chapter 4. The sensor's output can be presented in Euler Angles, which are pitch, roll and yaw. These three terms can be easily explained by the diagram below (See Figure 3-2). If we imagine the airplane as a human foot, the only rotation during walking, running and cycling is on pitch.

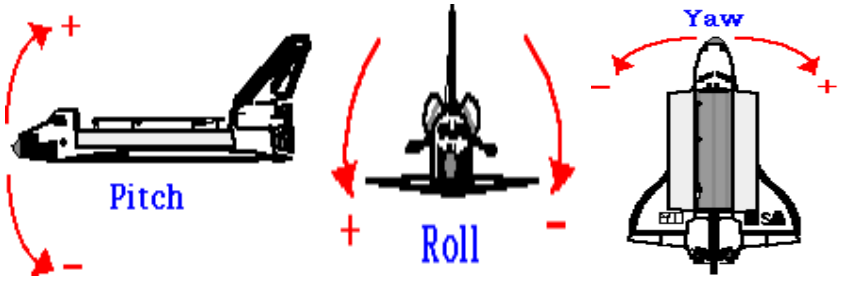


Figure 3-2 Pitch, Roll, Yaw Systems [22]

To design a solution that can detect different activities from the MT9 data, the first thing is studying on the MT9 data's characteristic on that particular activity. The MT9 sensor manufacturer Xsens Motion Technologies provides a simple software (See Figure 3-3). This software can connect the MT9 senor and out the sensed data in text files. The detailed introduction of the software can be found in Chapter 4.

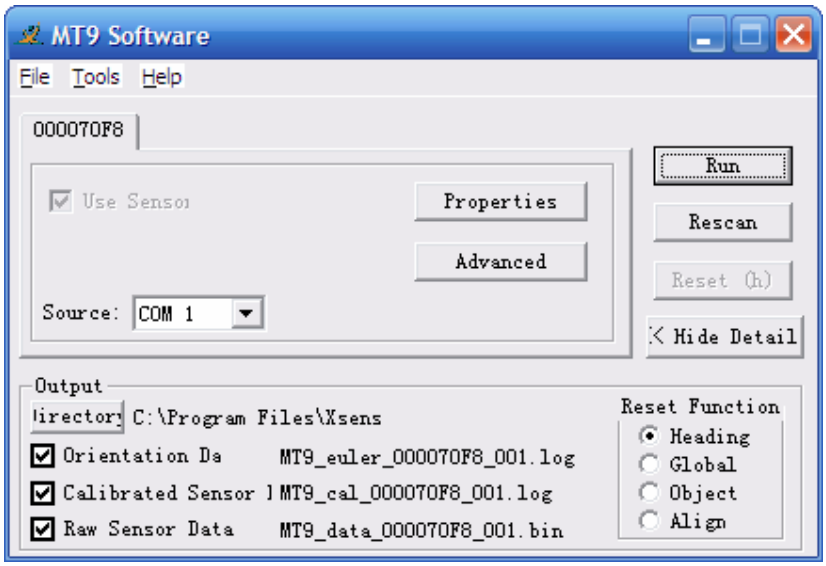


Figure 3-3 Screenshot of MT9 Software

After the MT9 software creates the two text files, one is orientation data another contains acceleration data, the data is graphed by using Microsoft Excel. Because the investigation begins with differentia walking and running, so I only record the data sets for walking and running at this stage (See Figure 3-4, 3-5, 3-6).

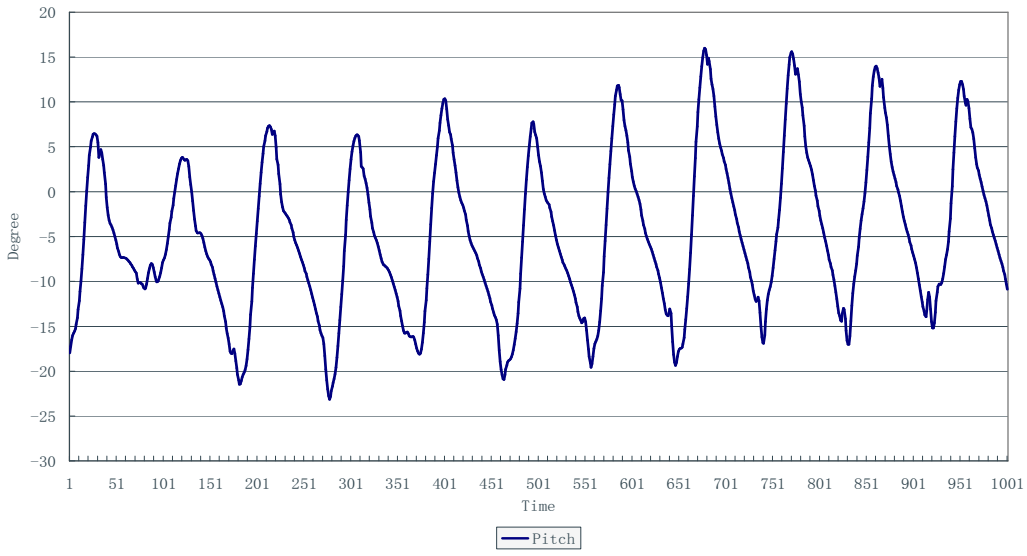


Figure 3-4 Example of Walking Pitch Graph

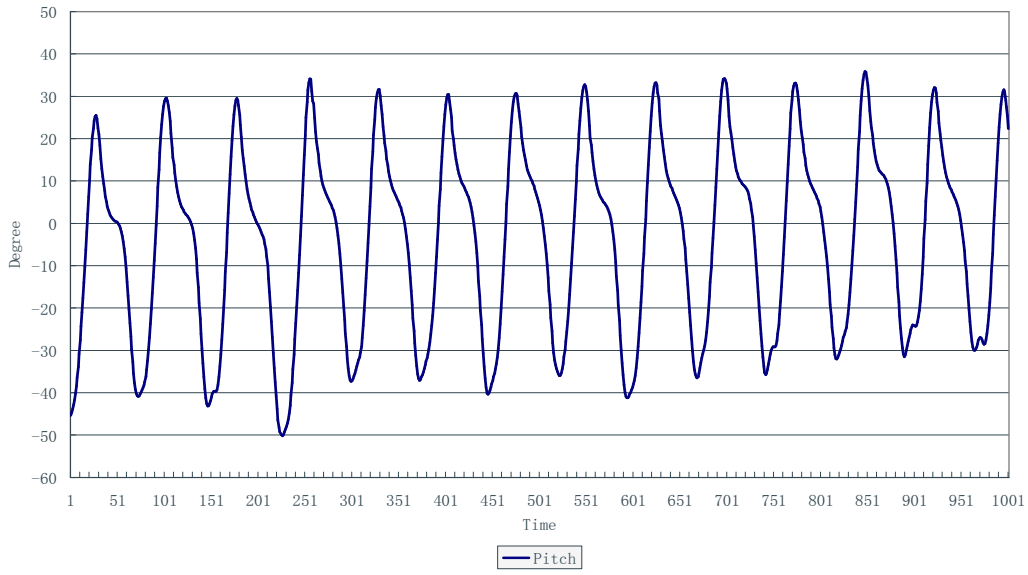


Figure 3-5 Example of Running Pitch Graph

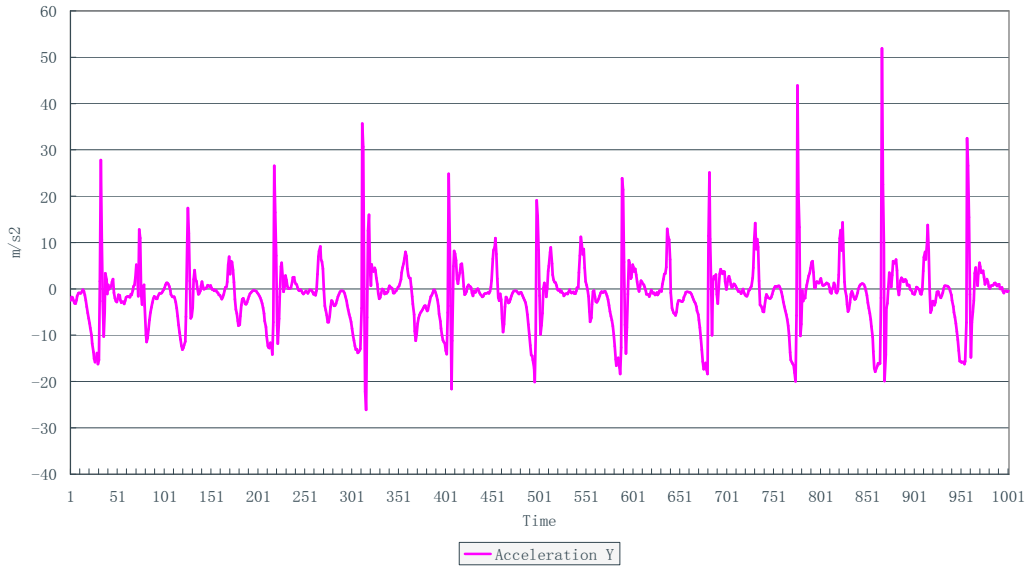


Figure 3-6 Example of Walking Acceleration on Y axis Graph

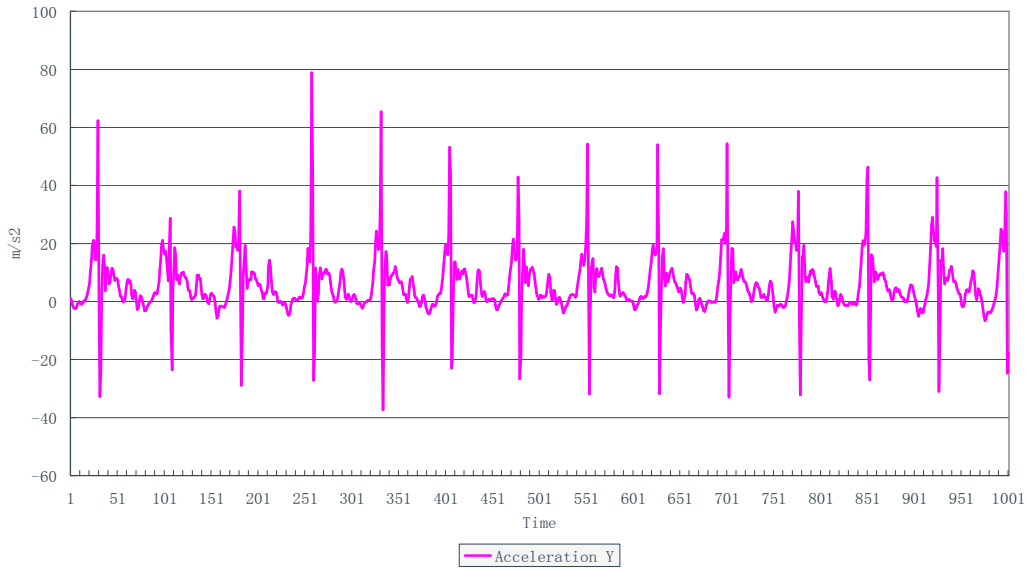


Figure 3-7 Example of Running Acceleration on Y Axis Graph

Feature Selection

To design the activity detector, I begin with differentiate between walking and running. What are the most significant features we use to tell about these two activities? The simplest answer is running should be faster than walking. Well, if we do not consider some special situation like walking very fast or running extremely slow, the previous statement is true. However, it is not reliable if determine an object is a football, only based on its shape. The same thing here, in case to make the detector more accurate, more features must be introduced. What feature from the MT9 can tell if the activity is fast or slow is about to be discovered in this section. The other things that can be considered as a feature will also be discussed here.

It is normally true, that running should be faster than walking. How do we know if the current activity is fast or slow base on the MT9 data? From the observation of human walking and running, the reason that running is faster than walking is only because running has higher frequency and longer step compare to walking. In this case, such information can be easily found from the MT9 data. From Figure 3-4 and 3.5, we can see that these two graphs look like irregular sine wave. We can clearly see the peeks and the periods. Both walking and running graphs are in same amount

of time period, which is 10 seconds. In the walking graph, there are 11 peaks. Each action takes about 10/11 seconds. By using the equation ***f = 1/T***, the frequency of walking is about 1.1Hz. The average amplitude is between 20° and 35°. In the running graph, there are 14 peaks. Each action takes about 10/14 seconds. The frequency of running is about 1.4Hz, which is higher than walking. The average amplitude of running is much higher than walking, as we see it is between 70° and 85°. Therefore, ***Frequency*** and ***Amplitude*** are considered as two features to differentiate walking and running.

The MT9 sensor has an accelerometer built in. Therefore the acceleration data towards each orientation can be collected easily. Because we only consider the Pitch value (the reason has been explained at very beginning of this section), so the acceleration data on Y axis is the only acceleration data I will study. If we graph the acceleration data on Y axis (see Figure 3-6 and 3-7), we can see it looks like a heart beat graph. We can clearly count the peaks in the graphs. There are same amount of peaks the number of peaks in the pitch graph. Each peak in the acceleration graph is correlated to the peak in the pitch graph. The acceleration value of walking is between $30 m/s^2$ and $70 m/s^2$. The value for running is higher, which is between $55 m/s^2$ and $110 m/s^2$. Therefore the ***Acceleration*** is also considered as a feature.

Feature Extraction

As I have selected the features, the problem now is how to extract these features from the MT9 incoming data. The first feature is ***Frequency***. From the elicitation of the frequency equation, I found that to calculate the frequency it is necessary to know the time period ***T***. The time period here means the amount of time that is used to take a step. To know the value of ***T***, it is essential to find out the start and the end of the cycle (See Figure 3-8). The start and the end of the cycle are the two troughs in the wave. The trough for the wave graph is explained as the section which lies below the undisturbed position. The searching for start and end of one cycle is about searching two adjacent local minima. After the ***T*** is known, the frequency can be calculated as ***F = 1/T***.

The next feature selected is ***Amplitude***. We move our leg both toward and backward when we are walking or running, so it is necessary to calculate both positive and negative amplitude. As we've already get the two troughs when we were calculating the frequency, the other element we need to calculate the amplitude is the crest, which is the local maximum between the two troughs. If the pitch value at the start is Θs, the end is Θe and the peak is Θp, then the amplitude $A = \frac{1}{2}((\Theta p - \Theta s) + (\Theta p - \Theta e))$.

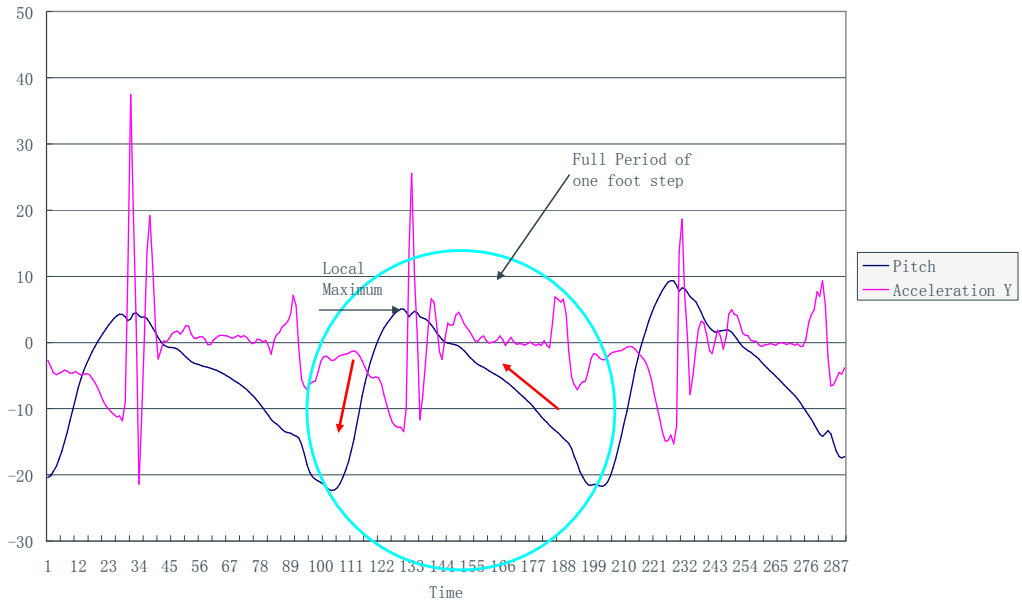


Figure 3-8 Full Cycle of One Step from Pitch and Acceleration Graph

In conclusion of calculating the frequency and amplitude of a step cycle, all we need to know is three points, which are two troughs and one crest. If we can find the three points and their correspondent pitch value, we can calculate both frequency and amplitude. One of the suitable common algorithms used in graph searching is the ***Hill Climbing***. The definition of Hill Climbing algorithm from the Oxford Dictionary of Computing is "*A fast but sometimes unreliable optimization method. When searching for the minimum/maximum value of a function a random step is taken; if the value improves it replaces the current value, then another random step is taken.*" [23] The reason I say Hill Climbing algorithm is suitable here is the characteristics of pitch value as well as the real time recognition requirement. The

pitch value is not random numbers without orderliness. It is the same in the real world, as people normally walk or run in similar pattern. If we represent it in a graph, it shall looks like a wave(see Figure 3-8). The crests and troughs can be found clearly on the graph. The Hill Climbing algorithm is fast, which is helps a lot on the real time requirement.

From the acceleration graph we can see, each point on the graph represents the acceleration of corresponding time. The problem is which acceleration value do we really need? Although the acceleration graph is not as simple as a wave graph, it is more likely be a heart beat graph. In each cycle there is a very high value and a very low value. They indicate the human foot is in the air, without touching the ground. The high value is high acceleration, which means that foot is beginning to taking the stride. Therefore that local maximum and minimum is the target acceleration value. From Figure 3-8 we can see the acceleration maxima and minima are all happened within the pitch cycle. In another word, the acceleration cycle should be related to the pitch cycle. Searching for the local maximum/minimum should be locked within the corresponding pitch cycle. Because the amount of data needs to search through is not very big, a simple list search algorithm like ***Linear Search*** is enough.

Supervised Classifier with Fuzzy Logic

After the features have been selected and extracted, it is time to make decisions. A classifier must be defined in order to decide which activity it belongs to (See Figure 3-9).

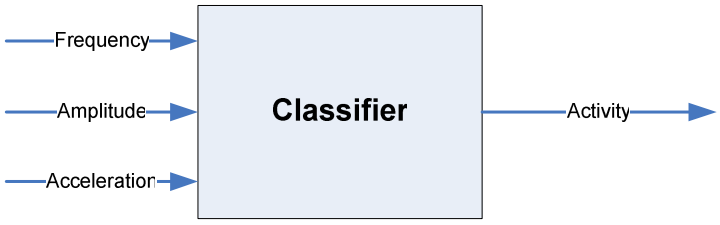


Figure 3-9 Supervised Classifier Model

In the purpose of finding the best way to design a classifier, I wrote a program. The program has a buffer to load the incoming MT9 data. Because the MT9 is running in 100Hz, it will store 100 set of data every second. These data will be examined every 3 seconds. During the examination, the program will search for 3 stride cycles (if there is). After the 3 cycles are found, it will use the Hill Climbing algorithm to get each crest and trough in order to find out the frequency and amplitude in average. Then based on the pitch cycle, the program will search the acceleration data set in order to find the local maxima and minima. The tables(Table 3-1 to 3-6) below show the result output from the program.

No. of Cycles	Frequency	Amplitude	Acceleration	Time Between
1	1.10	13.97	12.47	0.99
3	1.26	13.65	12.42	1.03
3	1.35	11.75	16.00	1.09
3	1.01	11.44	14.34	0.91
3	1.67	16.28	26.71	0.48
3	2.00	19.21	24.45	0.48
3	1.42	16.40	23.04	0.31
3	1.85	15.94	22.18	1.12
3	1.86	15.81	28.17	0.90
3	1.22	16.80	17.28	0.21
3	1.95	20.11	20.60	0.14
3	1.73	17.89	16.68	0.46
3	1.36	16.24	17.18	0.16

Highest Frequency	2.00	Highest Amplitude	20.11	Highest Acceleration	28.17
Lowest Frequency	1.01	Lowest Amplitude	11.44	Lowest Acceleration	12.42
Average Frequency	1.52	Average Amplitude	15.81	Average Acceleration	19.35

Table 3-1 Walking (A) Output Results

No. of Cycles	Frequency	Amplitude	Acceleration	Time Between
3	1.63	14.38	20.52	0.55
3	1.65	13.80	15.65	1.18
3	1.13	14.43	21.96	0.04
3	1.90	18.11	21.20	0.72
3	1.91	22.53	22.27	0.58
3	2.45	20.92	23.69	0.07
3	1.93	24.32	32.60	0.41
3	2.18	26.91	27.31	0.66
3	2.29	23.32	29.19	0.10

3	2.21	23.66	30.79	1.12
3	2.13	28.50	36.09	0.29
3	2.16	30.49	28.63	0.52
3	2.42	30.15	32.66	0.14
3	2.28	24.68	34.83	0.15
3	1.50	28.09	41.61	0.81
3	2.29	25.40	31.58	0.17
3	2.01	24.65	31.58	2.81

Highest Frequency	2.45	Highest Amplitude	30.49	Highest Acceleration	41.61
Lowest Frequency	1.13	Lowest Amplitude	13.80	Lowest Acceleration	15.65
Average Frequency	2.00	Average Amplitude	23.20	Average Acceleration	28.36

Table 3-2 Walking (B) Output Results

No. of Cycles	Frequency	Amplitude	Acceleration	Time Between
1	1.92	9.31	31.87	0.01
3	1.61	15.02	13.02	0.32
3	1.82	17.01	19.04	0.03
3	2.12	21.89	28.43	0.04
3	1.89	25.86	22.79	0.41
3	2.03	23.30	26.15	0.46
3	1.44	24.79	24.88	0.17
3	1.94	23.83	29.18	0.08
3	2.03	25.35	23.71	0.11
3	2.11	23.33	29.27	0.10
3	2.15	26.56	30.47	0.11
3	2.24	21.94	28.24	0.12
3	2.34	25.27	32.42	0.13
3	1.95	25.15	23.53	0.49
3	1.94	28.83	31.18	0.40
3	1.91	26.11	26.05	0.27
3	1.96	23.57	24.16	0.21
3	1.99	27.30	27.96	0.19
3	1.99	23.67	24.43	0.19

Highest Frequency	2.34	Highest Amplitude	28.83	Highest Acceleration	32.42
Lowest Frequency	1.44	Lowest Amplitude	9.31	Lowest Acceleration	13.02
Average Frequency	1.97	Average Amplitude	23.06	Average Acceleration	26.15

Table 3-3 Walking (C) Output Results

No. of Cycles	Frequency	Amplitude	Acceleration	Time Between
3	2.39	40.57	47.72	0.41
3	2.54	50.49	52.44	0.22
3	2.61	49.55	54.08	0.16
3	2.69	41.22	31.31	0.17
3	2.60	43.31	58.10	0.11
3	2.68	46.27	51.88	0.12
3	2.67	51.61	56.38	0.12
3	2.68	45.06	40.59	0.12
3	2.68	46.19	46.37	0.13
3	2.67	47.55	48.51	0.13
3	2.70	46.64	48.66	0.14
3	2.32	24.00	21.75	0.55

Highest Frequency	2.70	Highest Amplitude	51.61	Highest Acceleration	58.10
Lowest Frequency	2.39	Lowest Amplitude	24.00	Lowest Acceleration	21.75
Average Frequency	2.60	Average Amplitude	44.37	Average Acceleration	46.48

Table 3-4 Running(A) Output Results

No. of Cycles	Frequency	Amplitude	Acceleration	Time Between
3	2.97	29.06	42.27	0.37
3	2.60	40.59	56.36	0.30
3	2.68	45.23	52.92	0.29
3	2.69	48.57	50.29	0.33
3	2.67	43.27	43.67	0.33
3	2.62	40.08	58.15	0.26
3	2.67	37.79	39.58	0.29
3	2.70	39.70	39.20	0.30
3	2.67	38.89	33.63	0.27
3	2.63	43.70	22.87	0.24
3	2.74	41.01	40.00	0.30
3	2.68	45.28	24.86	0.31
3	2.70	47.60	39.40	0.36
3	2.69	42.53	28.48	0.38
3	2.69	44.55	22.10	0.43
3	2.67	42.44	33.74	0.43
3	2.80	48.87	29.10	0.57
3	2.78	49.84	25.71	0.69
3	2.68	44.65	23.93	0.72
3	2.62	44.40	34.96	0.71
3	3.11	35.92	46.64	0.28
3	2.56	39.75	30.67	0.59

Highest Frequency	3.11	Highest Amplitude	49.84	Highest Acceleration	58.15
Lowest Frequency	2.56	Lowest Amplitude	29.06	Lowest Acceleration	22.10
Average Frequency	2.71	Average Amplitude	42.44	Average Acceleration	37.21

Table 3-5 Running(B) Output Results

No. of Cycles	Frequency	Amplitude	Acceleration	Time Between
3	2.43	31.42	37.56	0.39
3	2.46	37.32	22.96	0.15
3	2.60	33.15	24.06	0.15
3	2.55	34.24	26.50	0.63
3	3.06	32.66	25.54	0.09
3	2.56	34.00	27.41	0.37
3	2.54	37.89	18.95	0.24
3	2.56	35.50	25.91	0.10
3	2.67	31.17	41.20	0.11
3	2.80	31.21	40.55	0.12
3	3.03	30.25	37.99	0.13
3	2.60	35.96	23.88	0.48
3	2.68	41.32	27.75	0.49
3	2.61	42.70	34.44	0.43
3	2.68	42.32	35.54	0.47
3	2.64	41.66	27.32	0.46
3	2.64	41.15	27.87	0.46
3	2.67	40.24	28.17	0.46
3	2.56	37.25	53.90	0.37
3	2.59	32.64	39.16	0.29
3	2.60	33.06	45.54	0.23
3	2.58	31.79	40.50	0.84
3	3.00	36.78	28.99	0.25
3	2.59	38.10	33.71	0.59
3	2.48	44.29	36.03	0.42
Highest Frequency	3.06	Highest Amplitude	44.29	Highest Acceleration	53.90
Lowest Frequency	2.43	Lowest Amplitude	30.25	Lowest Acceleration	18.95
Average Frequency	2.65	Average Amplitude	36.32	Average Acceleration	32.46

Table 3-6 Running(C) Output Results

The tables above show that there is no clear edge between walking and running in any features. We cannot define a threshold on any feature. Fuzzy Logic can be brought here to solve this problem.

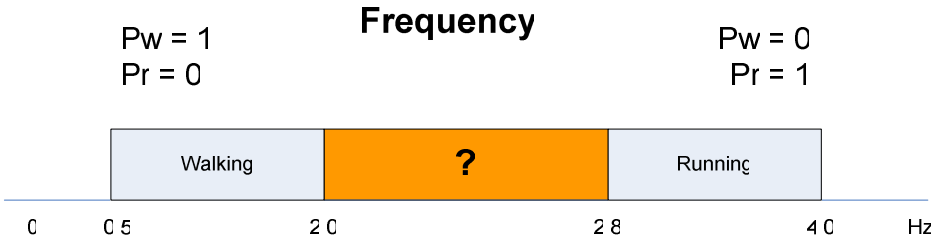
To use Fuzzy Logic, we need to find out the membership range for both walking and running. Then the fuzzy membership range must be defined. If the incoming activity belongs to the walking membership, then the probability of walking is 100% on that feature, the probability of running is 0% on that feature. If the incoming activity belongs to fuzzy membership, then the probability on walking or running must be calculated against the fuzzy membership range.

$$Pw = (Fr - F) / ((Fr - Fw) / 100)$$

$$Pr = 1 - Pr$$

Pw is the probability of walking on that feature. ***Pr*** is the probability of running on that feature. ***F*** is the incoming feature value. ***Fw*** is the edge of walking membership as well as the start of the fuzzy membership. ***Fr*** is the start of the running membership as well as the end of fuzzy membership.

Figure 3-10 shows the fuzzy membership for each features. E.g. if the incoming activity's frequency is 2.3Hz, then it doesn't belong to neither walking nor running. It is a fuzzy member. Therefore the probability of walking can be calculated as Pw = (2.8Hz -2.3Hz) / ((2.8Hz -2.0Hz) / 100) = 62.5%. The probability of running is Pr = 1 – Pw = 37.5%.



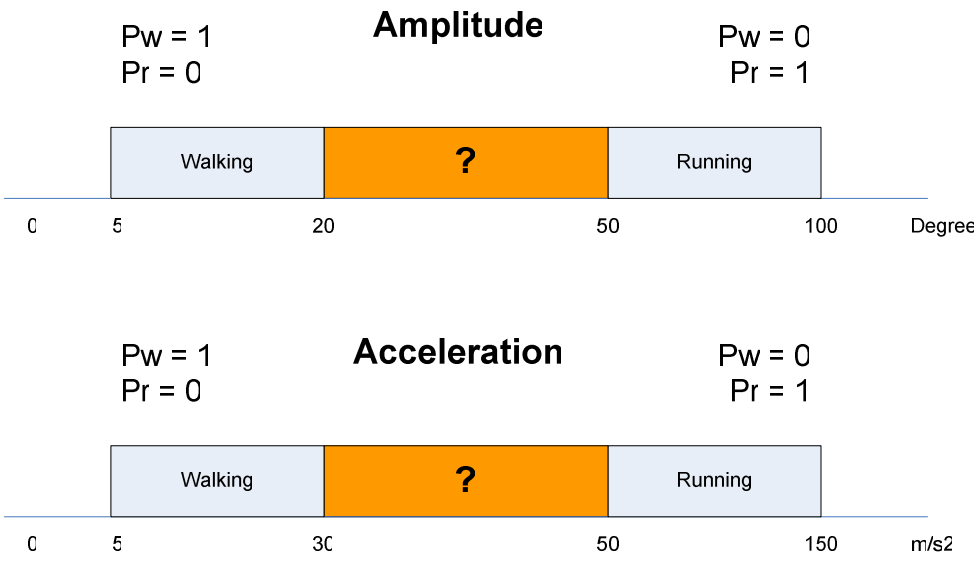


Figure 3-10 Fuzzy Membership For Each Features

After the probability on each feature is calculated, the probability on walking and running can be calculated base on the manually defined weight. The feature with high distinct and high steady gets more weight. In according to the observation on the previous data, the weight is defined as follow: ***Frequency (30%); Amplitude (50%); Acceleration (20%)***.

Finally the decision making is based on comparing the probabilities.

Problem with First Approach

The first approach gets reasonably good accuracy on differentiating walking and running within same person. However, it gives very bad recognition on different people. People walking and running are in different style. Different age group, different gender and different health condition are the facts that affect the moving style. Even people are moving in same speed, some people prefer short stride with higher frequency, some people could prefer longer stride with lower frequency. It is better to let the users use their own classifiers to differentiating the actives.

The current three features has problem on classifying the cycling, as the feature of cycling is quite similar to running. Base on these three features it is very hard to

distinct between cycling and running. Therefore the feature selection must be re-considered.

The next section will give the design of the second solution. The goal of the second solution should be overcome all the problems appear in the first approach. Instead using a fixed, supervised classifier, I hope I can find a more flexible, more intelligent method. More features will be introduced to increase the accuracy.

3.1.3 Second Approach

MT9 Data Re-analyze and Extra Feature Selection

The main aim of this part is design a solution that can recognize people's walking, running and cycling. The features selected in the first approach are frequency, amplitude and acceleration on Y axis. These three features are selected in order to detect walking or running. However, there is not enough information to detect cycling according to these three features. Therefore, the MT9 data must be re-analyzed in the purpose of finding more features.

By using the method introduced in the Section 3.1.2, I created following graphs for each actives. Instead of using the fixed time as a boundary, this time I picked three cycles for each activities. For each activity, there is a pitch graph, an acceleration graph and a combination graph. From the pitch graphs (Figure 3-11, 3-14 and 3-17) and acceleration graphs(Figure 3-12, 3-15 and 3-18), once again it proves that the three features that selected in last approach are correct.

If we comparing the pitch graph of walking(Figure 3-11) to the pitch graph of running(Figure 3-14) or cycling(Figure 3-17) here, we can see that the walking pitch graph is a bit unique. There is a minor crest beside each main crest. However, this feature cannot be selected, because this feature is very unstable. If we take a look another pitch graph(Figure 3-4) in the previous section, there is hardly find any minor crest. The minor crest can also found in the running pitch graph.

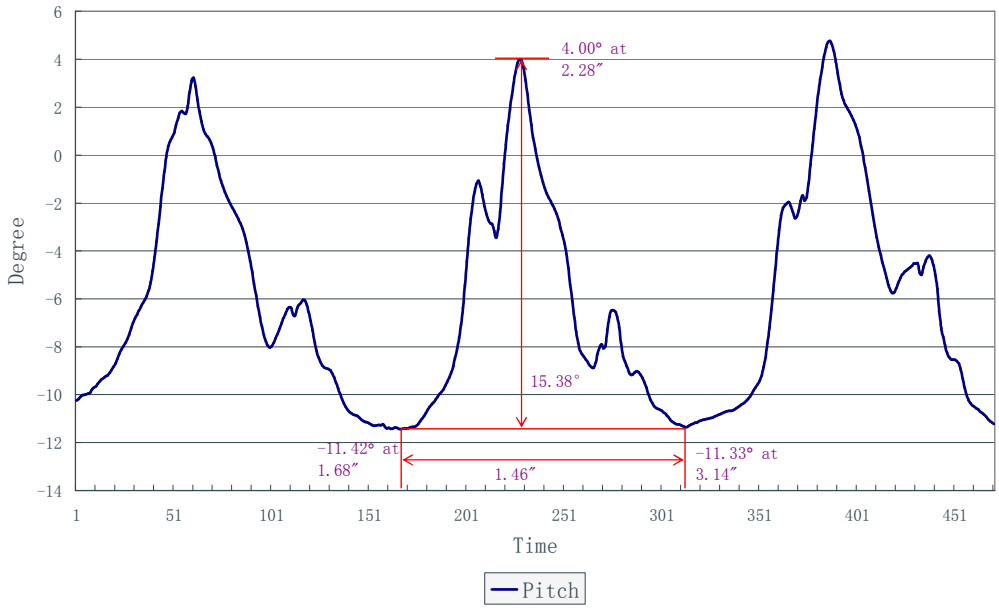


Figure 3-11 Pitch Graph of Walking

Data from Figure 3-11 : $\boldsymbol{T}$ = 1.46”; $\boldsymbol{F}$ = 1 / $\boldsymbol{T}$ = 0.69Hz; $\boldsymbol{A}$ = (4.00 – (-11.42)) + (4.00 – (-11.33)) / 2 = 15.38°

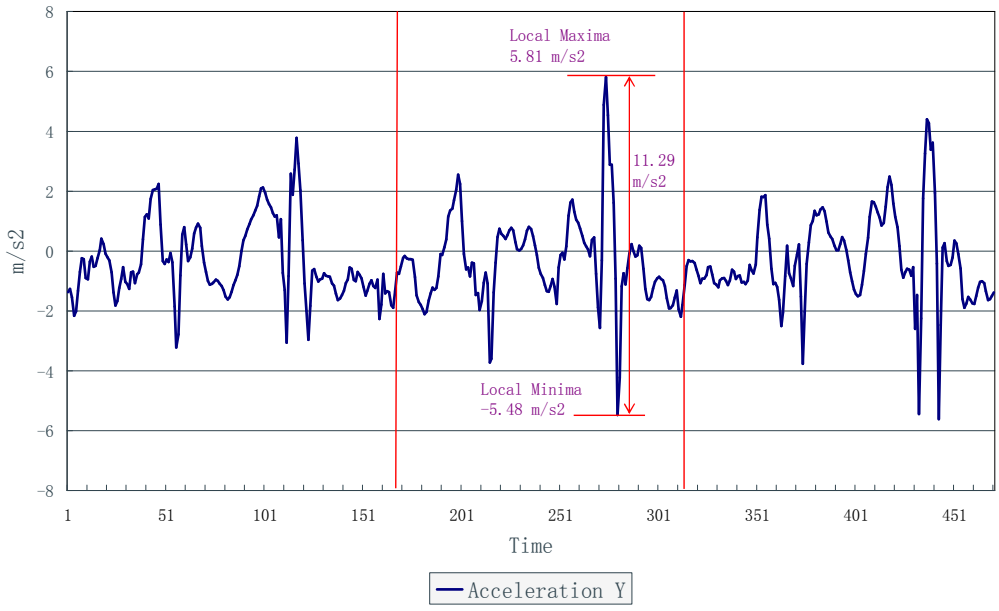


Figure 3-12 Acceleration Graph of Walking

Data from Figure 3-12 : ***Acc*** = Amax – Amin = 5.81 – (-5.48) = 11.29 m/s^2

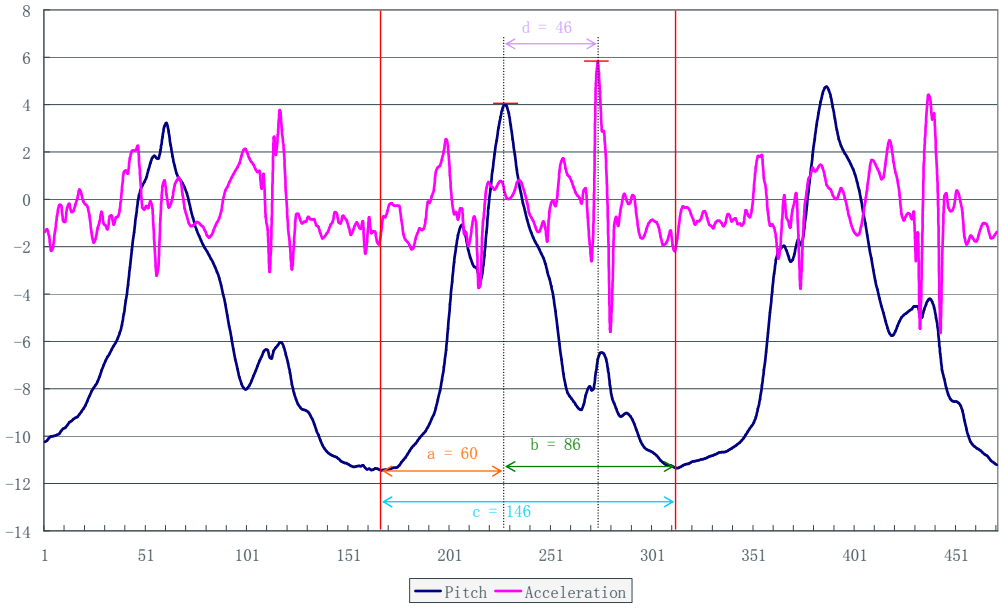


Figure 3-13 Pitch and Acceleration Combination Graph of Walking

Data from Figure 3-13 : a = 60; b = 96; c = 146; d = 46

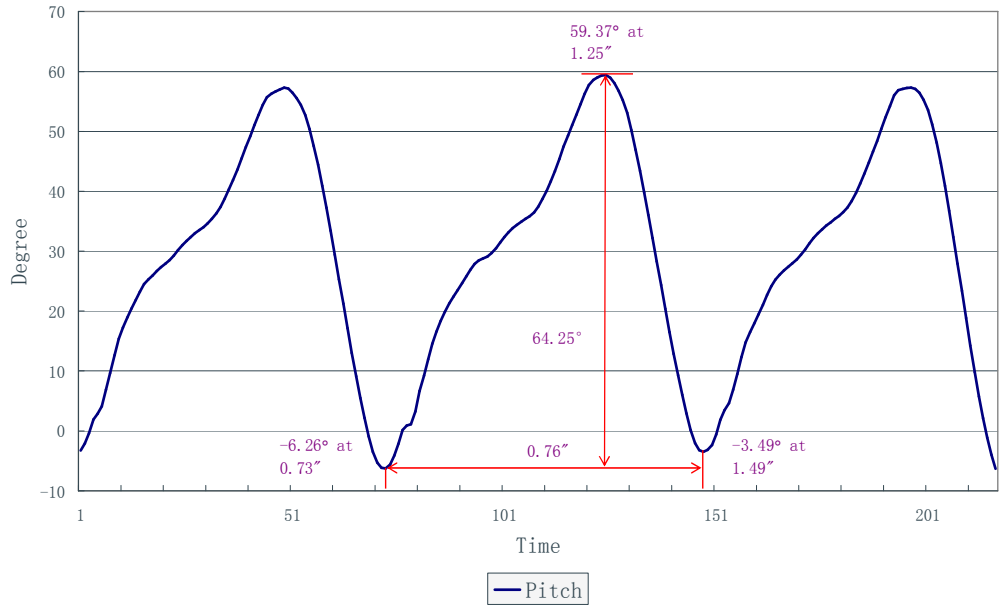


Figure 3-14 Pitch Graph of Running

Data from Figure 3-14 : $\boldsymbol{T}$ = 0.76”; $\boldsymbol{F}$ = 1 / $\boldsymbol{T}$ = 1.32Hz; $\boldsymbol{A}$ = (59.37 – (-6.26)) + (59.37 – (-3.49)) / 2 = 64.25°

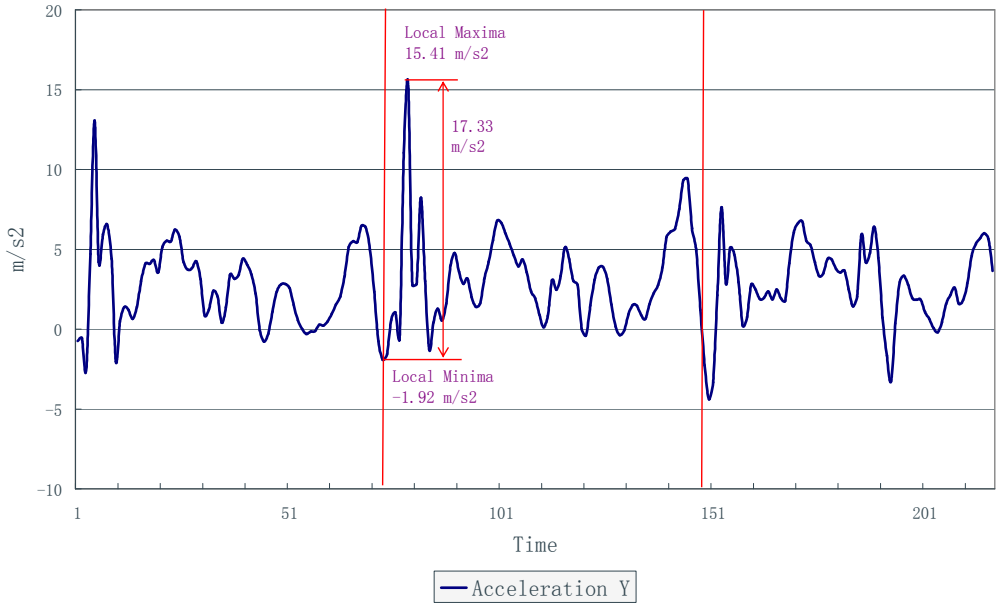


Figure 3-15 Acceleration Graph of Running

Data from Figure 3-15 : $\boldsymbol{Acc}$ = Amax – Amin = 15.41 – (-1.92) = 17.33 m/s^2

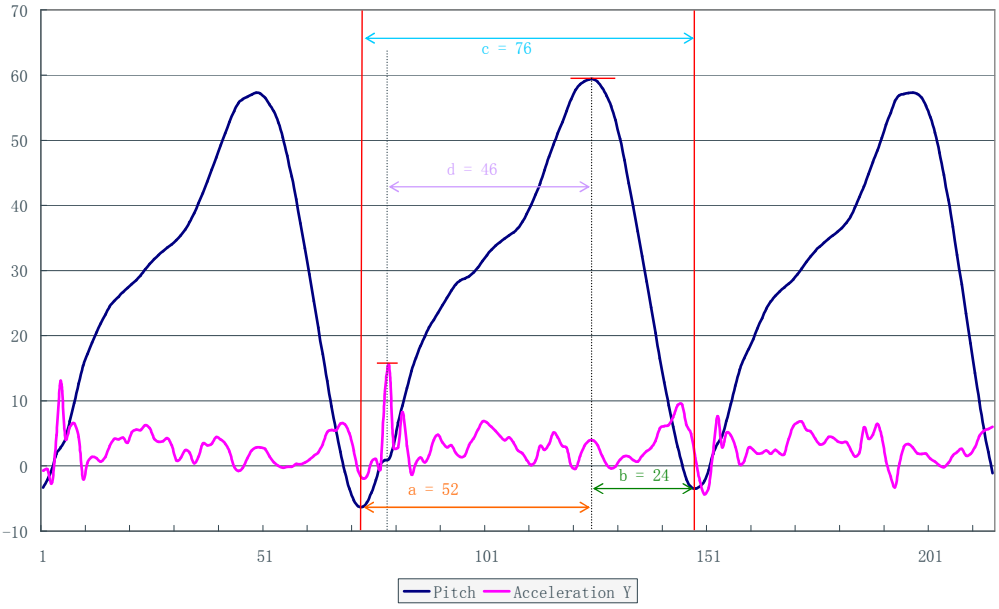


Figure 3-16 Pitch and Acceleration Combination Graph of Running

Data from Figure 3-16 : a = 52; b = 24; c = 76; d = 46

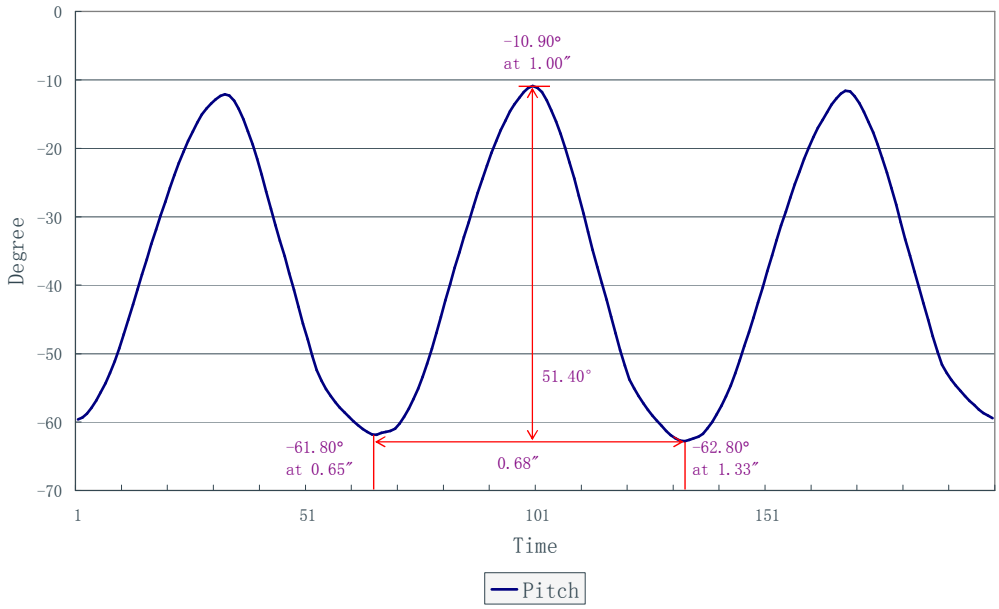


Figure 3-17 Pitch Graph of Cycling

Data from Figure 3-17 : $\boldsymbol{T}$ = 0.68”; $\boldsymbol{F}$ = 1 / $\boldsymbol{T}$ = 1.47Hz; $\boldsymbol{A}$ = ((-10.90) – (-61.80)) + ((-10.90) – (-62.80)) / 2 = 51.40°

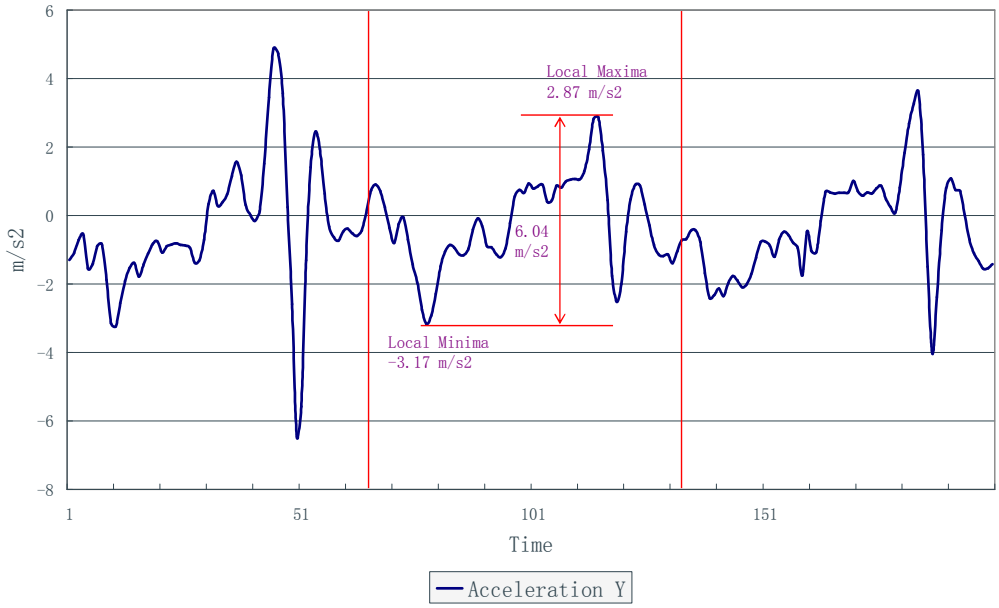


Figure 3-18 Acceleration Graph of Cycling

Data from Figure 3-18 : ***Acc*** = Amax – Amin = 2.87 – (-3.17) = 6.04 m/s^2

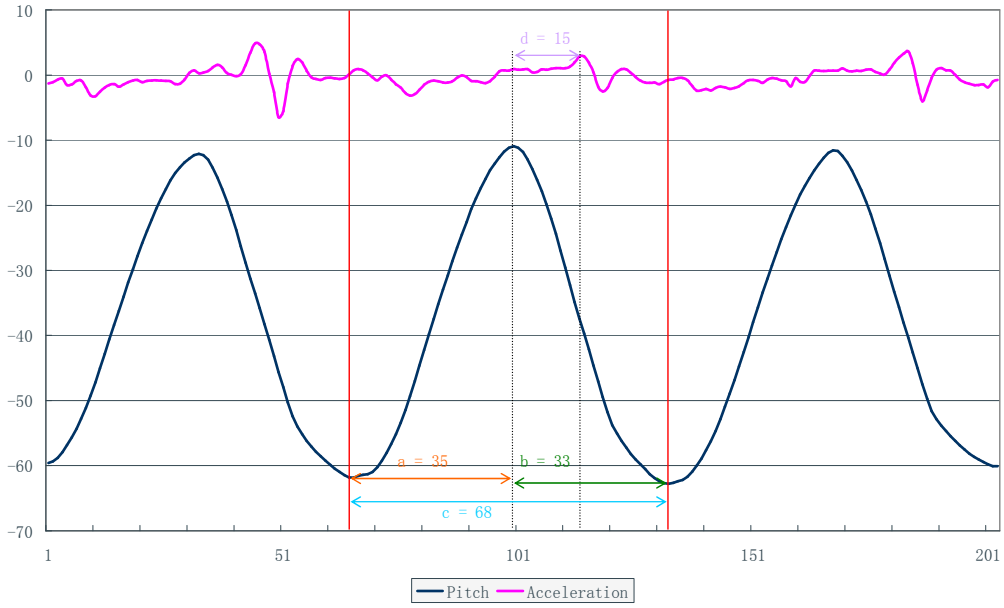


Figure 3-19 Pitch and Acceleration Combination Graph of Cycling

Data from Figure 3-19 : a = 35; b = 33; c = 68; d = 15

There is an obvious characteristic in the cycling pitch graph. Every cycle of graph is equally distributed. The slope of crest left side is similar to the slope of crest right side. This is quite distinct from walking and running. It is because when people cycling, the movement of his/her legs is more even and smoother than he/she is walking or running. Therefore comparison of slope between left and right side of crest can be seen as a feature to distinct cycling from walking and running.

According to the 3 combination graphs, there is another feature can be found. That is the time distance between the crest on the pitch graph and the local maximum of the acceleration. This piece of information indicates when the highest acceleration happens. Does it happen before user's leg rotate to the highest angle, or after? The feature can distinct different activities by telling the time distance between the maximum rotation and maximum acceleration.

Extra Feature Extraction

The two new features added in are ***1.***the slope comparison between left and right side of crest on the pitch graph; ***2.***time between the maximum leg rotation and the highest acceleration point.

To calculate the slope, we can use line equitation $y = mx + b$, as we've already have two points. One point is crest the other point is the one of two troughs. After two slopes are calculated, we can compare them by give the proportion. There is a simple way to make the comparison. Instead of calculating the slopes, all I need to get is the time distance between the crest and the two troughs. From the combination chart(Figure 3-13, 3-16 and 3-19), the time distance are represented by ***a*** and ***b***. This feature can be extracted as $\frac{a}{b}$. From the examples we can see, $\frac{a}{b}$ of walking is <1; $\frac{a}{b}$ of running is >1; $\frac{a}{b}$ of cycling is ≈1.

The second extra feature is just the time distance between the local maximum of pitch value and the local maximum of acceleration on Y axis. ***d*** on the combination charts (Figure 3-13, 3-16 and 3-19) is the time distance. Because the cycle for different activities is different, so we cannot compare ***d*** directly. E.g. ***d*** for walking (Figure 3-13) is 46. ***d*** for running (Figure 3-16) is 46 too. Obviously, the distance in running is much longer on the chart. That is because the cycle period for running is much shorter in this example. Instead of comparing the ***d*** directly, ***c*** can be introduced here to standardize the comparison. ***c*** is the distance between the two troughs, which is the cycle period of a movement. The feature can be extracted as $\frac{d}{c}$. From the examples we can see, $\frac{d}{c}$ of walking is ≈ 0.3; $\frac{d}{c}$ of running is ≈ 0.6; $\frac{d}{c}$ of cycling is ≈ 0.2.

Unsupervised classifier

The main problem of the first approach is on its supervised classifier. It is extreme difficult to model different people's activities in advance. Age, gender, size and

health condition etc. are the facts that affect people's activity style. If we manually defined a classifier, this system possibly only work well with only certain group of users.

If we want to use the monitor widely, we definitely need more than one classifier. One solution is testing the entire possible user delegate from that user group i.e. different age group with different gender, different body size and different health condition, etc. And then build one unique classifier for each user group. New user has to register to the system, and then the system would know which group the user belongs to. The suitable classifier will apply to this user to monitor his/her activities.

However, implementing such solution has huge difficulties. Firstly, it is almost impossible to testing all different user groups. Even if all the group delegates have been tested, the classifiers will be huge amount of data. We possibly need to use database to maintain the classifiers. To run a huge database on a mobile device is unreasonable. The monitor is designed for personal use, therefore there is no need to have other people type of people's classifier stored on his/her own monitor. Another vital problem with such solution is the way it classify the user into groups. Base on the user's personal details like age, gender and body size, to reckon the user's activity style is very imprecise. People's activity style is not only affected by the physical status. People's character and habit can also affect the activity style.

If we grab two features (frequency and amplitude) and make a graph in the feature space(See Feature 3-20), we can see there are clearly regions for each activity.

According to this characteristic, we can build an unsupervised classifier. The activity monitor is designed to detect three different activities, therefore in the feature space there are only three clusters. The classifier should know the centroid for each cluster in order to classify the activities by searching the minimum distance.

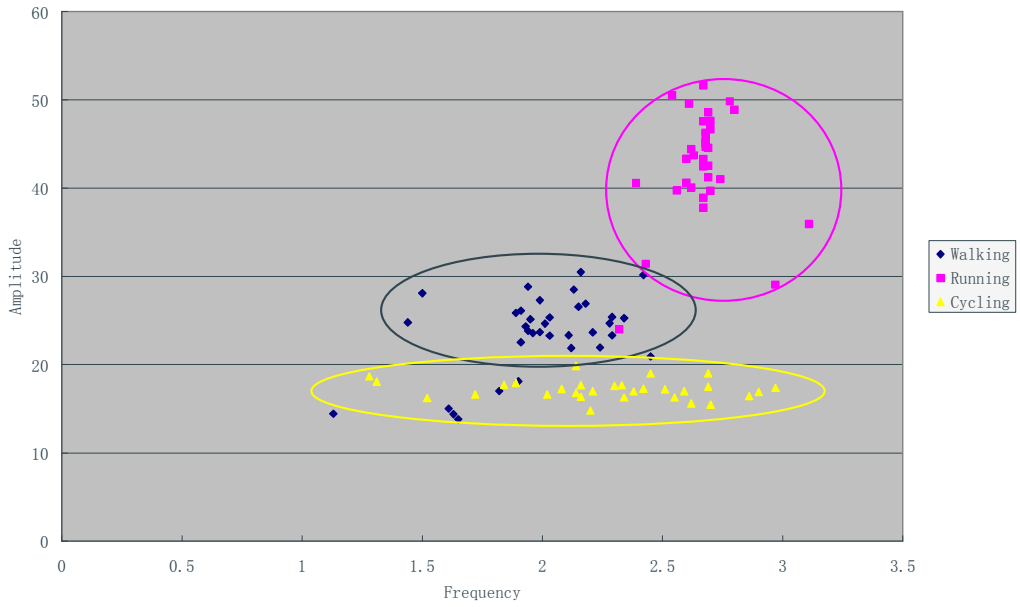


Figure 3-20 Activity Diagram in the 2-D Feature Space

To build such classifier, the most important thing is finding the centroids in the feature space for each cluster. An unsupervised training should be carried out, which means the activity monitor won't be able to use until it finish a training process. The training process will focus on learning each activity. Let's take training on walking as an example. The learning process begins with initiating centroids for each feature by randomly creating a value for each feature. When the user begins to walk, the MT9 data will be passed to the Feature Extractor. The Feature Extractor will extract 5 features from the raw data. And then the new centroid will be calculated by getting the mean value between the new feature value and the old centroid value. This process will be repeated many times until the centroid does not have big change. The number of times will be evaluated by testing in the feature. (See Chapter 5 Evaluation) When all three activities have been trained, the classifier building process for this user is finished. Although the method makes the monitor inconvenient to use, this will make the classifier act according to actual circumstances. Such defect can be improved from the application layer, which will be introduced later in this chapter(See Section 3.3).

Statistical Pattern Recognition

The decision making is achieved by the minimum distance classifier. During the user is doing activity, the 5 features will be extracted from the MT9 raw data. Then the distance to the every centroid of each feature will be calculated. The distance is just the absolute difference.

The next step is calculating the probability of certain activity on that feature. Each feature is weighted according to the same circumstance as it states in the first approach. The probability of one activity will be 100% under that feature's weight if it gets the minimum distance to the incoming feature. The probability of other activities will be calculated against the minimum distance activity. E.g. the frequency of walking centroid is ***Fw***, the frequency of running centroid is ***Fr***, and the frequency of cycling centroid is ***Fc***. The incoming activity's frequency is ***F***, and then the distance of walking in the Frequency space is $\boldsymbol{Dw = |Fw - F|}$;

distance of running in the Frequency space is $\boldsymbol{Dr = |Fr - F|}$;

distance of cycling in the Frequency space is $\boldsymbol{Dc = |Fc - F|}$;

if the ***Min(Dw, Dr, Dc) = Dw***, then the probability of walking ***P(w|Freq)*** under this feature is 100%. The probability of running ***P(r|Freq)*** under feature of Frequency is equal to ***Dw/Dr***. The probability of cycleing ***P(c|Freq)*** under feature of Frequency is equal to ***Dw/Dc***.

The total probability of each activity is the sum of probability of the activity given by one feature multiple by its weight. The activity with highest probability will be the recognized activity.

Advantages of Second Approach

The first advantage of second approach is its flexibility. The approach makes the application on top it suit everybody. The second advantage is about its extensibility. Such design makes the system much easier to add in new activity. The new activity can be learned by the system itself, without manually learning and studying on its

features. By using the second approach there is no need to model each activity for every user groups. The Activity Monitor user can have his/her own classifier to detect his/her own activities. Such design is much easier to implement and require much less resource.

3.2 Energy Calculation Stage

3.2.1 Goal

The application purpose of this project is calculating the energy expenditure while the user is doing activities. The goal of this stage is researching in the exercise physiology and finding out the equations that can calculate the energy expenditure based on recognized activities.

3.2.2 Research in Exercise Physiology

The purpose of researching in the exercise physiology is about understanding the human energy expenditure. There are two questions should be answered after the research. The first question is how to measure the human energy expenditure? The second question is what factors will effect the human energy expenditure?

There are two methods for determining human energy expenditure. One is direct method, which measures heat production in an appropriately insulated calorimeter. The other is indirect method, which measures energy expenditure from measurements of oxygen consumption and carbon dioxide production. The direct method is beyond the scope of this project. Therefore we need to use the indirect method, which is calculating the oxygen consumption, to measure the energy expenditure. [20]

For each person, heart rate and oxygen consumption relate linearly throughout a large range of aerobic exercise intensities. Therefore if we know the heart rate of user, we can estimate his/her oxygen consumption. Then the energy expenditure can also be calculated accordingly. However, this method requires extra hardware to detect user's heart rate. I want to use a method to measure energy expenditure with current available hardware only.

There is exciting equation to calculate the oxygen consumption base on type of activity, speed, and the person's weight. At the moment we have already known the user's activity type. User's weight cannot be detected directly, so we can ask the user to input his/her weight manually in advance. About the movement speed, this will be discussed in the next section. The equation of calculating the energy expenditure will be introduced in the Section 3.2.3.

3.2.3 Speed Calculation

From the research in exercise physiology, we know that it is very difficult to calculate human energy expenditure on certain activity without knowing the movement speed. So far, speed is still an unknown element. Although, the MT9 has a accelerometer built in, it will only return the acceleration value on certain axis. It will not tell any information about the object speed.

The initial design of calculating the speed is based on extra hardware support. The easiest method is using a GPS. By using the GPS, the distance of movement can be found. And then, the speed can be calculated as distance divide by time. However, there are two disadvantage of such method. Firstly, such design does not work in gym. If user walk or run on a treadmill, GPS will not detect any change in distance. Therefore, it will not be able to calculate the user's movement speed. If we use a GPS, all it does is checking distance. The extra hardware will make the monitor cost much more， which is the second disadvantage.

In the purpose of utilize the MT9 adequately, a solution of calculating the speed by using current available information is under investigation. By using the feature extractor, we can find out the frequency of the movement directly. At the meanwhile, the sensor can return the angle of leg movement. (See Figure 3-21) The angle θ can be found by calculating the difference between the maximum and minimum orientation change on pitch. From geometry point of view, we can calculate the step length if we know the leg length of the user $L_{step} = 2 \times (Sin \frac{\Theta}{2} \times L_{leg})$. The speed of the user can be calculated as $V = L_{step} \times Freq$. This method cannot precisely calculate the user's current speed, but estimate the speed.

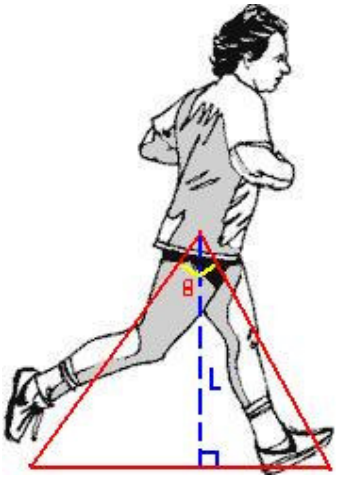


Figure 3-21 Estimating Speed

Calculating the cycling speed is much easier, because every cycle of a leg movement will course same length of actual movement. All we need to know is the frequency of the movement and the perimeter of the bicycle wheel. Then the speed can be calculated as $V = P_{wheel} \times Freq$.

3.2.4 Energy Expenditure Calculation

The ***basic equation*** for energy expenditure is **Oxygen Consumption** $\dot{V}O_2$ $(mL \cdot kg^{-1} \cdot \min^{-1}) =$ **Resting component** (1MET $[3.5\, mL \cdot kg^{-1} \cdot \min^{-1}]$)+**Horizontal component** (speed, $[m \cdot \min^{-1}] \times$ oxygen cost of horizontal movement) + **Vertical component** (percent grade × speed $[m \cdot \min^{-1}] \times$ oxygen cost of vertical movement).

$\dot{V}O_2$ = **Resting component + Horizontal component + Vertical component**

In the case of walking, Oxygen cost of horizontal component of movement equals to $0.1\,mL \cdot kg^{-1} \cdot \min^{-1}$, and $1.8\,mL \cdot kg^{-1} \cdot \min^{-1}$ for the vertical component. In the case of running, Oxygen cost of horizontal component of movement equals to $0.2\,mL \cdot kg^{-1} \cdot \min^{-1}$, and $0.9\,mL \cdot kg^{-1} \cdot \min^{-1}$ for the vertical component.

For example, a 60-kg person walks on a treadmill at $80\,m \cdot \min^{-1}$ up a 4% grade. The energy expenditure of this person can be calculated as:

Oxygen consumption $\dot{V}O_2$ = Resting component + Horizontal component + Vertical component

$$\dot{V}O_2 = \text{Resting}\,\dot{V}O_2\ (mL \cdot kg^{-1} \cdot \min^{-1}) + [\text{speed}(m \cdot \min^{-1}) \times 0.1\ mL \cdot kg^{-1} \cdot \min^{-1}] + [\%\ \text{grade} \times \text{speed}(m \cdot \min^{-1}) \times 1.8\,mL \cdot kg^{-1} \cdot \min^{-1}$$

$$= 3.5 + (80 \times 0.1) + (0.04 \times 80 \times 1.8)$$

$$= 17.26\,mL \cdot kg^{-1} \cdot \min^{-1}$$

$$\text{Energy Expenditure Kcal} \cdot \min^{-1} = \dot{V}O_2\ (mL \cdot kg^{-1} \cdot \min^{-1}) \times \text{Body mass (kg)} \times 5.05\ \text{kcal} \cdot LO_2^{-1}$$

$$= 17.26\,mL \cdot kg^{-1} \cdot \min^{-1} \times 60\text{kg} \times 5.05\ \text{kcal} \cdot LO_2^{-1}$$

$$= 5.23\ \text{Kcal} \cdot \min^{-1}$$

In this project, the vertical component will not be considered. By only using one MT9 sensor, it is very difficult to detect the gradient of the movement surface. Besides, the testing environment for the project is in GYM, there is no facility to get the uphill or downhill walking/running data.

The equation for calculating energy expenditure of cycling is a bit different from walking and running. Instead of considering horizontal and vertical component, the wind speed becomes a fact [19]. The equation can be represented as follow:

$$\dot{V}O_2 = -4.50 + 0.17V_R + 0.052V_W + 0.022W_R$$

Where V_R is rider speed in kph, V_W is wind speed in kph (expressed as positive number for a headwind, negative number for a tailwind), and W_R is rider weight in kg.

3.3 Application Architecture Stage

3.3.1 Goal

The goal of this stage is designing a real application on top of previous two stages. The application should provide user all necessary functions as well as a good interface.

3.3.2 Function Overview

There are 4 main functions that are provided by the application. There is only one actor interact with the application, which is the user. The use case diagram in the next page (Figure 3-21) briefly model the application architecture.

3.3.3 Application Analysis and Design

Login

The application is designed to have multiple users. Each user will have a profile stored as a file in the local machine. This file contains user's personal information such as gender, weight, height etc. as well as user's own activity detection classifier details. Once the application is turned on, the login management is the first function provides to the user. The application should promote the user to enter the user name. This username will be verified against local files. If the profile is found, which means the user has registered before, and then the activity monitor will be turned on. If user input an invalid username, the user will be blocked. On the login dialog, the application should also allow the new user to select the registration option. There is no security issue within this application, so it is not necessary to have password to authenticate each user. Figure 3-23 is the sequence diagram of the login management.

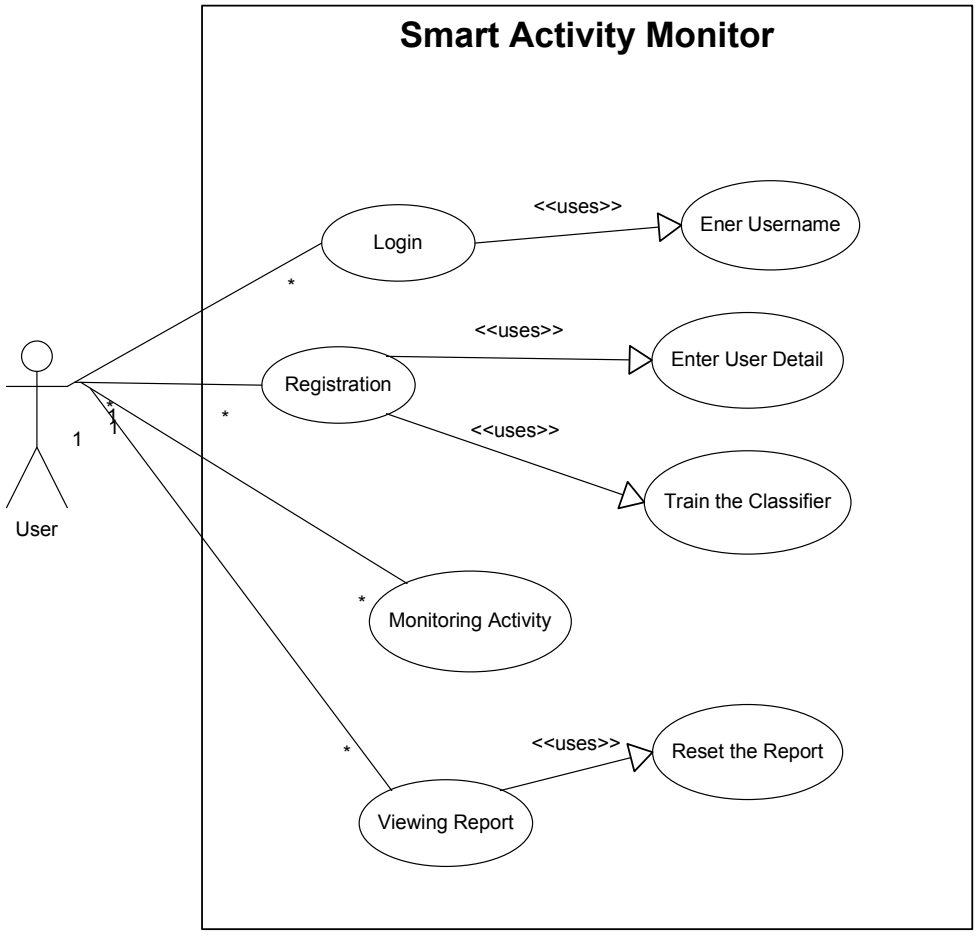


Figure 3-22 Use Case Diagram

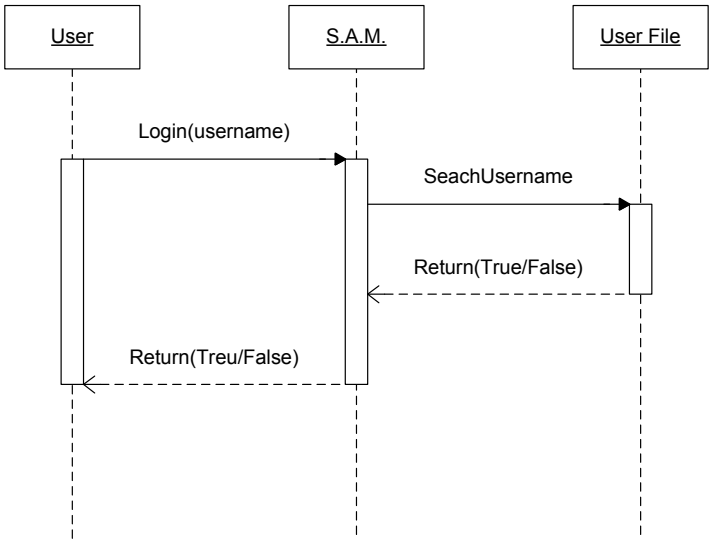


Figure 3-23 Sequence Diagram of Login Management

Registration

Registration is required to every new user. There are two steps during the registration. The first step is called Personal Details. The user is required to fill in a form during this step. The form contains user's personal information, which includes username, age, gender, height, weight. The username here will be the key to login the system in the future. The rest information will be used to calculate the energy expenditure.

After the form is finished, the next step is called Training. During this step the user will be asked to perform each activity in his/her normal way for a certain period of time. During the user is doing activity, the application will record the MT9 data and call the lower API, which is the activity detector to analysis the data. As soon as all the activities have been trained, the application will call the API to build a classifier for future activity detection (See Figure 3-25).

Finally, the application will store the user profile and the training result as a file in the local machine. Figure 3-24 shows the sequence diagram of registration.

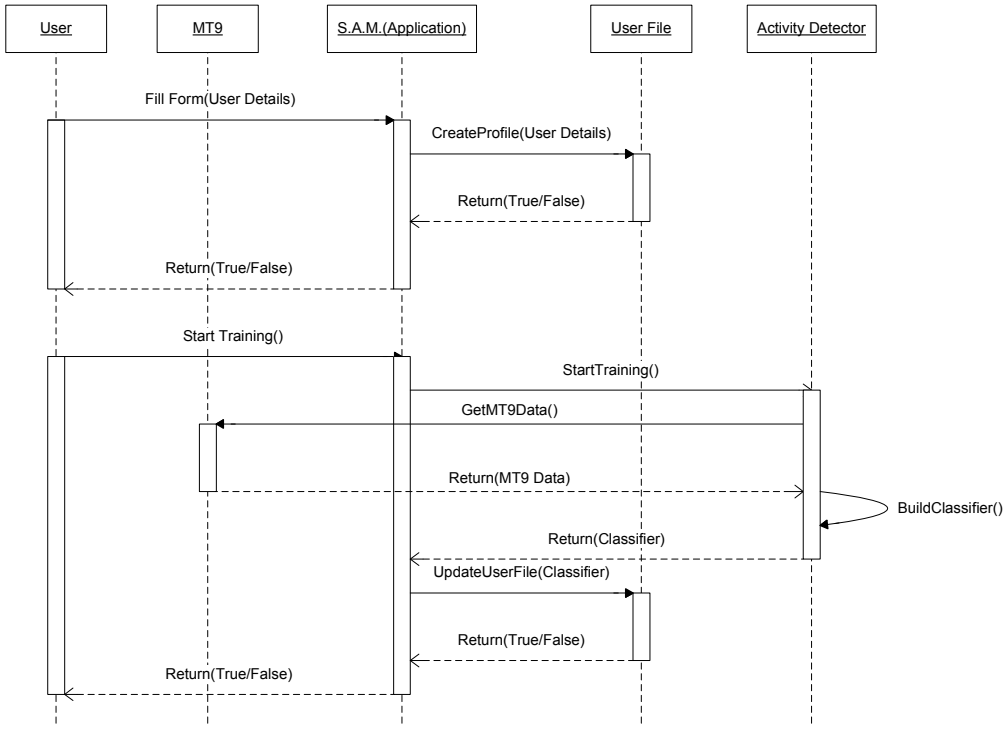


Figure 3-24 Sequence Diagram of New User Registration and Training

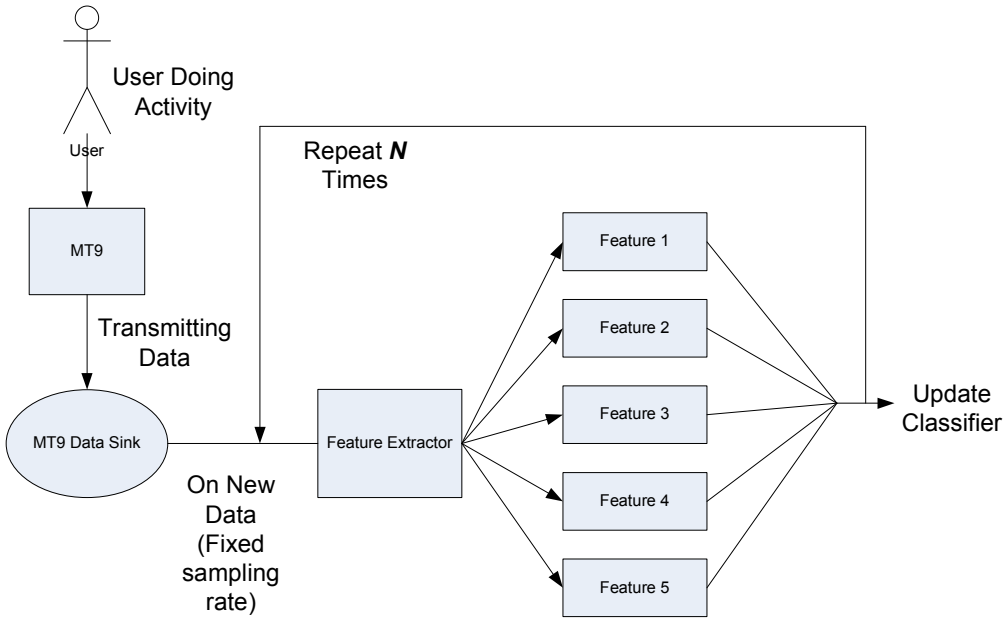


Figure 3-25 Classifier Training Component Overview

Monitoring

This application is called activity monitor. As the name tells everything, the monitoring is the core part of the application. However, this is the smart activity monitor. The most important and complex part does not mean is should be the most difficult part to use. To make the monitor smart, it should be very easy to understand by the user as well as easy to operate by the user. Basically the monitor should have one input and few outputs. The input is the MT9 data from the sensor, which attached to user's ankle. The data will be processed by the Activity Detector API. And then the recognized activity will be returned to the application. This information will be displayed on the monitor. The application will also calculate the user's speed and energy expenditure according to its activity type (See Figure 3-27). These two pieces of information should also be displayed on monitor in real time. The user should have the control of the monitor. The user should be able to turn on/off the monitor easily. Figure 3-26 is the sequence diagram of Monitoring.

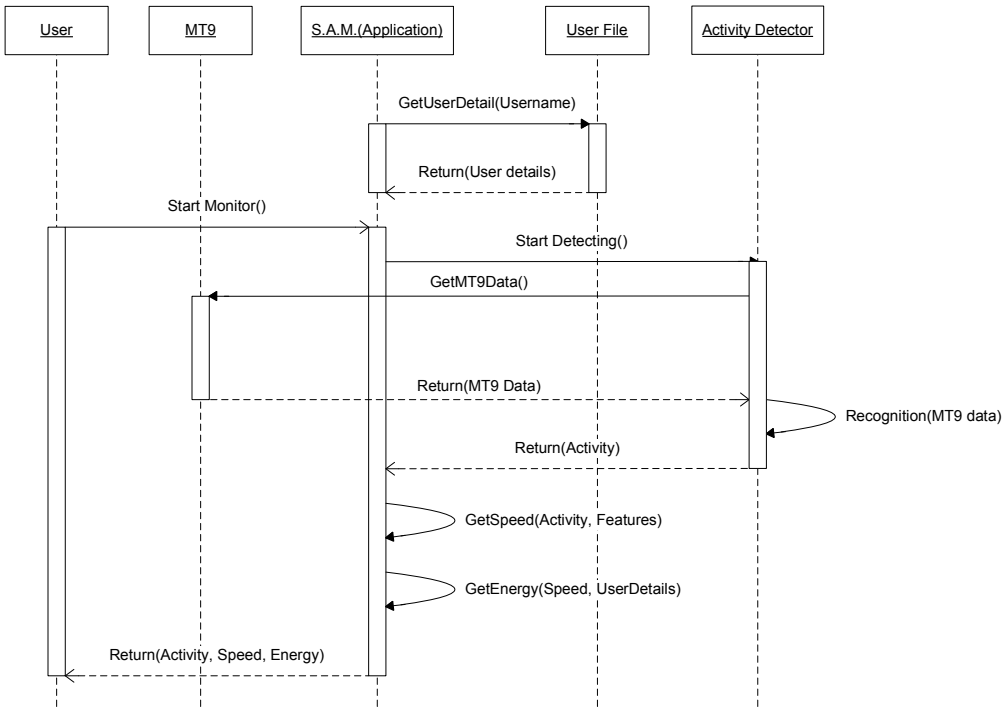


Figure 3-26 Sequence Diagram Activity Monitoring

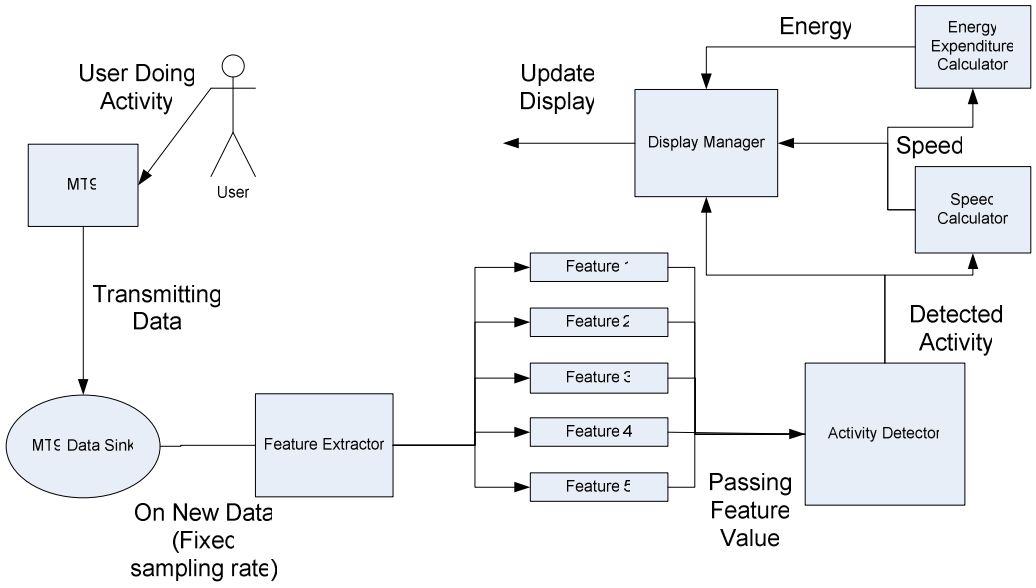


Figure 3-27 Activity Monitoring System Component Overview

Report Generation

The application can generate a simple report. This report will give user a summary of his/her previous activities. The user can know how much energy in total has he/she been spent, how long has he/she been involved in certain activity. The user can reset the report in order to start a new monitor. Figure 3-38 is the Sequence Diagram of Report Generation.

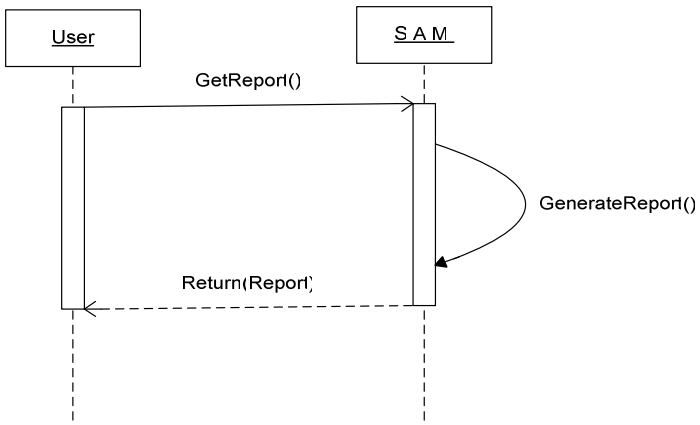


Figure 3-28 Sequence Diagram of Report Generation

3.3.4 Interface design

In this section I've designed some wire frames for the interface on PDA screen.

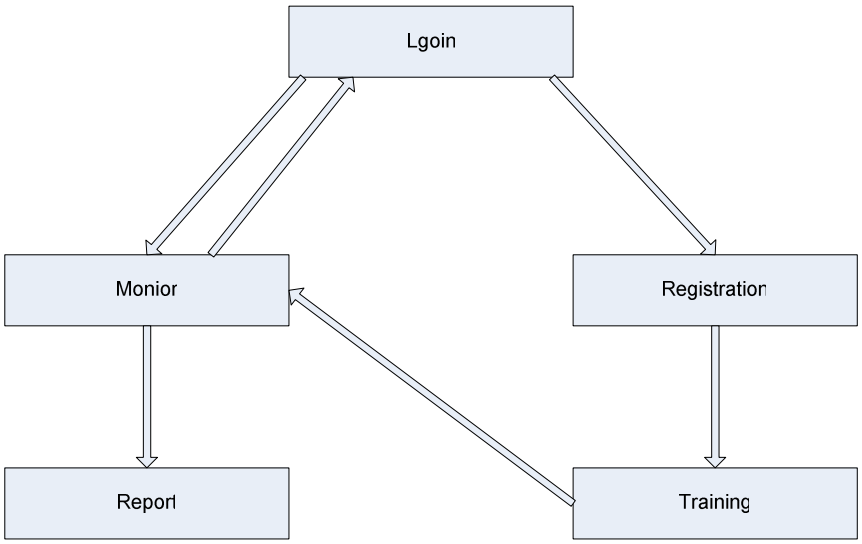


Figure 3-29 Map of Interface Navigation

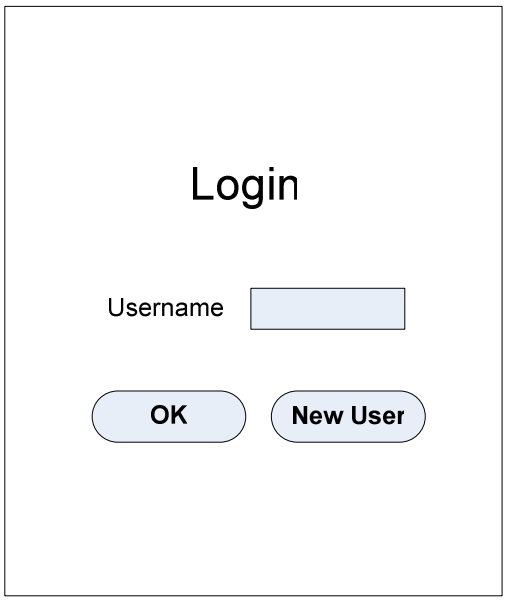


Figure 3-30 Wire Frame for Login

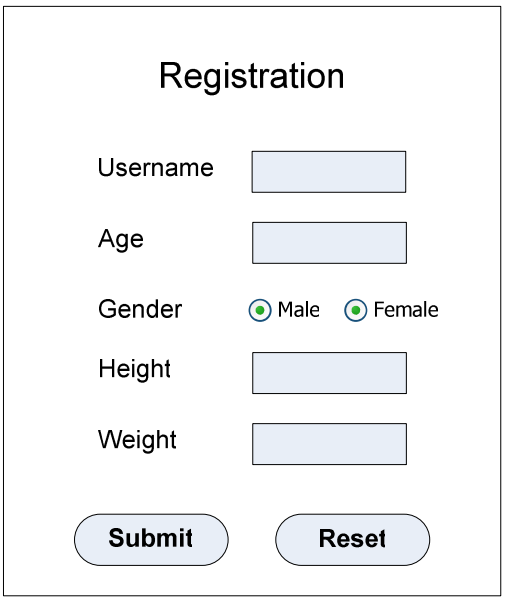


Figure 3-31 Wire Frame for Registration

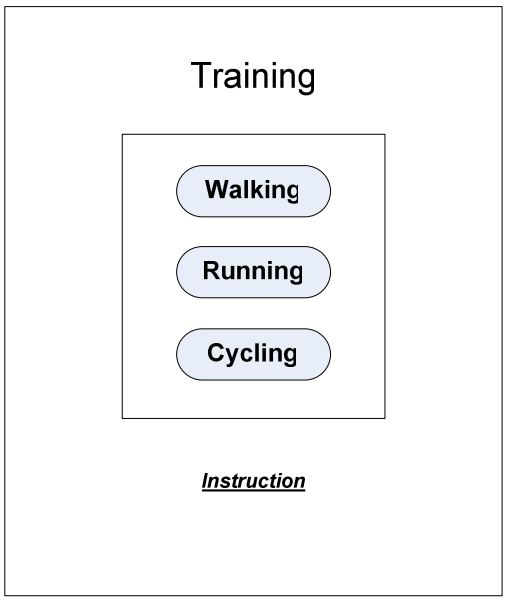


Figure 3-32 Wire Frame for Training

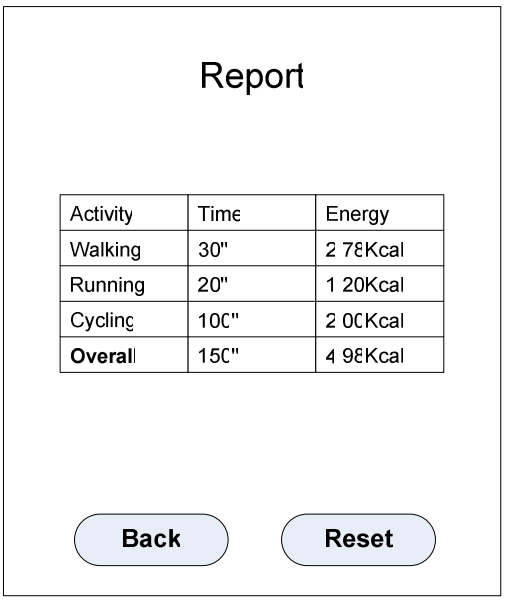


Figure 3-33 Wire Frame for Report

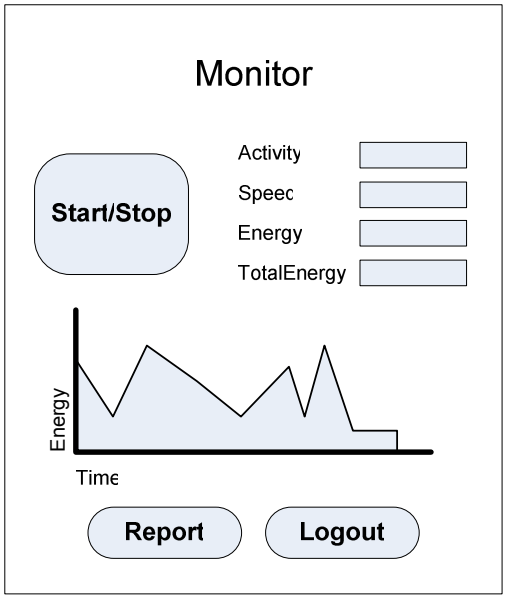


Figure 3-34 Wire Frame for Activity Monitoring

Chapter 4: Implementation

Following from the design and technologies review, it's time to implement the activity monitor. This chapter will document the decision made and the reasoning behind them.

4.1 Overview on Implementation Process

The implementation begins with studies on the technologies which will be used in this project. The studies includes both the hardware and software. The implementation process follows the steps of design process, which has 3 stages too. During the first stage, the activity detector must be implemented. Feature Extraction, Building Classifier and Activity Recognition are the three main parts need to be focused on. This part of project should be independent from the application. The second stage will be implementing the energy calculation. The last stage is the application level implementation. During this stage, a file system and all the user functions should be implemented. There will be also some extra functions added to this project to makes it more usable.

4.2 Technology Overview

This section describes the technologies that will be used in this project. The hardware including motion sensor, handheld and some accessories will be introduced first. And then the development environment and the SDK from the sensor manufacturer will be discussed.

4.2.1 Motion Sensor

The motion sensor used in this project is the MT9-B from the Xsens Technologies B.V. [24] The MT9-B contains 3 types of sensors, which are gyroscopes, accelerometers and magnetometers. Therefore it provides serial digital output of 3D acceleration, 3D rate of turn (rate gyro) and 3D earth-magnetic field data. The MT9 SDK, supplied by the manufacturer, contains a proprietary algorithm tailor-made to the MT9 that can accurately calculate absolute orientation in three-dimensional space according to the 3 types of sensed data. The picture below is the MT9-B (Figure 4-1).

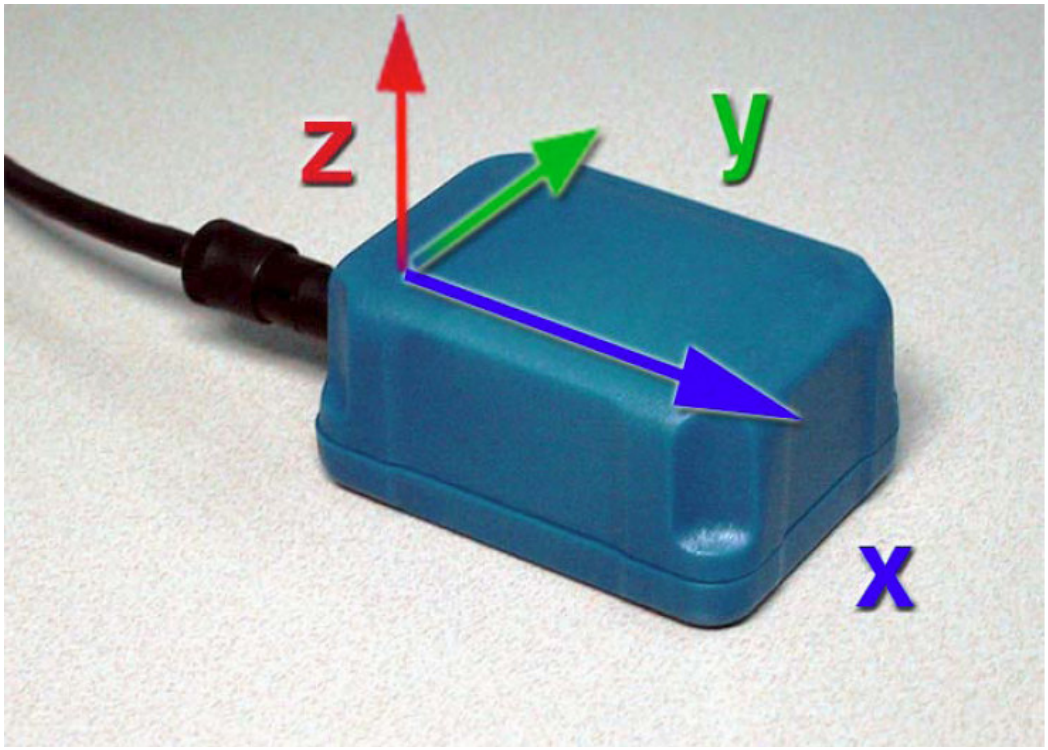


Figure 4-1 Picture of a MT9-B Sensor

The size of MT9-B is very compact. It's only 39mm x 54mm x 28mm, which is slightly bigger than a match box. It is also very light. This makes the sensor quite easy to attach on human's leg by using adhesive tape. It has serial interface to output the sensed data. The product does not support battery power supply. It is not a wireless device.

4.2.2 Handheld

The handheld used in this project is HP iPAQ 5500. This PDA has a 400 MHz Intel® XScale™ technology-based processor and 48 MB Flash ROM Memory as well as 128 MB SDRAM, which makes it very powerful as a handheld. This high specification mobile device is running on Microsoft Pocket PC 2003™. The Microsoft Pocket PC OS support a wide range of programming language and many tools for compiling them are readily available. The picture below is the HP iPAQ 5500 running the Pocket PC (See Figure 4-2). Another important feature of this handheld is it has 3 types of wireless communication interface on board, which includes Bluetooth, 802.11 and infrared.

Specifications:

- 400 MHz Intel® XScale™ technology-based processor
- 48MB ROM and 128MB RAM
- 240 x 320 (64k) touch-sensitive TFT color display
- Built-in Bluetooth/Infrared/WiFi
- SD Expansion Port, and ability to support a compact flash expansion jacket
- Integrated Biometric Fingerprint Reader
- Microsoft Pocket PC 2003
- 4.68” x 2.95” x 0.73”
- 206.8g [25]

Figure 4-2 Picture of HP iPAQ 5500

4.2.3 Accessories

The iPAQ itself does not have RS232 serial interface built-in. However, it support the PCMCIA interface by user the Dual PC Card Expansion Jacket (see picture). There are many serial I/O PC card available in the market. The one I am using is a Dual serial I/O PC card with 2 RS-232 COM ports manufactured by Socket™ (see Figure 4-3).

Figure 4-3 Accessories

The picture in the next page shows all the hardware needed for running the smart activity monitor (See Figure 4-4).

4.2.4 Development Environment

The application is designed to run on a Pocket PC. Although the Microsoft Pocket PC supports many different programming languages, we need to choose a most appropriate one. The application will process a big number of data in real time. The program speed must be concerned. Although the MT9 is language independent, the

MT9 SDK is written in C. The SDK has very good support in C++. Therefore I decide to develop the project in C++. And comparing to many other programming language like JAVA, C++ do have its advantage in speed.

Figure 4-4 Whole System

Microsoft provides a good IDE that can program the Window CE and latest version Pocket PC. The tool is called embedded Visual C++. The version I am using is 4.0. The interface of this version is quite similar to the Visual C++ 6.0, which is quite easy to use. Before we can program on the Pocket PC, there are two pieces of software must be installed in advance. One is the Pocket PC SDK. The Pocket PC SDK can be downloaded from the Microsoft's website. The version I am using for this project is Pocket PC SDK 2003. Another one is called Microsoft ActiveSync. This software is used to synchronies data between desktop and the Pocket. The complied program will be sent to the PDA via this software.

Before the project is deployed on the PDA, all the developing is on desktop. That is because I am designing and implementing the recognition algorithm first. This part of programming is platform independent. The desktop has many advantage in developing, therefore the early work are all on desktop. I am using embedded VC++ to develop the application on Pocket PC, so it is more convenient to do the early job in a similar environment. I am using Microsoft Visual C++.NET 2003 for developing the early algorithm. I also used the desktop development environment to develop the testing bed to test the accuracy of the algorithm. This will be discussed in detail in Chapter 5 Evaluation.

4.2.5 MT9 SDK

The MT9 SDK contains a proprietary algorithm developed by Xsens tailor-made to the MT9 that can accurately calculate absolute orientation in three-dimensional space from miniature rate of turn sensors (gyroscopes), accelerometers and magnetometers in real-time [26]. The orientation that calculated by the algorithm is claimed as driftless (See Figure 4-5).

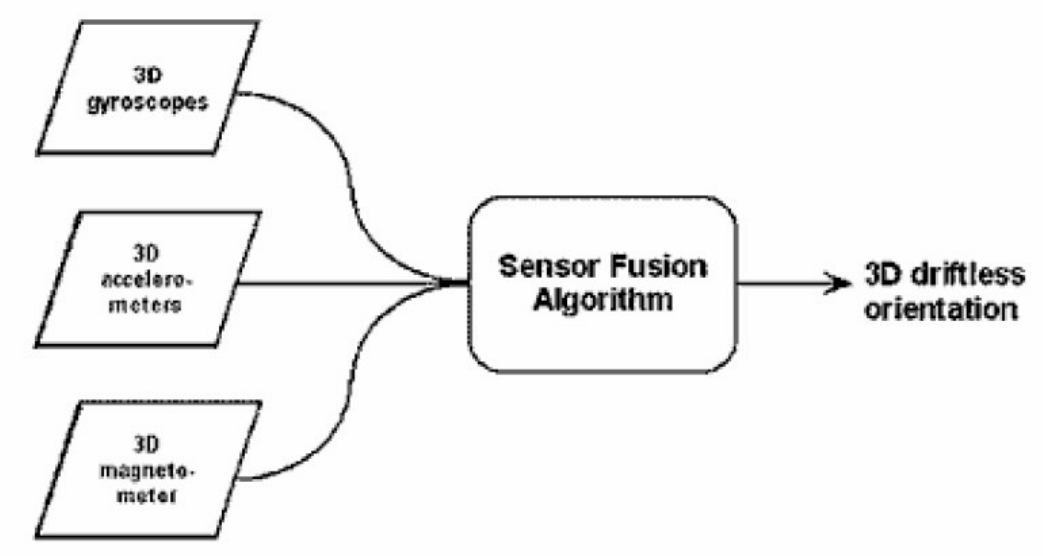


Figure 4-5 Sensor SDK Components

The MT9 SDK provides support so that we can integrate the capabilities of the MT9, attached directly to desktop or PDA. The functionality of the MT9 and the MT9 Software are made available trough a COM object. Through the motion tracker object we can set up the sensor as well as obtain the sensed data.

The MT9 SDK provides good support for Visual C++. It also provides the source code of some demo software that is using the running the MT9. However, there is not support for mobile device like Pocket PC on the company's website. The SDK downloaded from its website doesn't support the embedded VC++. Therefore I queried the company's technical support. And then they send me a Beta version of MT9 SDK for Pocket PC. There is one extra in the SDK file that must be copied to the iPAQ before it can run the motion tracker object.

4.3 MT9 Implementation

4.3.1 MT9 COM-Object

To make the MT9 data available in this project, there are a few steps need to follow. The first step is copying the necessary files from the MT9 SDK folder to the project folder. They are 'IMTObj.h', 'IMTObj.c', 'sink.h' and 'sink.cpp'. The next step is including these files in the pre-processor.

Now we can construct the Motion Tracker Object by:

```
IMotionTracker* pMT;
```

The next step is initializing the local motion tracker object.

```
    HRESULT hRes;
    hRes = CoInitializeEx(NULL, COINIT_MULTITHREADED);
    if (FAILED(hRes))
    {
```

```
                AfxMessageBox(_T("Could not initialize COM library!"));
                return FALSE;
        }
        // Create instance of MTObj COM object
hRes = CoCreateInstance(CLSID_MotionTracker, NULL, CLSCTX_INPROC_SERVER, IID_IMotionTracker, (void**) &pMT);
        if (FAILED(hRes))
        {
                CString csError;
        csError.Format(_T("Error %x in CoCreateInstance for MT object!"),hRes);
                AfxMessageBox(csError);
                return FALSE;
        }
```

The MT9 offers two mechanisms to retrieve the data available in the MT9 COM-Object. One is called the polling. This mechanism will continuously query the Motion Tracker Object. The user can define the own sampling rate of query the Object. Once the Object is queried, it will return the most recently calculated data. Another mechanism is called event. The event mechanism will not continuously query the Object. Instead, it will notify the sink as soon as there is a new data calculated. In this project, the event mechanism is used. Instead of using a predefined sampling rate to query the data, we rely on MT9's own sampling rate to return the data. Because the updating rate of monitor can be adjusted according to the sampling frequency of the MT9. As we are using the event mechanism, we need to initialize the sink object.

```
        // Create instance of internal sink object
        CMotionTrackerSink* pCob = NULL;
        if (SUCCEEDED(hRes))
        {
                // Create CMTObj Sink object to receive CMTObj events
                pCob = new CMotionTrackerSink(NULL,this,0);
```

```
        if (NULL != pCob)
        {
        // Save a pointer to the COMotionTracker IUnknown interface. Addref
        // because of this saved copy.
                m_pCMotionTrackerSink = pCob;
                m_pCMotionTrackerSink->AddRef();
        }
        else
                hRes = E_OUTOFMEMORY;
    }
```

Now we have finished the initialization step. It's time to start the MT9. Before we call to start the MT9, there are a few important parameters we can set, such as the com port number, output mode, etc. By default the acceleration data from the calibrated data will not be outputted. This also can be set by changing the Object parameter. Then the following piece of code much be called in order to connect the MT9 to the sink.

```
HRESULT MainPart::ConnectMTSink(void)
{
    HRESULT hr = E_FAIL;
    DWORD       dwKey;
    IConnectionPoint* pConnPoint;
    // Connect the MotionTrackerSink to the server MTObj COM source.
    if (!m_dwMotionTrackerSink)
    {
        // Get MotionTracker Sink connection point.
        pConnPoint = GetConnectionPoint(IID_IMotionTrackerEvents);
        if (NULL != pConnPoint)
        {
```

```
        // Connect the object in the server to the Motion Tracker Sink in this client
                hr = pConnPoint->Advise(m_pCMotionTrackerSink, &dwKey);
                        if (SUCCEEDED(hr))
                                m_dwMotionTrackerSink = dwKey;
                        // Release the connection point. We're done with it
                        pConnPoint->Release();
                        pConnPoint = NULL;
                }
        }
        return hr;
}
```

Finally we can call the following method to start the MT9 processing:

```
pMT->MT_StartProcess();
```

The MT9 SDK also provides the method that stops the MT9 processing. Once the MT9 COM-Object is no longer needed, there is a destruction method will be called.

4.3.2 MT9 Data Retrieving

Due to the event mechanism, once the new data is calculated it will be stored in a safearray. Therefore the MT9 data can be retrieved from the safearray. Before we access the safearray, we need to create a local variable. Therefore we can copy the value from the safearray to the local variable. E.g. fData[9] = 0; The following code shows how to access the safearray.

```
// Variable of VARIANT type to retrieve data from output of COM object

VARIANT buffer;
// Pointer to array data
```

```
void* pDest;
m_pMT->MT_GetOrientationData(&nNew, &buffer); // Or MT_GetCalibratedData
if (nNew == MT_NEWDATA) {
        //check if array is empty
        if (buffer.vt != VT_EMPTY)
                SafeArrayAccessData(buffer.parray, &pDest);
        // Copy data from the VARIANT array to the local fData array
        memcpy(fData, pDest, buffer.parray->rgsabound->cElements);
        //invalidate pointer
        SafeArrayUnaccessData(buffer.parray);
        //Data now is copied to the fData
        }
}
else if (nNew != 0)
//Check error code (nNew) when no new data
```

4.4 Activity Recognition Implementation

4.4.1 Feature Extractor

In order to extract features, we need to have a container to store the MT9 sensed data. The simple way to implement to the container is using an array. The size of the array is according to the monitor's refresh rate i.e. if the MT9's sampling rate is 100Hz and the monitor's refresh rate is 1Hz, then the size of the array should be 100. This simple method doesn't work! If we define the array size according to the monitor's refresh rate, then we will either get to few data to examine or have too low refresh rate. This problem can be solved by using a big sized array with a counter. The size

of array must be big enough to get all the necessary information. The number of counter is according to the monitor's refresh rate. As the project is designed to monitor the real time activity, it is more important to know the current activity. Therefore the examination will be carried out backward, which is from the most current received data first. E.g. if the monitor's refresh rate is every 3 second, then the counter number will be 300. In this case, if we define an array size 900, then there are 300 sets of data for the first examination index from 299 to 0, 600 for the second index from 599 to 0 and 900 for the third index from 899 to 0. However, such design always has ***n*** number of examinations, that don't have enough data sets, for every ***N*** number of examinations. In the case of the example above, there is one examination that only has 300 data sets, every 3 examinations. We probably cannot completely resolve the problem, but we can definitely improve it by changing the data structure.

Circular Buffers are used for data transfer between two processes. The Producer process places items into the Circular Buffer and the Consumer process removes them. Initially there is a tail and a head on the buffer; they are both at index 0. The index of head will be increased along with the producer. The producer keeps writing the data sets into the buffer, the counter, which is the timer, will interrupt the producer every certain amount of time. When the buffer is full, the producer will overwrite the buffer. During the interrupt, the consumer will examine the data sets from the head to tail. If overwrite happens, the tail will be offset. E.g. if the head is 299, the tail is 600 and the overwrite is true, then the data sets on the buffer will be index from 299 to 0 as well as index from 899 to 600. Both ***samplingrate*** and ***samplewindow*** are pre-defined constant, one indicates the monitor's refresh rate and the other indicates the size of the buffer.

The next step is extracting features from the buffer. According the features selected in the design stage, these features can be extracted by searching the cycle of step on the pitch graph. The hill climbing algorithm is used to search the graph. The searching process won't begin until it adjacent element has higher pitch value. However, some noise must be filtered. It will check the following ***n*** number of elements to see if it keeps increasing. If the start of increasing is found, it will mark the index. And then it will begin the hill climbing. The hill climbing includes two

processes. One is searching for the peak; another is searching for the end of the cycle. The algorithm can be described as follows:

1. Start with current-state = initial-state.
2. Until current-state = goal-state OR there is no change in current-state do:
   a) Get the successors of the current state and use the evaluation function to assign a score to each successor.
   b) If one of the successors has a higher (lower) score than the current-state then set the new current-state to be the successor with the best score.

Searching for the peak begins at the start of the cycle; the index of the peak is initially set as the index of start plus one. The temporary peak will compare with its left side neighbor. If its neighbor is larger than itself, the index will be increased by one. If the left neighbor is smaller than itself, it will check the flowing ***n*** number of element in order to filer noise. If all the left elements are smaller, then the peak is at current index. Otherwise, the temporary peak index will be the left side elements, which has larger value than the current temporary peak. Once the peak is found, it will begin to search for the end of the cycle. Searching for the end is similar as searching for the peak. It starts from the peak index, then shifting to the left searching for the index of minimum value on the graph. The function will return the index of start, peak and end of the cycle as well as if the offset is allowed.

After the cycle is found another filter will be applied. This filter will quickly calculate the amplitude and the length of the wave. Noise will be filtered out, i.e. tiny waves.

The eligible wave will be carried out calculation. The direct calculations include frequency, amplitude. Then a linear search is used to get the local maximum and minimum of the acceleration. The slope ratio is just the length between index of start and the peak against the length between peak and the end.

4.4.2 Unsupervised Classifier

Basically, ***k-mean*** algorithm is used here to calculate the most suitable value of each feature in order to represent one activity. Firstly, an object array is defined to store

the features that extracted by the Feature Extractor (see last section). The size of the array presents how long the monitor will be trained. This will be discussed in the Chapter 5 Evaluation. The code below shows the training on walking. Once the user select to train on walking, he/she begin to walk. As soon as the feature of walking is extracted, the value of each feature will be recorded locally.

Once the array is filled, the training on that activity is finished. Before we begin to train the next activity, the centroids of that activity must be calculated.

To calculate the centroids for each feature, we begin with initializing the centroids with a random number. A for loop then is used to go through all the elements in the array. The mean value of each feature will be calculated accordingly and replace the current centroid. When the 'for loop' is finished, the difference between the previous centroid and the current centroid will be calculated by using a distance function. If the difference is above a threshold, the 'for loop' will be gone through again. The purpose of doing this is increasing the accuracy of the centroids. The code below shows the entire process. Once the difference is small enough, the centroids are found.

```
void MainPart::UnsuperviousLearning(Feature learningSet[], Feature *centriod)
{
centriod->setFrequency(((float) rand())*(((float) 5.0)/((float) RAND_MAX)));
centriod->setAmplitude(((float) rand())*(((float) 100.0)/((float) RAND_MAX)));
centriod->setAcceleration(((float) rand())*(((float) 70.0)/((float) RAND_MAX)));
centriod->setRatio(((float) rand())*(((float) 2.0)/((float) RAND_MAX)));
centriod->setAccRatio(((float) rand())*(((float) 1.0)/((float) RAND_MAX)));
        float tempFreq = centriod->getFrequency();
        float tempAmp = centriod->getAmplitude();
        float tempAcc = centriod->getAcceleration();
        float tempRatio = centriod->getRatio();
        float tempAccRatio = centriod->getAccRatio();
        float difference = 999999.9999f;
```

```
        while(difference > 10.0){

                for ( int i = 0; i < TrainingSampleNumber; i++)

                {

                        float x = learningSet[i].getAmplitude();

                        float y = learningSet[i].getRatio();

                        tempFreq = (tempFreq * ((float)i) + ((float)learningSet[i].getFrequency())) / ((float) i + 1.0f);

                        tempAmp = (tempAmp * ((float)i) + ((float)learningSet[i].getAmplitude())) / ((float) i + 1.0f);

                        tempAcc = (tempAcc * ((float)i) + ((float)learningSet[i].getAcceleration())) / ((float) i + 1.0f);

                        tempRatio = (tempRatio * ((float)i) + ((float)learningSet[i].getRatio())) / ((float) i + 1.0f);

                        tempAccRatio = (tempAccRatio * ((float)i) + ((float)learningSet[i].getAccRatio())) / ((float) i + 1.0f);

                }

                difference = DistanceTo(tempFreq, tempAmp, tempAcc, centriod->getFrequency(), centriod->getAmplitude(), centriod->getAcceleration());



                centriod->setFrequency(tempFreq);

                centriod->setAmplitude(tempAmp);

                centriod->setAcceleration(tempAcc);

                centriod->setRatio(tempRatio);

                centriod->setAccRatio(tempAccRatio);

        }

}
```

4.4.3 Recognition Decision Making

After the classifier is built, we can now recognize the activity according to its features. The decision making rule is very simple, which is 'shortest distance'. However, there are 5 different features, with different level of importance. The question is how to combine them all together, so we can have one single rule to make decision.

The features are weighted as follow:

Frequency	15%
Amplitude	20%
Acceleration	20%
SlopeRatio	30%
AccRatio	15%

There are 3 different activities excite in this project. The classifier we have here contains 3 different sets of centroids. Now it's time to compare the distances. Once the features of current activity are extracted, they will be compared to each set of centroids. E.g. if the current frequency is f, then f will be compared to the frequency centroid of walking, running and cycling separately. If the distance between f and walking centroid is closest, then we believe the probability of walking on frequency is 100%. The probability of other activities will be calculated against distance to walking. The following code is the 'if statement' of when distance to walking in the frequency feature space is shortest, then the probability of walking ***WalkProb*** is equal to itself plus 100% under the weight of feature frequency. In this case the probability of running and cycling can be calculated by using the same method. The probability of rest activity is equal to the ratio of distance to the walking against the distance to the target activity multiple to the weight of frequency. Therefore, the lower distance will get higher probability.

```
if(distanceFrequencyW < distanceFrequencyR && distanceFrequencyW < distanceFrequencyC)

{

                    WalkProb = WalkProb + FrequencyWeight;

                    RunProb = RunProb + ((distanceFrequencyW / distanceFrequencyR) *                     FrequencyWeight);

                    CycleProb = CycleProb + ((distanceFrequencyW / distanceFrequencyC)                     * FrequencyWeight);

}
```

The other features will use the same method to calculate the probability of each activity on the particular feature. The total probability of each activity is cumulated. The activity with highest probability will be returned as recognized activity.

4.5 Energy Expenditure Implementation

This part of implementation is about using programming language to represent the physiology equation. In this stage, we have already known what kind of activity the user is doing. By loading the user's profile, we can also get the necessary personal information about the user. Before we apply the energy expenditure equation, there is one more thing need to do, which is calculate the speed.

4.5.1 Speed Calculation

Base on the speed equation designed in the previous chapter (Chapter 3 Design), the implementation of speed calculation is simple. There are two functions, one is used to calculate the speed of walking and running, another is used to calculate the speed of cycling. The source code below is the function that calculates the speed of walking and running. There are two parameters are passed in. They are frequency and amplitude, which are extracted from the activity. To calculate the speed here, we

need to know the distance of movement. There is no way to measure the distance directly. All we can do is based on current available information to predicate the distance. The equation of predicating the step length has been explained in Section 3.2.3. Once the distance is known, we can use the speed equation ***v =d/t*** to calculate the speed.

```
float MainPart::GetSpeed(float freq, float amp)
{
        float s;
        double stepLength;
        float angle = float(( amp * PI / 180 ) / 2 );
        double Length = (height / 100) * HeightWeight;
        stepLength = 2 * ( sin(angle) *  Length );
        s = float(float(stepLength) * freq * 3.6);
        return s;
}
```

The speed calculation for the cycling is even easier. All we need to know is the perimeter of the wheel. According to the frequency of running wheel, we can know the length of movement (see code below).

```
float MainPart::GetCyclingSpeed(float freq)
{
        float s;
        s = float(freq * WheelPerimeter * 3.6);
        return s;
}
```

4.5.2 Energy Expenditure Calculation

Base to the equation, implementation the calculation of energy expenditure for walking and running is very straight forward. As we don't consider grade level when doing activity in this project, all we need to calculate is the horizontal component and the resting component. The following source code is function of calculate the energy expenditure for walking.

```
float MainPart::GetWalkingEnergy()
{
        float e;
        double horizontalComponent = ( speed / 3.6 ) * 60 * 0.1;
        double RestingComponent = 3.5;
        double vo2 = RestingComponent + horizontalComponent;
        e = float((float(vo2) * weight / 1000) * 5.05);
        return e;
}
```

The equation for calculating energy expenditure of cycling is also simple. However, there is a problem when implement it. The problem occurs when the speed of cycling is low. The result of energy expenditure could get a zero of a negative number.

If we graph the equation, we will get the following diagram (see Figure 4-6). We can find a clear threshold that divide the entire graph. If the speed is above the threshold, we can still use the cycling energy equation as normal. However, if the speed is below the threshold, a new liner equation will be applied. We calculate the energy expenditure when the speed is on the threshold. And then we connect this point and the origin, as a result we will get a line. The energy expenditure on the low cycling speed will be found on this line according to the speed.

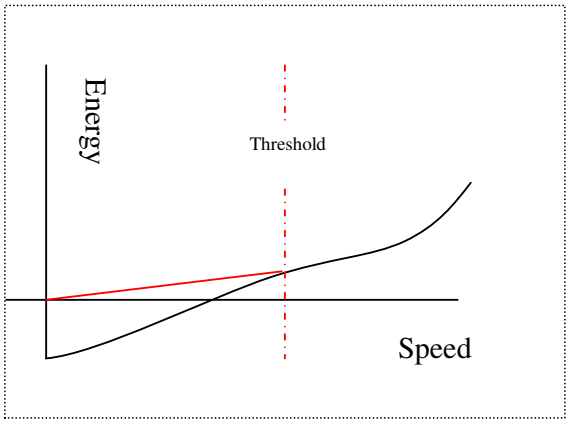


Figure 4-6 Analysis on Low Speed Cycling

4.6 File System Implementation

The project is about monitoring human's activities as well as calculating the energy expenditure during the activities. The purpose of having a file system here is making the monitor easier to use. Due to the nature of unsupervised classifier, the monitor must be trained by the user before it can be used. In order to calculate user's energy expenditure, the monitor needs to know some personal information about the user, e.g. weight. Without file system these information must be inputted to the monitor every single time the user need to user it. Such system is very inconvenient to use, especially the training process, which takes about 3-5 minutes. Therefore a file system is suggested here to make the user's life easier.

The file system here is simple with no security issue involved. Therefore sequential-access file is chosen to implement here. Sequential files are so named because data is stored on a first come, first served basis.

The file system consists three parts. The first part is creating the file. This part happens in the new user registration. The second part is verification. This part happens in the user login. The last part is loading the file. This part happens as soon as the user is verified.

The system will ask the new user to fill a form, including the user name and some personal details. Then it will ask the user to do different activity in order to train the classifier. Once everything is finished, it's time to create the file. All the files are stored in a folder called "UserData". The file name is the username. And then all the information about the user, including the user personal details and classifier data, will be written to the file in a specific sequence.

The login verification is about checking the files in the "UserData" folder. Because the system is using the user name as every file's name, therefore once the user input a user name in the Login dialog, the system can simply check if such file exists in the folder. If the file is found, the system will begin to load the file. If the file name is not found, it will signal the user that the user name is wrong.

Loading file is similar to writing a file. It begins with creating local variables. Then copy the value from the file to local variables. There is one rule, that the sequence of reading the file must be exactly same as it is written. Otherwise, it will get the wrong value.

4.7 Additional Features Implementation

4.7.1 Push graph

Once the current energy expenditure is calculated, it will be displayed on the monitor. Therefore the user can always know how intensity he/she is doing. The initial design of the monitor is displaying the energy expenditure value directly on the monitor as number. However, the problem is the screen of PDA is very small. There are many different information need to display on such small screen at the same time. So it is not possible to use huge font to display the energy expenditure. Another problem is it is very difficult to read for users while he/she is doing activity. It is also very dangerous if the user pay too much attention on reading while he/she is doing outdoor activity. Therefore a more intuitive display method is required here.

A 2D Push Graph is decided to use in here. There is no existing push graph control available in the embedded VC++. I can either create my own custom control or use a third party control. Because creating a complex custom control is beyond the scope of this project, so I decide to use a third party control. The push graph control I am using is created by **Stuart Konen**. It is published on The Code Project [27].

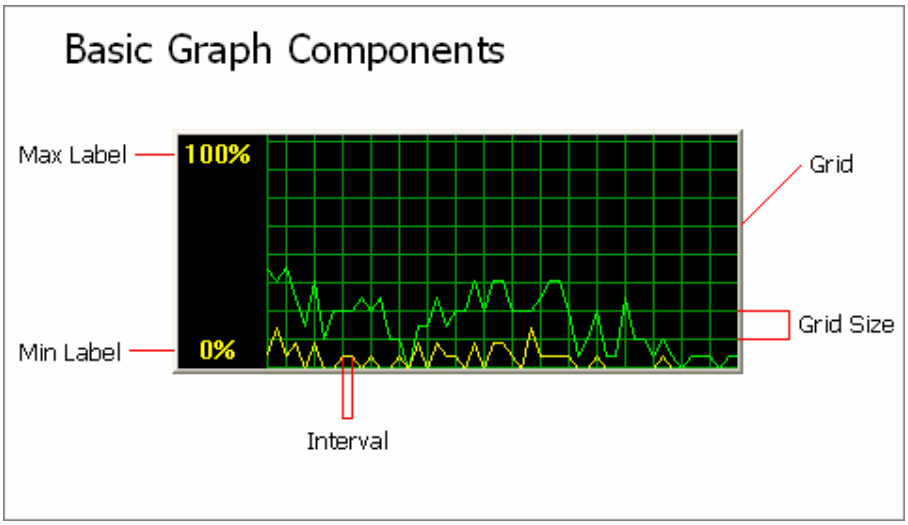


Figure 4-7 Basic Push Graph Components [27]

The picture above is the basic components of the push graph. The graph is quite similar to the push graph from the Windows Task Manager. Once the monitor is started, a line will be added into the graph. The boundary of the graph is set between 0 Kcal/h and 700 Kcal/h. The graph refresh rate is same as the monitor's refresh rate. Therefore every time the monitor is refreshed, a new value on the graph will be calculated and a line will be drawn as well. The picture on the left is the screenshot of the push graph implemented on the PDA.

Figure 4-8 Push Graph Screenshot

4.7.2 Audio Assistant

The purpose of having audio assistant in this project is making the monitor easier to use. The monitor is quite difficult to use for the new user, as it requires the user to train every single activity once. Without audio assistant, the user must keep watching the screen and waiting for different commands as text. For the first time user, it is very difficult to concentrate on the activity and watch the command at the same time. It is the same as the first time you go the gym. You will feel panic without an instructor tell to what to do. Therefore I implement an audio assistant that helps the new user to finish the whole training process.

The implementation involves two jobs. The first job is record the voice. There is no professional audio record facility available to this project. Therefore I used the Windows Recorder to collect my voice as the instruction. The voices are saved as Wave files.

The second job is playing these Wave files at the right time. It is very simple to play Wave file under MFC. All we need to do is call the PlaySound function. The code below is an example of playing the "ask user to run" command from the "Sound" folder.

```
PlaySound(_T("Sound/trunbegin.wav"), NULL, SND_ASYNC | SND_FILENAME);
```

Chapter 5: Evaluation

In this chapter, I shall evaluate the design and implementation of the project. All the problems encountered during the course of this project will be discussed. A series of tests will be carried out in order to check how well the project has achieved. The method of testing and the testing results will be documented in this chapter.

5.1 Testing Bed

5.1.1 Goal

The purpose of testing bed is testing the accuracy of activity recognition. Therefore the goal here is designing a testing software that can run the activity detector, which has been implemented, as well as evaluate the results.

5.1.2 Offline Processing

The testing bed will use offline processing of MT9 data. Instead of processing the MT9 data from the senor directly, it will read the MT9 data from binary files.

It is not possible to design a testing bed that can tell the accuracy of the activity detector base on the direct sensed data. If we are using online processing, how can the testing bed tell us if the recognition results are correct or not? If we can implement such testing bed, which means we can implement a 100% accuracy activity detector. And then the testing is meaningless. Therefore if we want the testing bed to be online processing, then we have to judge the result manually. The

problem of such method is taking too much time. Every online test takes about 15 to 20 minutes. We expect to takes as many tests as possible here so we can get a convictive evaluation. If a single test takes such long time, this will make the evaluation out of project plan. Another draw back of online processing for testing bed is it is very inflexible. All the test data can only be used once. If we change any thing in the program, we have to do the whole tests once again from the very beginning. Therefore we use the offline processing for the testing bed. The idea of offline processing is letting us to label the binary files in advance. Then we can compare the recognition results with the label, so we can know the accuracy. E.g. suppose we have a binary file that is 30 seconds walking. If the activity detector returns 10 walking (refresh rate of the monitor is every 3"), then the accuracy is 100% on this file.

The binary files can be obtained by using the MT9 Demo Software while the testing user is doing activities. The MT9 SDK also provides the support for offline processing.

5.1.3 Data Sets

There are 10 volunteers contribute to this project as testing users. The data sets used for testing are collected from these 10 people in the GYM. Every testing user will be asked to walk, run and cycle for certain period of time. At the end of the day, there will be 3 sets of walking data, 3 sets of running data and 2 sets of cycling data for every testing user.

10 * (3 Walking + 3 Running + 2 Cycling)

5.1.4 Manual Testing Bed Design

According to the structure of data sets, we can get following table to represent data sets for one testing user.

Walking	Running	Cycling
W1	R1	C1
W2	R2	C2
W3	R3	

Table 5-1 Table of Data Sets Structure

The first thing need to do is categorizing the data sets. All the files will be stored according to its type of activity. Therefore, we can easily know what kind of activity in the program in advance. All the folders that belong to one person will be stored under this person's folder.

All the available testing users will be shown on the testing bed dialog as a choosing option. Once the testing user is chosen, all the files will be shown on the dialog box for choosing. Due to the nature of the unsupervised learning, before the system can recognize different activities, it requires training on different activities. To finish a test we must select the files to be the training data and a file to be the testing target. According to the table above, we can combine 18 different classifiers as following table:

W1+R1+C1	W1+R1+C2	W2+R1+C1	W2+R1+C2	W3+R1+C1	W3+R1+C2
W1+R2+C1	W1+R2+C2	W2+R2+C1	W2+R2+C2	W3+R2+C1	W3+R2+C2
W1+R3+C1	W1+R3+C2	W2+R3+C1	W2+R3+C2	W3+R3+C1	W3+R3+C2

Table 5-2 Table of All Possible Classifiers

Each classifier can test the rest 5 data files. E.g. Classifier W1+R1+C1 can test W2, W3, R2, R3 and C2.

We are using the offline processing here. Instead of reading the data from MT9 sensor, the binary files will be simulated as MT9 sensed data. The selected training files will be used to train the classifier. And then it will try to recognize the selected target file. The testing bed will read the target file from the beginning to the end. The first 6 seconds of data will be ignored. That is because people cannot do the

activity as soon as the test begins. When the file reaches the end, the recognition results will be judged against the target file's label.

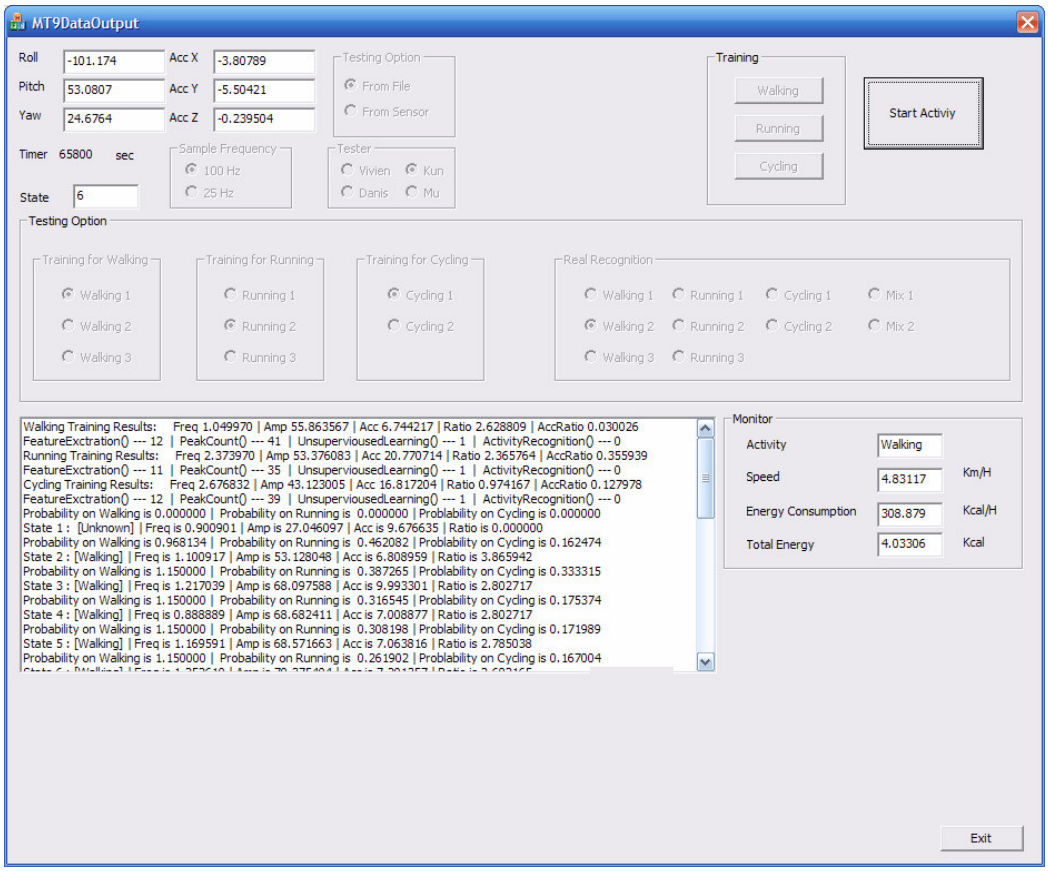


Figure 5-1 Manual Testing Bed Screen Shot

The screen shot above is the manual testing bed's interface. All the options are displayed as Radio Button. Once the option is chosen, the option group will be disabled. The recognition results will be displayed in the list box. The test will be stopped automatically when the file is end.

5.1.5 MT9 Sampling Frequency

The MT9 sampling frequency can be set between 25Hz to 512Hz. The default sampling frequency is 100Hz. Before the project is deployed on the PDA, it is using

the default setting. However, when monitor is implemented on PDA, I found that the iPaq 5500's processing ability cannot handle 100Hz MT9 sampling frequency. The project is supposed to be real time. Although the monitor has a fixed refresh rate, the result should reflect the last few second until now. The 100Hz sampling rate will make the monitor has big delays, which is not what we expect for this project. Therefore the activity detector's accuracy on reduced sampling rate should also be evaluated.

All the data sets collected early on are recorded at 100Hz. It is not possible to collect another group of data sets at this stage. Therefore I am still using the 100Hz binary files. But instead of reading the entire files, I read every 4^{th} data. This will also make the 25 data readings every second.

5.1.6 Training Data Size

The size of training data will affect how long does the user need to train the monitor. If the size is too small, the training might be inadequate, which will affect the recognition results. If the size is too big, the user will spend too much time in training, which will affect the usability of the application. In this section I am going to find out a most appropriate size for the training data.

I ran a small test on the manual testing bed. I choose one of testing user's data sets. And I randomly pick up 3 files as the training file and one file as the testing target file and begin the test. Before I run the program, I change the number of exemplars as 3. And then increase 1 every time until the result is stable. The following table shows the testing results:

Exemplars	Accuracy
3	50%
4	62.5%
5	66.7%
6	70.8%
7	79.2%
8	87.5%
9	91.7%

10	91.7%
11	95.8%
12	95.8%
13	95.8%
14	95.8%

Table 5-3 Results of Exemplar Size Testing

The results begin to reach its best when I using 11 exemplars. In the purpose of giving the best recognition I am using 12 exemplars as my training data size. Because the refresh rate of the monitor is every 3 seconds, therefore 12 exemplars will takes about 36 seconds. The overall net training period will be 108 seconds.

5.1.7 Automatic Testing Bed Design

According to the data sets we have, there are 3 walking files, 3 running files and 2 cycling files. These files can combine 18 different classifiers. Each classifier can test 5 different target files. Therefore we can have 18 * 5 = 90 different test for each testing user. We can 10 volunteers as testing users. So we can have 90 * 10 = 900 different tests. If we want to test different sampling rate, we will have ***N*** number of 900 tests. This is an impossible number for manual testing bed. Therefore an automatic testing bed is desired.

The automatic testing bed still have some options need to choose in advance. The target testing user and the sampling frequency must be chosen before the test. Once the test starts, it will train all the possible classifier at the beginning. These classifiers will be stored in local memory. Once the training is finished, it will take each classifier in turn to recognize the rest files. After the 900 tests, the program will give the results before it stops. Figure 5-3 is a screenshot of automatic testing bed, which has less options comparing to the manual version.

I made some change on the automatic testing bed. Instead of testing the entire files, it will only read the 30 seconds from each file. This will make the testing result more convictive. Figure 5-2 is an example of automatic testing results from the testing bed.

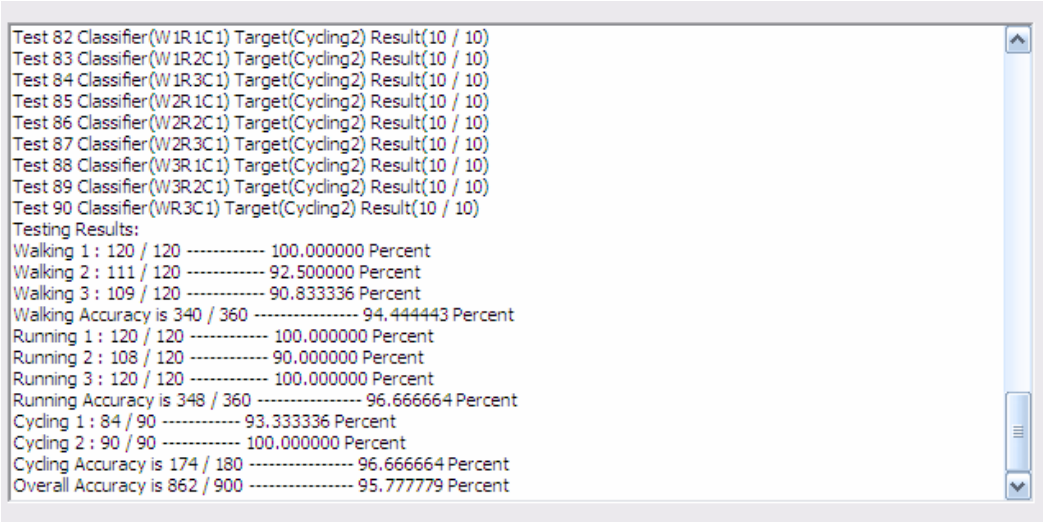


Figure 5-2 Example of Testing Results From Automatic Testing Bed

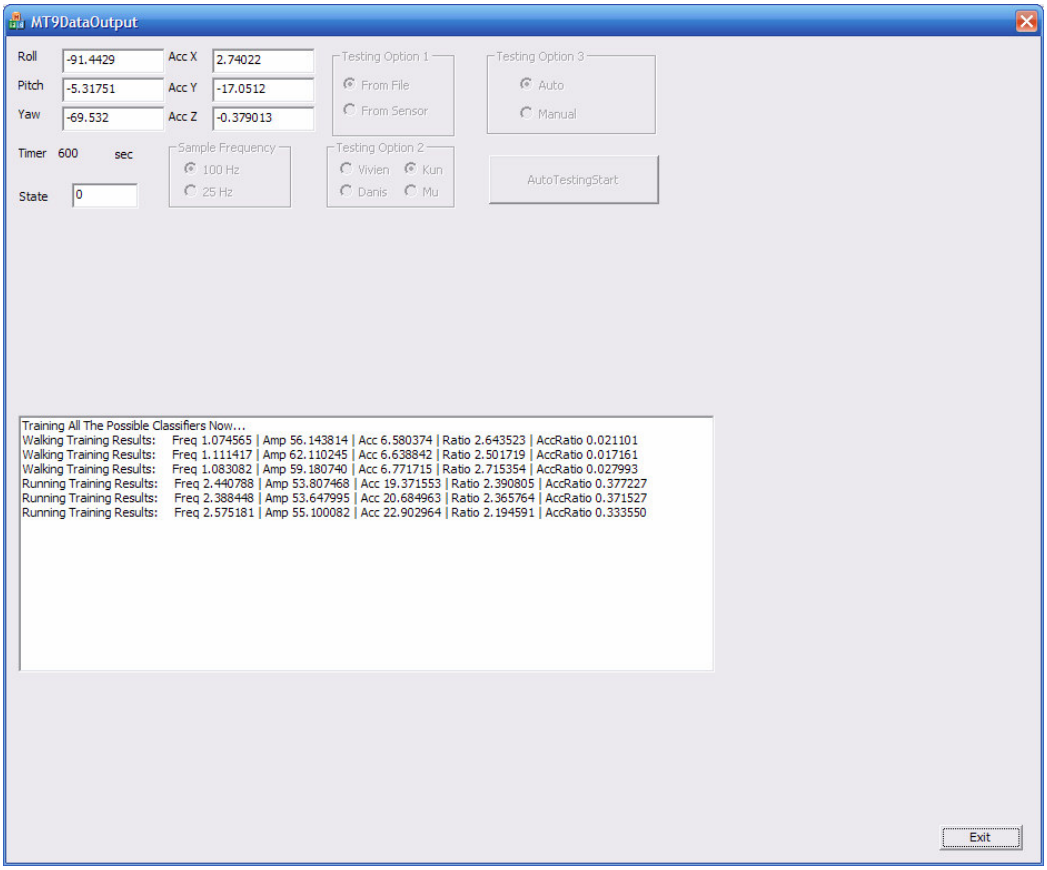


Figure 5-3 Automatic Testing Bed Screen Shot

5.2 Testing Results

In this section all the testing results will be documented. I will give the testing results from the manual testing bed first. According the project plan, I only did manual tests for 3 testing users. And then I will show the testing results for all the testing users from the automatic testing bed.

5.2.1 Testing User Profile

There are 10 volunteers have contributed to this project as the testing users. There are four female testing users age between 24 to 27 years old. Their heights are between 160cm and 166 cm. Their weights are between 50kg and 70kg. The rest 6 male volunteers are aged between 19 and 30 years old. Their heights are between 173 cm and 188 cm. Their weights are between 60kg and 85kg.

5.2.2 Manual Testing Bed Results

The data sets from 3 persons have been manually tested. I give the result details for the first testing user as an example. The rest people I only show its summary. The tests involve both 100Hz MT9 sampling rate and 25Hz.

Person 1 (100Hz)

Training Data Sets Results

File	Frequency	Amplitude	Acceleration	Ratio	AccRatio
Walking 1	1.48	14.56	19.45	0.42	0.049
Walking 2	1.78	19.24	23.44	0.42	0.021
Walking 3	1.78	20.75	22.93	0.33	0.018
Running 1	2.23	35.38	38.01	0.39	0.050
Running 2	2.47	37.32	39.91	0.40	0.064
Running 3	2.36	30.09	26.49	0.37	0.068
Cycling 1	1.76	15.76	4.57	0.76	0.205

Cycling 2	1.81	19.60	4.86	0.87	0.192

Walking 1 Data Sets

Time Period: 48.93”

Total States: 16

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W2+R1+C1	11/16	W2+R2+C1	13/16	W2+R3+C1	10/16
W3+R1+C1	12/16	W3+R2+C1	12/16	W3+R3+C1	10/16
W2+R1+C2	12/16	W2+R2+C2	13/16	W2+R3+C2	12/16
W3+R1+C2	12/16	W3+R2+C2	12/16	W3+R3+C2	11/16
Summary	140 / 192				**72.9%**

Walking 2 Data Sets

Time Period: 54.01”

Total States: 18

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R1+C1	15/18	W1+R2+C1	16/18	W1+R3+C1	15/18
W3+R1+C1	11/18	W3+R2+C1	15/18	W3+R3+C1	11/18
W1+R1+C2	15/18	W1+R2+C2	16/18	W1+R3+C2	15/18
W3+R1+C2	11/18	W3+R2+C2	15/18	W3+R3+C2	11/18
Summary	166 / 216				**76.9%**

Walking 3 Data Sets

Time Period: 56.81”

Total States: 19

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R1+C1	13/19	W1+R2+C1	17/19	W1+R3+C1	06/19
W2+R1+C1	19/19	W2+R2+C1	19/19	W2+R3+C1	15/19
W1+R1+C2	12/19	W1+R2+C2	17/19	W1+R3+C2	06/19
W2+R1+C2	19/19	W2+R2+C2	17/19	W2+R3+C2	15/19
Summary	175 / 228				**76.8%**

Running 1 Data Sets

Time Period: 45.25”

Total States: 15

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R2+C1	13/15	W1+R3+C1	08/15	W2+R2+C1	15/15
W2+R3+C1	15/15	W3+R2+C1	15/15	W3+R3+C1	15/15
W1+R2+C2	13/15	W1+R3+C2	08/15	W2+R2+C2	15/15
W2+R3+C2	15/15	W3+R2+C2	15/15	W3+R3+C2	15/15
Summary	162 / 180				**90.0%**

Running 2 Data Sets

Time Period: 81.71”

Total States: 27

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R1+C1	18/27	W1+R3+C1	12/27	W2+R1+C1	22/27
W2+R3+C1	19/27	W3+R1+C1	27/27	W3+R3+C1	27/27
W1+R1+C2	18/27	W1+R3+C2	12/27	W2+R1+C2	22/27
W2+R3+C2	19/27	W3+R1+C2	27/27	W3+R3+C2	27/27
Summary	250 / 324				**77.2%**

Running 3 Data Sets

Time Period: 81.16”

Total States: 27

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R1+C1	21/27	W1+R2+C1	21/27	W2+R1+C1	24/27
W2+R2+C1	24/27	W3+R1+C1	25/27	W3+R2+C1	25/27
W1+R1+C2	21/27	W1+R2+C2	22/27	W2+R1+C2	24/27
W2+R2+C2	24/27	W3+R1+C2	26/27	W3+R2+C2	25/27
Summary	282 / 324				**87.0%**

Cycling 1 Data Sets

Time Period: 93.96”

Total States: 31

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R1+C2	31/31	W2+R1+C2	31/31	W3+R1+C2	31/31
W1+R2+C2	31/31	W2+R2+C2	31/31	W3+R2+C2	31/31
W1+R3+C2	31/31	W2+R3+C2	31/31	W3+R3+C2	31/31
Summary	279 / 279				**100%**

Cycling 2 Data Sets

Time Period: 73.12”

Total States: 24

Classifier	Accuracy	Classifier	Accuracy	Classifier	Accuracy
W1+R1+C1	22/24	W2+R1+C1	22/24	W3+R1+C1	22/24
W1+R2+C1	22/24	W2+R2+C1	22/24	W3+R2+C1	22/24
W1+R3+C1	20/24	W2+R3+C1	20/24	W3+R3+C1	20/24
Summary	192 / 216				**88.9%**

Person 1 Summary

Walking	481 / 636	75.6%
Running	694 / 828	83.8%
Cycling	471 / 495	95.2%
Overall	1646 / 1959	**84.0%**

Table 5-4 Manual Testing Results for Person 1 at 100Hz

Person 2 (100Hz)

Training Data Sets Results

File	Frequency	Amplitude	Acceleration	Ratio	AccRatio
Walking 1	1.70	25.34	9.08	2.06	0.352
Walking 2	1.68	11.36	30.83	1.99	0.362
Walking 3	1.58	45.63	12.10	2.43	0.091
Running 1	2.34	47.41	30.83	1.99	0.362
Running 2	2.39	53.68	22.34	1.88	0.379
Running 3	2.46	50.64	28.21	1.94	0.393
Cycling 1	2.24	33.95	10.78	1.09	0.125
Cycling 2	2.48	34.13	15.66	1.03	0.140

Person 2 Summary

Walking	483 / 792	61.0%
Running	726 / 756	96.0%
Cycling	243 / 243	100%
Overall	1452 / 1791	**81.1%**

Table 5-5 Manual Testing Results for Person 2 at 100Hz

Person 3 (100Hz)

Training Data Sets Results

File	Frequency	Amplitude	Acceleration	Ratio	AccRatio
Walking 1	1.05	55.86	6.74	2.63	0.030
Walking 2	1.08	62.66	6.76	2.50	0.023
Walking 3	1.08	59.43	7.01	2.72	0.023
Running 1	2.44	53.99	19.37	2.39	0.377
Running 2	2.37	53.38	20.77	2.37	0.356
Running 3	2.55	54.67	23.01	2.19	0.321
Cycling 1	2.68	43.12	16.82	0.97	0.128
Cycling 2	3.23	45.50	27.18	1.00	0.117

Person 3 Summary

Walking	768 / 768	100%
Running	851 / 876	97.1%
Cycling	414 / 414	100%
Overall	2033 / 2058	**98.8%**

Table 5-6 Manual Testing Results for Person 3 at 100Hz

100Hz Summary

Walking	**1732 / 2196**	**78.9%**
Running	**2271 / 2460**	**92.3%**
Cycling	**1128 / 1152**	**97.9%**
Overall	**5131 / 5808**	**88.3%**

Table 5-7 100 Hz Manual Testing Results Summary

Person 1 (25Hz)

File	Frequency	Amplitude	Acceleration	Ratio	AccRatio
Walking 1	1.50	12.90	13.74	0.39	0.110
Walking 2	1.77	16.98	23.44	0.41	0.053
Walking 3	1.75	19.00	15.87	0.34	0.039
Running 1	2.23	29.09	26.14	0.40	0.098
Running 2	2.56	29.90	26.84	0.40	0.086
Running 3	2.36	25.27	18.62	0.37	0.158
Cycling 1	1.70	15.48	4.39	0.78	0.141
Cycling 2	1.86	18.66	4.59	0.87	0.233

Person 1 Summary

Walking	385 / 636	60.5%
Running	712 / 828	86.0%
Cycling	438 / 495	88.5%
Overall	1535 / 1959	**78.4%**

Table 5-8 Manual Testing Results for Person 1 at 25Hz

Person 2 (25Hz)

File	Frequency	Amplitude	Acceleration	Ratio	AccRatio
Walking 1	1.59	23.30	6.38	1.99	0.194
Walking 2	1.69	25.93	9.00	2.37	0.042
Walking 3	1.57	40.36	10.37	2.49	0.019
Running 1	2.39	40.94	14.95	2.07	0.201
Running 2	2.40	47.35	15.60	1.90	0.239
Running 3	2.42	45.44	13.40	1.91	0.223
Cycling 1	2.21	32.87	10.93	1.12	0.105
Cycling 2	2.60	31.30	14.06	1.07	0.106

Person 2 Summary

Walking	443 / 792	55.9%
Running	670 / 756	88.6%
Cycling	240 / 243	98.8%
Overall	1353 / 1791	**75.5%**

Table 5-9 Manual Testing Results for Person 2 at 25Hz

Person 3 (25Hz)

File	Frequency	Amplitude	Acceleration	Ratio	AccRatio
Walking 1	1.08	51.57	6.08	2.62	0.030
Walking 2	1.09	58.88	5.48	2.57	0.012
Walking 3	1.08	53.96	6.15	2.79	0.019
Running 1	2.44	47.80	14.90	2.38	0.160
Running 2	2.39	47.92	14.49	2.36	0.167
Running 3	2.63	49.44	15.08	1.92	0.150
Cycling 1	2.66	40.00	15.67	0.98	0.101
Cycling 2	3.21	40.45	25.46	0.92	0.096

Person 3 Summary

Walking	740 / 768	96.4%
Running	849 / 876	96.9%
Cycling	405 / 414	97.8%
Overall	1994 / 2058	**96.9%**

Table 5-10 Manual Testing Results for Person 3 at 25Hz

25Hz Summary

Walking	**1568 / 2196**	**71.4%**
Running	**2231 / 2460**	**90.7%**
Cycling	**1083 / 1152**	**94.0%**
Overall	**4881 / 5808**	**84.0%**

Table 5-11 25Hz Manual Testing Results Summary

Comparison between 100Hz and 25Hz

		100Hz	25Hz
Person 1	**Walking**	75.6%	60.5%
	Running	83.8%	86.0%
	Cycling	95.2%	88.5%
	Overall	84.0%	78.4%
Person 2	**Walking**	61.0%	55.9%

	Running	96.0%	88.6%
	Cycling	100%	98.8%
	Overall	81.1%	75.5%
Person 3	**Walking**	100%	96.4%
	Running	97.1%	96.9%
	Cycling	100%	97.8%
	Overall	98.8%	96.9%
Overall	**Walking**	**78.9%**	**71.4%**
	Running	**92.3%**	**90.7%**
	Cycling	**97.9%**	**94.0%**
	Overall	**88.3%**	**84.0%**

Table 5-12 Manual Testing Results Comparison Between 100Hz and 25Hz

5.2.3 Automatic Testing Bed Results

In this section I will show all the testing results generated by the automatic testing bed. The personal details of the first 3 testing users have been written in last section.

Person 1

	100Hz	**25Hz**
Walking 1	51/120	48/120
Walking 2	112/120	74/120
Walking 3	90/120	89/120
Walking Summary	253/360 (70.28%)	211/360 (58.611%)
Running 1	88/120	91/120
Running 2	66/120	70/120
Running 3	108/120	96/120
Running Summary	262/360 (72.78%)	257/360 (71.38%)
Cycling 1	90/90	83/90
Cycling 2	81/90	81/90
Cycling Summary	171/180 (95.00%)	164/180 (91.11%)
Overall	686/900 (76.22%)	632/900 (70.22%)

Table 5-13 Automatic Testing Results for Person 1

Person 2

	100Hz	25Hz
Walking 1	92/120	102/120
Walking 2	67/120	57/120
Walking 3	70/120	77/120
Walking Summary	229/336 (63.61%)	236/360 (65.56%)
Running 1	120/120	101/120
Running 2	116/120	109/120
Running 3	117/120	115/120
Running Summary	353/360 (98.06%)	325/360 (90.28%)
Cycling 1	90/90	90/90
Cycling 2	90/90	90/90
Cycling Summary	180/180 (100%)	180/180 (100%)
Overall	762/900 (84.67%)	741/900 (82.33%)

Table 5-14 Automatic Testing Results for Person 2

Person 3

	100Hz	25Hz
Walking 1	120/120	108/120
Walking 2	120/120	96/120
Walking 3	120/120	108/120
Walking Summary	360/360 (100%)	312/360 (86.67%)
Running 1	120/120	120/120
Running 2	120/120	117/120
Running 3	119/120	101/120
Running Summary	359/360 (99.17%)	338/360 (93.89%)
Cycling 1	90/90	87/90
Cycling 2	90/90	90/90
Cycling Summary	180/180 (100%)	177/180 (98.33%)
Overall	889/900 (99.89%)	827/900 (91.89%)

Table 5-15 Automatic Testing Results for Person 3

Person 4

	100Hz	25Hz
Walking 1	120/120	116/120
Walking 2	111/120	118/120
Walking 3	109/120	114/120
Walking Summary	340/360 (94.44%)	348/360 (96.67%)

Running 1	120/120	119/120
Running 2	108/120	94/120
Running 3	120/120	110/120
Running Summary	348/360 (96.67%)	323/360 (89.72%)
Cycling 1	84/90	87/90
Cycling 2	90/90	90/90
Cycling Summary	174/180 (96.67%)	177/180 (98.33%)
Overall	862/900 (95.78%)	848/900 (94.22%)

Table 5-16 Automatic Testing Results for Person 4

Person 5

	100Hz	**25Hz**
Walking 1	108/120	110/120
Walking 2	116/120	112/120
Walking 3	117/120	112/120
Walking Summary	341/360 (94.72%)	334/360 (92.78%)
Running 1	120/120	120/120
Running 2	119/120	109/120
Running 3	120/120	114/120
Running Summary	359/360 (99.72%)	343/360 (95.28%)
Cycling 1	88/90	89/90
Cycling 2	85/90	82/90
Cycling Summary	173/180 (96.11%)	171/180 (95.00%)
Overall	873/800 (97.00%)	848/900 (94.22%)

Table 5-17 Automatic Testing Results for Person 5

Person 6

	100Hz	**25Hz**
Walking 1	98/120	91/120
Walking 2	106/120	98/120
Walking 3	110/120	113/120
Walking Summary	314/360 (87.22%)	302/360 (83.89%)
Running 1	117/120	119/120
Running 2	103/120	98/120
Running 3	119/120	107/120
Running Summary	339/360 (94.17%)	324/360 (90.00%)
Cycling 1	90/90	89/90

Cycling 2	90/90	90/90
Cycling Summary	180/180 (100%)	179/180 (99.44%)
Overall	833/900 (92.56%)	805/900 (89.44%)

Table 5-18 Automatic Testing Results for Person 6

Person 7

	100Hz	**25Hz**
Walking 1	120/120	118/120
Walking 2	118/120	109/120
Walking 3	115/120	113/120
Walking Summary	353/360 (98.06%)	340/360 (94.44%)
Running 1	120/120	119/120
Running 2	120/120	120/120
Running 3	120/120	118/120
Running Summary	360/360 (100%)	357/360 (99.17%)
Cycling 1	88/90	84/90
Cycling 2	90/90	98/90
Cycling Summary	178/180 (98.89%)	172/180 (95.56%)
Overall	891/900 (99.00%)	869/900 (96.56%)

Table 5-19 Automatic Testing Results for Person 7

Person 8

	100Hz	**25Hz**
Walking 1	108/120	99/120
Walking 2	87/120	108/120
Walking 3	113/120	104/120
Walking Summary	308/360 (85.56%)	311/360 (86.39%)
Running 1	119/120	120/120
Running 2	109/120	94/120
Running 3	120/120	117/120
Running Summary	348/360 (96.67%)	331/360 (91.94%)
Cycling 1	90/90	90/90
Cycling 2	87/90	87/90
Cycling Summary	177/180 (98.33%)	177/180 (98.33%)
Overall	833/900 (92.56%)	819/900 (91.00%)

Table 5-20 Automatic Testing Results for Person 8

Person 9

	100Hz	25Hz
Walking 1	119/120	118/120
Walking 2	120/120	119/120
Walking 3	120/120	120/120
Walking Summary	359/360 (99.72%)	357/360 (99.17%)
Running 1	120/120	100/120
Running 2	118/120	117/120
Running 3	120/120	98/120
Running Summary	358/360 (99.44%)	315/360 (87.50%)
Cycling 1	90/90	88/90
Cycling 2	90/90	90/90
Cycling Summary	180/180 (100%)	178/180 (98.89%)
Overall	897/900 (99.67%)	850/900 (94.44%)

Table 5-21 Automatic Testing Results for Person 9

Person 10

	100Hz	25Hz
Walking 1	115/120	90/120
Walking 2	120/120	115/120
Walking 3	98/120	111/120
Walking Summary	333/360 (92.50%)	316/360
Running 1	99/120	107/120
Running 2	96/120	84/120
Running 3	102/120	89/120
Running Summary	297/360 (82.50%)	280/360 (77.78%)
Cycling 1	81/90	85/90
Cycling 2	88/90	88/90
Cycling Summary	169/180 (93.89%)	173/180 (96.11%)
Overall	799/900 (88.78%)	769/900 (85.44%)

Table 5-22 Automatic Testing Results for Person 10

Overall Summary

	100Hz	25Hz
Walking	3190/3600 (88.61%)	3067/3600 (85.19%)

Running	3383/3600 (93.97%)	3193/3600 (88.69%)
Cycling	1762/1800 (97.99%)	1748/1800 (97.11%)
Overall	92.61%	88.98%

Table 5-23 Automatic Testing Results Summary

5.2.4 Comments

From the testing results we can see, the accuracy of the activity recognition is high. 92.61% for the desktop processing and 88.98% for the Pocket PC is very good achievement. Because of the using the unsupervised learning, the high accuracy of activity recognition is applied to almost every testing users.

The accuracy on recognition of walking is the lowest. Especially for the first two testing users, the accuracy is about 60% to 70%. The main reason for this phenomenon is because of bad training. When I first time collecting the data from the testing users, I ask them to walk a bit fast. That is because I want the activity detector to be able to detect the fast walking as walking. However, if I am using the fast walking data sets to train the classifier, it will make it believe that his/her walking should be such intensive. That is why we have so many bad recognitions from the first two testing users. The accuracy is increased for the rest of testing users when I slow down a bit on training of walking. The accuracy for both running and cycling is very high.

I have carried out the tests in both 100Hz MT9 sampling rate and 25Hz. The accuracy from 100Hz MT9 sampling rate is higher than 25Hz. However, the advantage of using high sampling rate is not very evident. It is below 5% difference between these two sampling rate. However, the 25Hz sampling rate has reduced 75% of incoming data. The result of 25Hz convict us that the smart activity monitor can be built on PDA with its current processing ability.

One downside to the evaluation of this project is the inability to test the energy expenditure calculation part of the project. To get a precise energy expenditure figure, I need to have many physiology and medical equipments. And then I can

compare the results that calculated by my project to the equipments' results. There is a shortage on both facilities and related knowledge of testing the energy expenditure.

5.3 Performance Evaluation on Pocket PC

In order to evaluate the application performance on iPAQ, a third-party profiler software is used. This software is called SpeedDemon Profiler, which is developed by NoctemWare [16]. This is commercial software, which cost $299. I am using its trail version in this project. The goal of using this profiler software is finding out how much time needed to detect activity. FeatureExtraction() and ActivityRecognition() are the two main parts of activity detector. If we know the time consuming on these two functions and their sub-functions, we'll know the time needed for detecting activity as well as how much delay we have on the monitor.

Function	Max. F time	Min. F time	Avg. F time
FeatureExtraction()	0.04927 ms	0.01747 ms	0.02653 ms
ActivityRecogniton ()	0.00070 ms	0.00002 ms	0.00330 ms

Table 5-24 Profiling Results On Two Main Functions

The average amount of time spends on extracting features and recognizing activities is about 0.03 milliseconds. In the case of refresh rate at every three second, there will be a 0.03 milliseconds delay every 3 seconds accumulatively. Such performance is reasonably good as the amount time delay is very short. The user will not feel the delay even use the monitor for an entire day.

Chapter 6: Conclusion

From the introduction chapter I have shown that overweight is a huge health problem worldwide. People who have willing to control their weight have difficulties in knowing the energy expenditure. Unbalancing between energy consumption and energy expenditure will affect people's weight and health. In this project I've successfully designed and implemented a smart activity monitor to overcome this problem. The smart activity monitor is using a MT9 sensor to collect the motion data from human's lower body. The MT9 data then will be transferred to a HP iPAQ 5500 running a Microsoft Pocket PC 2003 and processed by the software application. Finally the activity type as well as the activity speed and energy expenditure will be displayed to the user in real time. In this project, I have achieved detecting 3 types of leg-only avidities, including walking, running and cycling as well as estimate the speed and calculate the energy expenditure.

There are many human activity detection techniques available. I found the "Inside-In" system [1] which placed one or multiple sensors on the user's body to collect the source data from the user, is the most appropriate approach to this project. Such system is easy to set up, low computation and it can be mobile.

The activity detector I designed using unsupervised learning to built classifier and using the statistical pattern recognition to recognize activities. The reason of using the unsupervised learning here is because I found that different people's movement style is different. It is not possible to build a perfect classifier with predefined threshold which suits everyone. The classifier I designed requires user training in advance. Features of each activity for that user will be learnt during the course of training. This activity detector has achieved very good recognition accuracy. The unsupervised classifier is also easier to learn new activity. I have fully evaluated the algorithm by using a specific designed testing bed. I've collected data sets from 10 volunteers. Each person has 6 different data sets on different activities. I used

technique that combines different data sets as classifier to test the rest of data sets. In this case, I can carry out 900 different tests. I've also made the tests fully automatic, which saves huge amount of time. The overall accuracy on 100Hz MT9 sampling rate is 92.61%. The activity detector part of project is independent from the application. This makes the activity detector adaptable to any context awareness application which requires the user's activity information.

The system has been successfully implemented on a handheld. During the course of deployment, there are many problems have been solved. Because the processing ability of handheld is much lower, I found that it had difficulty in handling the 100Hz MT9 data flow. Therefore I decide to reduce the sampling rate of MT9 to 25Hz. The accuracy of activity detector under 25Hz MT9 data has also been tested by using the automatic testing bed. The overall accuracy is 88.98%. The performance of the monitor on the handheld is as good as its accuracy. I've achieved a real-time monitoring based a fixed refresh rate. There is only 0.03 millisecond delay every time the monitor refreshes.

Base on the activity detector, I've also designed and implemented the application that can estimate the user' speed by using the specific designed algorithm as well as calculate the energy expenditure. The application is also supported by a file system, which is used to store the users' profile. The file system increases the usability of the application as the user does not have to train the classifier every time he/she uses the monitor.

According to the complex evaluation of the project as well as comparing to the original proposal defined at the very beginning, I believe I have completed my project and achieved my goal. However, anything can be perfect in this world. If we consider this project as a beginning of using motion sensor to monitor human activity, there are many work can be carried out in the future.

6.1 Future Work

The future improvement of this project can be both in hardware and software. The current smart activity monitor can only detect the energy expenditure on leg-only activities. However, we spend energy all the time even when we are sleeping. If we want to monitor our energy expenditure 24/7, the smart activity monitor I designed can only reflect the energy expenditure when the users are standing, sitting or sleeping. That is because I am using an indirect method [20] to calculating the energy expenditure. If we want monitor that part of energy expenditure in the future, we can add in a cardiotachometer. We can monitor the energy expenditure according to the heart rate changes. By add-in a cardiotachometer, we can monitor the energy expenditure directly. The accuracy of energy expenditure calculation can also be improved.

Both the sensor and handheld used in this project is still very big in their size. HP iPAQ 5500 is used as the data processor as well as for the annotation in this project. I hope in the future we can use mobile phone to handle this job instead of PDA, because mobile phone is a more pervasive device. The communication between the MT9 and the handheld is wired in this project. This is also a draw back, because it is both uncomfortable and dangerous to have cable tied on user's body when the user is doing activities. If the sensor itself can be smaller, lighter and have a wireless communication interface built-in, it will be more likely be accepted by users. The communication between the sensor and mobile phone can be carried out by some low power wireless communication technique. In the future I hope we can find out a better power supply solution for the sensor, such as kinetic power.

From the software aspects, a more optimizer algorithm can be carried out for further development. For example, the current activity detector is using an unsupervised classifier to classify the activities. However, the weight of each feature is predefined. If I can have more time for this project, would optimize the feature weighting scheme.

Appendix A: Screen Shots

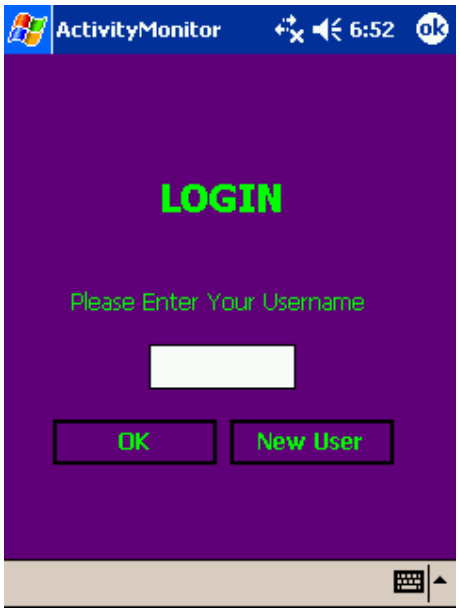


Figure 6-1 Login Screen

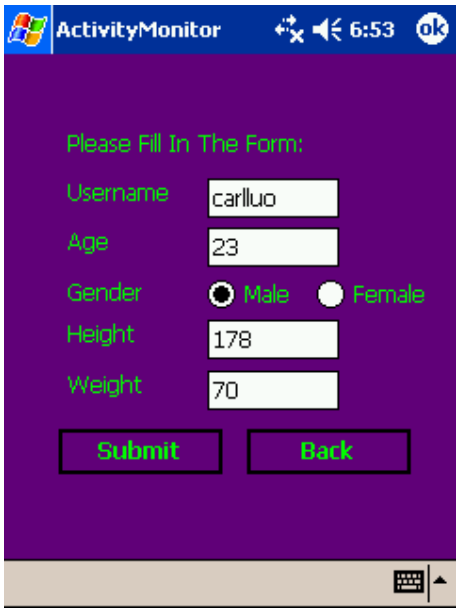


Figure 6-2 Registration Screen

Figure 6-3 Training Screen

Figure 6-4 Training Screen

Figure 6-5 Activity Monitoring Screen

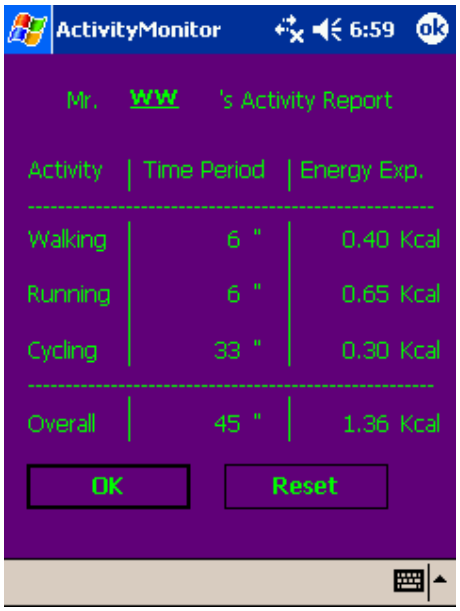


Figure 6-6 Report Screen

Appendix B: References

[1] A. Mulder, "Human movement tracking technology" In Technical Report 94-1, Simon Fraser University, (1994). http://www.xspasm.com/x/sfu/vmi/HMTT.pub.html

[2] H. Zhou, H. Hu, "A Survey - Human Movement Tracking and Stroke Rehabilitation" In Technical Report: CSM-420 ISSN 1744 – 8050, University of Essex, (2004)

[3] G. Johansson, "Visual motion perception" Scientific American, 232:76–88, (1975)

[4] B. Rosenhahn, U.G. Kersting, A.W. Smith, J. K. Gurney, T. Brox and R. Klette, "A System for Marker-Less Human Motion Estimation" The University of Auckland, New Zealand, Springer-Verlag Berlin Heidelberg (2005)

[5] CODA motion, "Codamotion Products Introduction, System Overivew, Hardware, Software, Interfacing" Charnwood Dynamics Ltd, (2006), http://www.charndyn.com/Products/Products_Intro.html

[6] QUALISYS, "Method – Capture, View and analyze" Qualisys Medical AB and Optomatrix Technology AB, (2006), http://www.qualisys.com/method.html

[7] J. Choi, S. Choi, D. Shin and D. Shin, "Real-Time Human Tracker Based Location and Motion Recognition for the Ubiquitous Smart Home" Sejong University, Korea, Springer-Verlag Berlin Heidelberg (2006)

[8] T. Dowad, "PAWS: Personal Action Wireless Sensor" Personal and Ubiquitous Computing, Springer-Verlag London Limited (2006)

[9] R. Aipperspach, E. Cohen and J. Canny, "Modeling Human Behavior from Simple Sensors in the Home", Berkeley institute of Design, University of California, Berkeley, PERVASIVE 2006, Springer-Verlag Berlin Heidelberg (2006)

[10] J. Baek, G. Lee, W Park and B Yun, "Accelerometer Signal Processing for User Activity Detection", Kyungpook National University, Korea, Springer-Verlag Berlin Heidelberg (2004)

[11] N. Kern, B. Schiele and A. Schmidt, "Multi-sensor Activity Context Detection for Wearable Computing", ETH Zurich, Switzerland, Springer-Verlag Berlin Heidelberg (2003)

[12] J. Lester, T. Choudhury and G. Borriello, "A Practical Approach to Recognizing Physical Activities", University of Washington, Seattle, WA 98195, USA, PERVASIVE 2006, Springer-Verlag Berlin Heidelberg (2006)

[13] R. Weber-Collins, S. Tsvetkov and M. Haahr "Smart Sword" CAMS project, Trinity College Dublin (2003) http://www.dsg.cs.tcd.ie/dynamic/?category_id=351

[14] J. McKnight and E. Meehan "Gesture Recognition Pilot project" Trinity College Dublin (2005)

[15] "Sofas get smart for hi-tech homes" Article on BBC News World Edition, 14 September, 2003 http://news.bbc.co.uk/2/hi/technology/3107746.stm

[16] "SpeedDemon Profiler Manual" NoctemWare http://www.noctemware.com/index.html

[17] "Obesity and overweight - Global Strategy on Diet, Physical Activity and Health" World Health Organization, http://www.who.int/dietphysicalactivity/publications/facts/obesity/en/

[18] "Obesity" Health Promotion Unit, Ireland, http://www.healthpromotion.ie/topics/obesity/

[19] J. Hagberg and S. McCole " Energy Expenditure During Cycling" From Chapter 8 of Book "High-Tech Cycling" edited by E. Burke, Human Kinetics (1996)

[20] W. McArdle, F. Katch and V. Katch "Exercise Physiology – Energy Nutrition and Human Performance" Fifth Edition, Section 2 (Chapter 8, 9 and 10), Lippincott Williams & Wilkins (2001)

[21] N. Gurewich & O. Gurewich " Teach Yourself Visual C++ 5 in 21 Days" Fourth Edition, Sams Publishing (1997)

[22] "Pitch, Roll, Yaw Systems" Liftoff to Space Exploration, NASA (1995) http://liftoff.msfc.nasa.gov/academy/rocket_sci/shuttle/attitude/pyr.html

[23] " Dictionary of Computing" 4th Edition, Oxford University Press, Page 226

[24] Xsens Technologies. Motion Tracker Technical Documentation MT9-B and MT6-B http://www.xsens.com, May, 2005

[25] "HP iPAQ Pocket PC h5500 series - Specification" From Product Packet.

[26] Xsens Technologies. MT9 Software Development Kit Documentation http://www.xsens.com, May, 2005

[27] Stuart Konen "C2DPushGraph: A Push Graph Control" http://www.codeproject.com/miscctrl/C2DPushGraph.asp (2005)

[28] H.M. Deitel and P.J. Deitel, "C How to Program" Fourth Edition, Prentice-Hall, Inc. (2004)

[29] T. Grandon Gill, "Introduction to Programming Using Visual C++ .NET", Unviersity of South Florida, Published by John Wiley & Sons (2005)

[30] W. Savitch, " Problem Solving with C++, The Object of Programming", Fifth Edition, Pearson (2005)

[31] T. Archer and A. Whitechapel, "Visual C++ .NET Bible", Wiley Publishing, (2002)

[32] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, "Nunerical Recipes in C++, The Art of Scientific Computing", Second Edition, Cambridge University Press (2002)

[33] Andrew R. Webb, " Statistical Pattern Recognition", Second Edition, Wiley Publishing (2002)