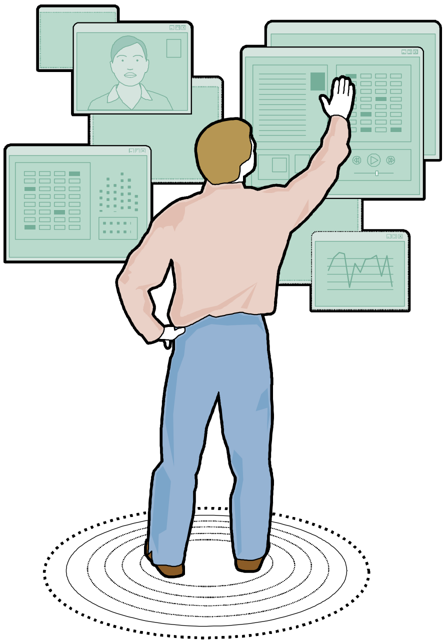
University of Melbourne
Department of Computer Science and Software Engineering

433659 MEDC PROJECT REPORT

3D Accelerometer Based Gesture Control Human Computer Interface



Project Supervisor: Dr. Lars Kulik

Student	Peng Deng	263497
	Qifeng Mao	272746

June 11, 2008

TABLE OF CONTENT

Authors:
Peng Deng MEDC Student dengpeng.cn@gmail.com
Qifeng Mao MEDC Student maoqifeng@gmail.com

ABSTRACT
We developed a gesture based human computer interaction interface. Wireless sensor node is used to capture human body acceleration data. To segment received acceleration data stream, we developed an algorithm based on sliding window and standard deviation. To recognize gesture, Hidden Markov Model (HMM) which is a machine learning algorithm is used. Series prototype applications are built to demonstrate possible gesture based applications in future. We conducted several experiments as well. Finally, we got highest 96% accuracy and lowest 17% accuracy.

KEYWORDS: body sensor network, wireless sensor network, data stream processing, hidden markov model, machine learning, gesture recognition, human-computer interface

1. Introduction

The world we live in has become suffused with computer technologies. Computers are now embedded within a huge range of materials and artefacts, and take on roles in almost all aspects of life. These changes will be continued, it is not only on our desktops and in our hands, but also in virtually all aspects of our lives, in our communities, and in the wider society of which we are a part. Some changes are spurred on by technology, and others are by the reason of technological innovation[1].

Most of us learned how to use a computer by interacting with a personal computer, using a keyboard and mouse to point, click and select, then the computer will help us to do the tasks. The GUI has dominated the way we interact with computers for over twenty years, and for most of the case it is quite forgiving. But on the other hand, researchers have pointed out that pointing, clicking and dragging are not ideal forms of interaction for many tasks. For example, try to drawing a flower or signing your name using a mouse[1].

In the last few years, new input techniques have been developed that are richer and less prone to the many shortcomings of keyboard and mouse interaction. For example, there are tablet computers that use stylus-based interaction on a screen[2], and even paper-based systems that digitally capture markings made on specialized paper using a camera embedded in a pen[3]. These developments support interaction through sketching and handwriting.

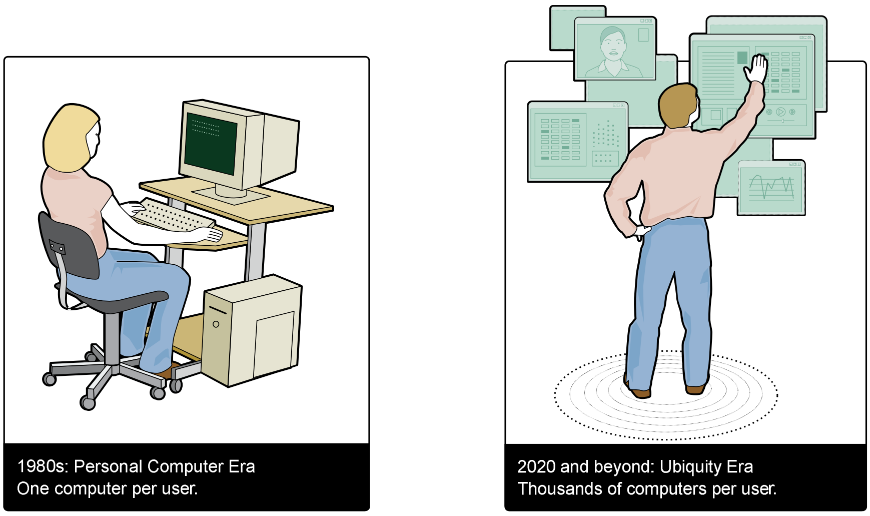


FIGURE 1.1 The evolution of human computer interaction[1]

From GUIs to multi-touch[4], speech to gesturing, the ways we interact with computers are diversifying as never before. Both researchers and industry have agreed that nature interface will dramatically change how we interact with computer in near future. Some products with natural interface have already reached market, for example Nintendo Wii game console.

We are now standing at the edge of booming of natural interface. In this project we try to free user from keyboard and mouse, and bring a more natural free space gesture human computer interaction interface to users with the help of sensor technology.

2. Related Works

In future, “off the desk” interaction can be contact based like multi-touch screen[4] or free-space based systems. This chapter discusses two main solutions to collect human motion from free space, the vision-based solution and the sensor-based solution. Comparison is made to decide which solution we should choose in our project.

2.1 Vision based solutions

Vision-based solutions collect human motion from one or multiple cameras. Vision-based devices can handle properties such as texture and color for analyzing a gesture, while sensor cannot. A tracker also needs to handle changing shapes and sizes of the gesture-generating object (that varies between individuals), other moving objects in the background, and noise[5].

Vision-based techniques can vary among themselves in[5]: 1. the number of cameras used; 2. their speed and latency; 3. the structure of environment (restrictions such as lighting or speed of movement); 4. any user requirements (whether user must wear anything special); 5. the low-level features used (edges, regions, silhouettes, moments, histograms); 6. whether 2-D or 3-D representation is used; and 7. whether time is represented.

Researchers from Georgia Institute of Technology use GT2k which is a gesture recognition toolkit to create an American Sign Language game for deaf children[6]. A pair of pink glove is used. The students pushed a button and then signed a gesture, and the data was collected using an IEEE 1394 video camera. The hand is pulled from the video image by its bright color. User-dependence model was validated in a 90/10 (training/testing) split, with word accuracy in the low 90s. User-independent models were lower with an average word accuracy of 86.6%.

Researchers from National Taiwan University of Science and Technology use VICON system which is a professional vision based motion tracking system to extract features[7]. The features are then sent into a back-propogation neural network (BPNN). The best result accuracy was 94.65%.

2.2 Sensor based solutions

Sensor-based solution typically requires the user to wear a cumbersome device nowadays. This hinders the ease and naturalness of the user’s interaction with the computer. According to Moor’s law, sensors are getting smaller and cheaper as time goes on. It will be a pervasive in future, we believe.

Researchers from UC Berkeley developed a wearable motion sensor network to recognize human activities[8]. They use both accelerometer and gyroscope on Tmote Sky platform.

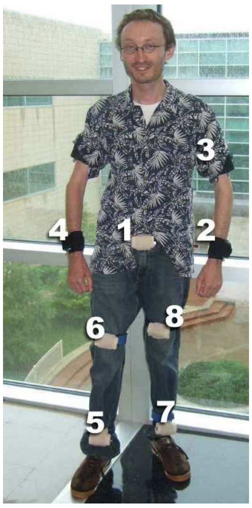


FIGURE 2.1. Wearable sensors from UC Berkeley

They introduced distributed pattern recognition. Each mote (mote is a sensor network terminology means node interchangeably) can do classification locally and sends events to global classifier to recognize motion. Compare to centralized approach, it dramatically reduced the communication and energy consumption. It increases the robustness of system, user can active necessary node when needed on the fly.

For segmentation and recognition, they implemented linear discriminate analysis as global classifier. On sensor side, basis pursuit-based algorithm is implemented. In experiment, they got 98.8% accuracy when all nodes activated and 89.8% accuracy when only one sensor is activated.

Researchers from Finland developed an accelerometer based gesture interaction to control DVD player[9]. They developed their own measure instrument. HMM is used to recognize gesture. They manually do the segmentation on data stream. The interesting point is a procedure based on adding noise-distorted signal duplicates to training set is applied and it is shown to increase the recognition accuracy while decreasing user effort in training. With the help of noisy data, user can have only 2 training samples for each gesture.

Researchers from Waswda University Japan use wired accelerometers and gyroscopes to extract features of gestures and create a music tempo system[10]. Changes of acceleration are measured using a fuzzy partition of radial angles. They use threshold level of acceleration to identify start and end of gesture. The authors recognize or classify gestures using squared error.

2.3 Comparison and analysis

	Vision based solutions	**Sensor based solutions**
Cost	☺	😐
User Comfort	☺	😐
Calibration	☺	😐
Computing Power	☹	😐
Portability	☹	☺
Data loose	☹	☺
Ubiquity	😐	☺

TABLE 2.1 Comparison between vision based and sensor based solutions

Cost: The cost of sensor based system is relatively high. On the other hand, a vision-based solution is relatively inexpensive, especially since modern-day PCs are equipped with cameras. As time goes on, there will be little difference between them.

User Comfort: Compare with sensor based solution, vision based system does not require user to wear any device. However, from another point of view, sensors are getting smaller and smaller. They will be small enough to embed into cloth which will not be noticed by user in future.

Computing Power: Depending on the algorithms used, both sensor-based and vision based solutions can require significant computing power. However, in general, the vision-based approach takes more computing power due to the image processing necessary. Sensor-based solutions have a slight advantage over vision-based solutions in that the data the sensors send to the computer can easily be transformed into records that are suitable for recognition.

Calibration: In general, a calibration procedure or step is required in sensor based solution for every user and, in some cases, every time a user wants to run the system. In some vision-based solutions, however, a general calibration step can be used for a wide variety of users.

Portability: Vision-based solutions are quite difficult to use in a mobile environment due to camera placement issues and computing power requirements.

Data lose: Vision-based system may have line of sight problem. Parts of the user's body may occlude each other from camera. Although, multiple cameras can be used, vision-based system inherently loses some critical data when mapping 3D data to 2D data.

Ubiquity: Sensor is becoming cheaper and pervasive. We believe every mobile phone in future will have variety types of embedded sensors which can be used to interact with both user and environment nearby. Not only input, but also feed back can be supported on sensor, for example force feed back.

From the comparison and analysis above, we decide to implement a sensor based human motion recognition system.

3. Theory and Technical Details

Gesture recognition is an interdisciplinary research, since developer should consider issues from both physical world and computer system. To recognize human motion, first we need to capture it and transmit motion data to computer. This raw data is then analyzed by various recognition algorithms to extract meaning or context from the data in order to perform tasks in application. In this chapter, we are going to discuss the choice of hardware platform and recognition algorithms.

3.1 Capture human motion data

To recognize human gestures and motions, we need to gather raw data from human body. As we discussed in section 2.2, there are several choices. In following sections, we will discuss body sensor network platform and why we choose Sun SPOT as our development platform in detail.

3.1.1 Body sensor network platform

Originally, wireless sensor network was designed for military applications such as battlefield surveillance. However, wireless sensor networks are now used in many civilian application areas, including environment and habitat monitoring, home automation, and traffic control[13]. Recently, use wireless sensor nodes in health and human motion monitoring become a popular topic.

The idea of Body Sensor Network(BSN) is first introduced by researchers from Imperial College London[12]. According to [11], WSN node provides a suitable development platform for pervasive health care applications. Various physiological sensors can be integrated into WSN node. It revolutionizes the health care system by allowing inexpensive, continuous and ambulatory health monitoring with real-time updates of medical records via Internet. A number of intelligent physiological sensors can be integrated into a wearable wireless body area network, which can be used for computer assisted rehabilitation and even early detection of medical conditions.

Finally, we decide to use wireless sensor node in our project rather than build our own specific hardware, because wireless sensor node provides an extensible and easy to use platform which guarantees possible applications in future.

3.1.2 Sun SPOT and advantages over other platforms

There are plenty types of wireless sensor nodes we can choose, from legacy motes like mica2[15], micaZ[16], Tmote Sky[17] to latest motes, like Sun SPOT[18] and Imote2 Build[19].

Sun SPOT (Sun Small Programmable Object Technology) is one of the latest sensor nodes developed by Sun Microsystems. Developer can write Java application to deploy to this small device. There are numbers of sensors already integrated on board.

FIGURE 3.1. Sun SPOT[18]

Imote2 Builder is a joint product developed by Crossbow and Microsoft. .Net Micro Framework is embedded in Imote2, similar to Squawk Virtual Machine in Sun SPOT. .Net Micro Framework supports C# programming language.

	Legacy Motes	**Sun SPOT**	**Imote2 Builder**
Hardware capability	☹	☺	☺
Extendibility	☹	☺	☺
Programmability	☹	☺	☺
Open source	😐	☺	☹
Scalability	☺	😐	😐
Battery life	😐	☹	☹
Affecting	😐	☺	☺
Security	☹	☺	☺
Technical support	😐	☺	😐
Affordability	☺	😐	☹

TABLE 3.1. Comparison between motes

Hardware capability: Compare with legacy motes, new generation nodes have more powerful processor and larger memory space. More computing tasks can be directly processed locally instead of sending message back to central server. Powerful hardware platform ensures extendibility of future application. Both Sun SPOT and Imote2 have accelerometer sensor which we need to use in this project.

Extendibility: Not only more powerful hardware, but also stackable architecture is used in new generation mote. Users can buy or build their own sensor board to attaché to without re-design or modify existing platform. Both Sun SPOT and Imote2 support analog and digital I/O. Even actuators like servos, speaker and force feedback component can be integrated easily.

Programmability: As we know, develop embedded application is a tricky work. First, developers have to configure multiple standalone toolkits like IC burner and compiler to make them work together. In contrast, new systems provide a total solution. Now developers can work in their favorite IDE like Netbeans and Visual Studio without worrying about configuration. Second, developers have to learn special architecture of tinyOS and nesC. Developers have to know a lot of specific low level hardware knowledge, like how to manage memory space and how to make mote sleep. On the other hand, with the help of virtual machine like Squawk and .Net Micro Framework, developers are separated from hardware level. They can create embedded applications using the technology they already familiar with, for example Java and C#. Thanks to the powerful hardware, richer functionality like multi-thread, multi-task and garbage collector are supported in emerging embedded system.

Open source: Sun SPOT is an open source platform. Not only software, but also hardware is open source. Developers can freely customize it to meet any special requirements. Compare with Sun SPOT, both Imote2 and .Net Micro Framework are close source. Sun SPOT is supported by open community. For Imote2 Builder, the support is relative close and hard to use.

Security: Both Sun SPOT and Imote2 Builder can apply RSA cryptography algorithm which is more secure.

Compare with mature legacy system, still, new generation motes have many other problems like the battery life, scalability and cost. As time goes on, these differences will get smaller and smaller. In this case, we choose to develop our application based on Sun SPOT instead of legacy system and Imote2 Builder.

3.2 Recognize human motion

Now, we have hardware platform to capture human motion. Acceleration data can be transmitted wirelessly to computer and waiting for further processing. The characteristics and challenges of motion recognition will be presented in this section. The theories of solutions to these challenges are given in second and third sections.

3.2.1 Characteristics and challenges of human motion

Human motion is an inherent continuous event and difficult to predict. Basically the system we want to develop is a real time data stream processing system. Compare data stream processing system with traditional data analyzing system, stream passed through the system once only, and both processing time and memory usage is restricted.

Human motion recognition problem is similar to voice recognition. The main difference between these two is motion occurred in three dimensional which requires at least 3 attributes to represent.

Making best use and reuse of a motion data stream requires recognizing the motions in the stream with high accuracy. Recognizing motions in such continuous multi-attribute streams also involves segmentation that is, identifying the beginning and end of a motion in a stream. To recognize and segment motion streams with high accuracy, several challenges need to be addressed[20]:

- ✧ **Similar motions can have variations:** Different attributes can have different variations and can have various temporal shifts due to motion variations.

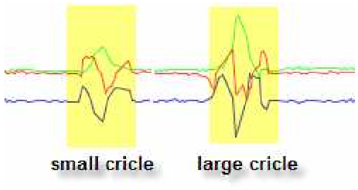


FIGURE 3.2. Variations of similar motions

- **Similar motions can have different lengths:** Motions performed at different times or by different subjects have different durations, and motion sampling rates can also be different at various times. In contrast to subsequence matching, motions in a stream can have longer lengths than similar reference motions.

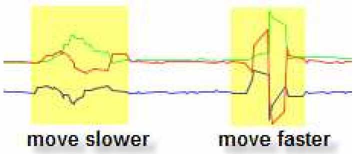


FIGURE 3.3. Similar motions with different length

- **Complete motions need to be compared with both incomplete and over completed motion segments:** As shown in Figure 3.4, complete motions are concatenated by brief transitions, and the motion candidates in a stream can contain these transitions. Hence, the differences between complete motions and motion candidates with missing or extra segments should be captured.

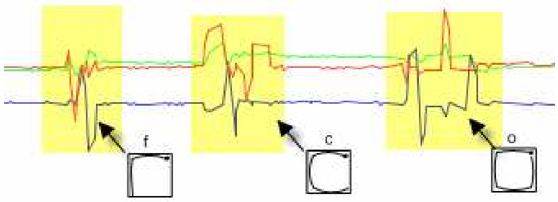


FIGURE 3.4. Three similar motions with different meanings

- **Motions can follow similar trajectories in different directions, and their semantic meanings may differ:** For example, to sit down from a standing pose can follow a similar trajectory as that of standing up from this pose, yet the two motions are different.

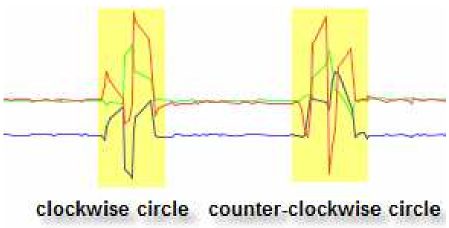


FIGURE 3.5. Similar trajectories in different directions

3.2.2 Recognition algorithms

Obviously, we need to identify different motions user is performing. If the data sequence is pre-segmented, classification becomes straightforward with many classical algorithms to choose from. There have been varied approaches to handle gesture recognition, ranging from mathematical models based on hidden Markov chains to tools or approaches based on soft computing.

It is believed that artificial intelligence techniques can be applied to recognize human motions. Here we discuss three most popular algorithms in this field[5]:

	Accuracy	**Training**	**Previous Work**	**Adv. Knowledge**
NN	96%	Extensive	Extensive	Yes
IBL	N/A	Extensive	Minimal	No
HMM	96%	Extensive	Extensive	Yes

TABLE 3.2. Comparison between three popular recognition algorithms

Neural Networks

A neural network is an information processing system loosely based on the operation of neurons in the brain. Neural networks have been used principally in the artificial intelligence community to build certain types of autonomous agents and recognize patterns.

Neural networks are a useful method for recognizing hand gestures, yield increased accuracy conditioned upon network training, and work for both sensor based and vision-based solutions. However, they have distinct disadvantages. First, different configurations of a given network can give very different results, and it is difficult to determine which configuration is best without implementing them. Another disadvantage is the considerable time involved in training the network. Finally, the whole network must be retrained in order to incorporate a new gesture. If the gesture set is known beforehand this is not an issue, but if postures and gestures are likely to change dynamically as the system develops, a neural network is probably not appropriate.

Strengths

1. Can be used in either a vision- or sensor-based solution
2. Can recognize large posture or gesture sets
3. With adequate training, high accuracy can be achieved

Weaknesses

1. Network training can be very time consuming and does not guarantee good results
2. Requires retraining of the entire network if hand postures or gestures are added or removed

Instance-based Learning

Instance based learning is another type of machine learning algorithm. The most famous Instance based learning algorithm is K-Nearest-Neighbor. Basically, training data is mapped to a Euclidean space. Test sample need to map to space as well to calculate the distance between test sample and training samples. Majority vote can be applied to classify test sample to improve accuracy.

Instance-based learning techniques have the advantage of simplicity, but they have a number of disadvantages as well. One major disadvantage is the cost of classifying new instances. All of the computation must be done whenever a new instance is classified, which means there will be response time issues when dealing with a large amount of training examples. Another disadvantage of these methods is that not all of the training examples may fit in main memory, and thus will also increase response time.

Unfortunately, very little work has been done on instance-based learning in recognizing hand postures and gestures. More research is needed to determine whether the technique can be applied to hand gestures and if the accuracy can be improved.

Strengths

1. Except for case-based reasoning, instance-based learning techniques
2. are relatively simple to implement
3. Can recognize a large set of hand postures with moderately high accuracy
4. Provides continuous training

Weaknesses

1. Requires a large amount of primary memory as training set increases
2. Response time issues may arise due to a large amount of computation
3. at instance classification time
4. Only a little reported in the literature on using instance-based learning
5. with hand postures and gestures

Hidden Markov Models

HMM is defined as a set of states of which one state is the initial state, a set of output symbols, and a set of state transitions. Each state transition is represented by the state from which the transition starts, the state to which transition moves, the output symbol generated, and the probability that the transition is taken.

In the context of hand gesture recognition, each state could represent a set of possible hand positions. The state transitions represent the probability that a certain hand position transitions into another; the corresponding output symbol represents a specific posture and a sequence of output symbols represent a hand gesture. One then uses a group of HMMs, one for each gesture, and runs a sequence of input data through each HMM. The input data, derived from pixels in a vision-based solution or from sensor values, can be represented in many different ways, the most common by feature vectors.

Like neural networks, HMMs must be trained and the correct number of states for each gesture must be determined to maximize performance. If the number and types of hand posture and gestures are known beforehand, HMMs are a good choice for recognition. If the hand postures and gestures are determined as the system is developed, the development process can be more time-consuming due to retraining. Although HMMs require extensive training, and their hidden nature makes it difficult to understand what is occurring within them, they still may be the technique of choice since they are well covered in the literature and the accuracies reported are usually above 90 percent.

Strengths

1. Can be used in either a vision- or sensor-based solution
2. Can recognize large posture or gesture sets
3. With adequate training, high accuracy can be achieved
4. Well discussed in the literature

Weaknesses

1. Training can be time consuming and does not guarantee good results
2. As with multi-level neural networks, the hidden nature of HMMs makes it difficult to observe their internal behavior

In this project, we will use HMM, because it is a proven technique with high accuracy. Second reason is there are lots of mature toolkits that we can directly utilize. However, human gesture is a continuously event. Identify gesture needs both recognition and segmentation as we discussed in 3.2.1.

3.2.3 Segmentation algorithms

A gesture may be affected by the context of preceding as well as following gestures. The recognition of natural continuous gestures requires their temporal segmentation. One technique is user manually specify the start and end points of a gesture in terms of the frames of movement, both in time and in space. This can be done by switches or voice command.

Another technique is auto segmentation which is more nature but much more difficult than manual method. Based on literature review, there have been two major approaches in the past to provide partial solutions to simultaneous segmentation and recognition of human actions on wearable sensors[8]:

1. We can assume different actions are separated by a "rest" state, and such states can be detected by energy threshold level or a special classifier to distinguish between rest and non-rest. However the validity of the rest state between actions is not physically guaranteed. For example, non-transient actions such as walking and running may last for a long period.
2. We can assume all sensors in the network are available at all time, and rejects invalid samples based on the sample distance between the test and training examples. The drawback is not robust when the number of active sensors changes over time. In this case, tuning a list of different distance thresholds to reject outliers when the number of sensors changes can be difficult, which still highly depends on the condition on the training samples? Dynamic Time Warping algorithm and neural networks belong to this category.

In this project, the algorithm we implemented belongs to first category. Since we use human hand gesture to interact with computer, we assume when user done a gesture, user will take a short time waiting for response from computer, and then user can proceed another gesture command. Noisy gestures should be filtered out and not to be considered. Our algorithm is based on acceleration threshold enhanced with standard deviation and sliding window technique. We will explain this in next chapter in detail.

4. Implementation Details

In this chapter, we will discuss the general architecture of the system and each key module in detail.

4.1 System architecture

The architecture of system is pretty straightforward. Sun SPOT is attached to human hand. It continuously samples and sends sampling acceleration data to base station wirelessly. Base station takes responsibility to receive data stream from SPOT. Raw data stream from base station is processed on host PC. Recognition module takes the responsibility to identify human motions. Gesture-based applications then get notified when motion is recognized.

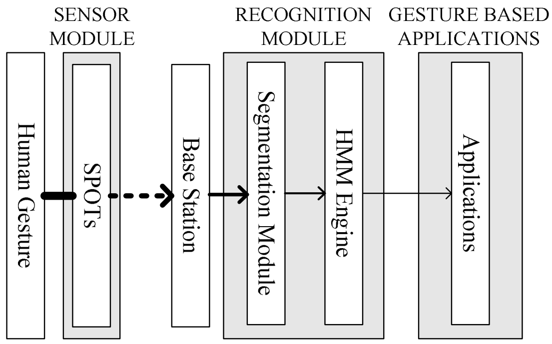


FIGURE 4.1. System architecture

4.2 Sensor module

The application logical inside Sun SPOT is relatively simple. Since we need to track human motion data continuously, Sun SPOT iteratively measures acceleration data from accelerometer and sends data back to base station through radio communication.

The accelerometer chip used on Sun SPOT is ST Microsystems LIS3L02AQ which supports 2 detection models: One has reading range within ± 2G (G is gravity acceleration $9.8m/s^2$) with higher resolution ($600mv/g$), the other one has reading range within ± 6G with lower resolution. On sensor side, we choose ± 6G detection range, since according to research the maximum value may exceeds 20G even in ordinary human motion[10].

Obviously, acceleration data from 3 dimensions are important. Another key feature is tilt on 3 axes. These 6 parameters will be analyzed by machine learning engine. Switches on SPOT are used as well providing manually segmentation mechanism. Switch status will not provide to HMM engine. In total, 9 parameters will be packed into one radiogram packet and transmit through 802.15.4 wireless protocol.

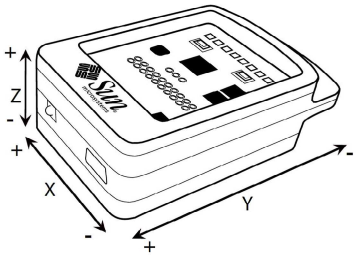


FIGURE 4.2. Accelerometer X, Y and Z axes[18]

In Sun SPOT, it supports two types of wireless transmitting protocols: radiostream and radiogram. Radiostream is very similar to TCP which guarantees reliability. Radiogram is close to UDP which can be used to broadcast packet without reliability. We choose radiogram instead of radiostream in this project is because wireless transmission is not very reliable in nature compare with weird network. The occasionally dropped packet is not a problem in data stream application.

The sampling rate is around 40Hz. For real time application, it is a bit low. It is possible to increase the sampling rate. However, we do not want to increase it too much, since wireless communication may drain battery very quickly and block other wireless devices nearby.

The radiogram packet now has 9 features:

$$Packet_{radiogram} = adr_{SPOT} + a_x + a_y + a_z + t_x + t_y + t_z + sw_1 + sw_2$$

- adr_{SPOT} : The unique address of SPOT which can be used to identify.
- $a_x + a_y + a_z$: Acceleration of 3 axes, integer value.
- $t_x + t_y + t_z$: Degree of tilt of 3 axes, integer value.
- $sw_1 + sw_2$: Represent the status (pressed or released) of switches on SPOT. User can explicitly segment data stream by press and release switch. In this project, only sw_1 is used. For auto segmentation method, the two arguments not necessary. We will explain it in next section.

4.3 Recognition module

This module consists of two main sub modules: segmentation and HMM engine module.

4.3.1 Segmentation module

The system supports both manual and auto segmentation mechanisms on data stream.

For manual segmentation, user has to explicitly mark the start and end point of each gesture. User can segment data stream through buttons on GUI and switches on SPOT. Note that for training session, only manual segmentation is supported to ensure high quality training samples.

When switch is pressed, latter packets in data stream will be recorded to another array until the switch is released. At the moment of switch releasing, the segmentation module will automatically submit the recorded array to HMM engine module.

Example

We have a raw acceleration data series which is shown in Figure 4.3. When user press switch on SPOT, data elements in stream will be recorded until the switch is released. In this example, data “3, 1, 2” will be copied to another array and submitted to HMM engine module.

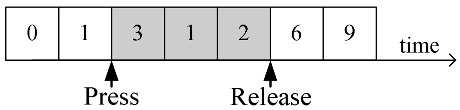


FIGURE 4.3. Segment data stream manually

For auto segmentation, it is a bit complex. 4 key algorithms are used:

- ✧ **Data aggregation:** Combine multi attribute data stream to one. In this project, absolute acceleration values from 3 axes are sum up to create this aggregated data stream.
- ✧ **Sliding window:** It is a FIFO data structure and mainly used to process data stream. Aggregated data stream keeps flowing through sliding window. Figure 4.4 shows an example of sliding window. We only focus on data within window which consists of a sequence of recent data. In this project, we set the window size to 5.

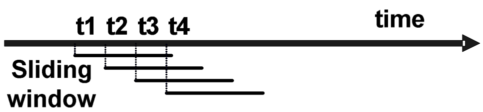


FIGURE 4.4. Sliding window

- ✧ **Standard deviation:** It is a measure of statistical dispersion which indicates how stable the data stream in sliding window is. It is defined as the square root of the variance. In our assumption, perform gesture will lead to unstable value.
- ✧ **Two threshold levels:** If standard deviation value exceeds first threshold level, data will be recorded to array. Later if value below second threshold level, recorded array will be submitted to HMM engine. We set the first threshold value greater than second value. The reasons are: 1. higher first threshold level can help to filter out noisy data; 2. lower second threshold can ensure valuable motion data

are indeed collected as much as possible. In our implementation, 1st threshold =25.5, 2nd threshold =10.5.

Example

We have a raw acceleration data series below. We need to find the start point and end point of gesture based on threshold levels. Acceleration data series between start point and end point will be copied to another data series and submit to recognizer later. Obviously, packets (2,-1,2) and (4,-2,3) are gesture that user performed.

Raw acceleration data series, each packet contains acceleration readings from 3axes (a_x, a_y, a_z):

(0,0,0), (0,-0.1,0), (0,-0.1,0.1), (2,-1,2), (4,-2,3), (-0.1,0.2,0.1), (0,0.1,0),(0,0,0), (0,0,0) time →

Two threshold levels are: START_THRESHOLD=1, END_THRESHOLD =0.2

For simplicity, we set the window size to 3 in this example. The standard deviation value σ_i can be computed only if there are 3 elements within window.

1. Sum up first 3 packets with absolute values and fill them into window, so within window we have: $\boxed{0, 0.1, 0.2}$

2. Compute $\sigma_1 \approx 0.082$,

3. Sum the forth packet, we get 5. Remove the oldest element in window and add the fourth packet to window. So the window now contains: $\boxed{0.1, 0.2, 5}$

4. Compute $\sigma_2 \approx 2.287$,

$\because \sigma_2 \geq START\_\mathrm{THRESHOLD} \quad \therefore$ (2,-1,2) is copied to another data series

5. Sum the fifth packet, we get 9. Remove the oldest element in window and add the fifth packet to window. So the window now contains: $\boxed{0.2, 5, 9}$

6. Compute $\sigma_3 \approx 3.640$,

$\because \sigma_3 \geq START\_\mathrm{THRESHOLD} \quad \therefore$ (4,-2,3) is copied to another data series

7. Sum the sixth packet, we get 0.5. Remove the oldest element in window and add the sixth packet to window. So the window now contains: $\boxed{5, 9, 0.5}$

8. Compute $\sigma_4 \approx 4.418$,

$\because \sigma_4 \geq START\_\mathrm{THRESHOLD} \quad \therefore$ (-0.1,0.2,0.1) is copied to another data series

9. Sum the seventh packet, we get 0.1. Remove the oldest element in window and add the seventh packet to window. So the window now contains: $\boxed{9, 0.5,0.1}$

10. Compute $\sigma_5 \approx 2.973$,

$\because \sigma_5 \geq START\_\text{THRESHOLD} \quad \therefore$ (0,0.1,0) is copied to another data series

11. Sum the eighth packet, we get 0. Remove the oldest element in window and add the eighth packet to window. So the window now contains: $\boxed{0.5,0.1,0}$

12. Compute $\sigma_6 \approx 0.191$,

$\because \sigma_6 \leq END\_\text{THRESHOLD} \quad \therefore$ Now we have (2,-1,2) (4,-2,3) (-0.1,0.2,0.1) (0,0.1,0) in array

13. Sum the ninth packet, we get 0. Remove the oldest element in window and add the ninth packet to window. So the window now contains: $\boxed{0.1,0,0}$

14. Compute $\sigma_7 \approx 0.058$. Rest period between two gestures in our assumption.

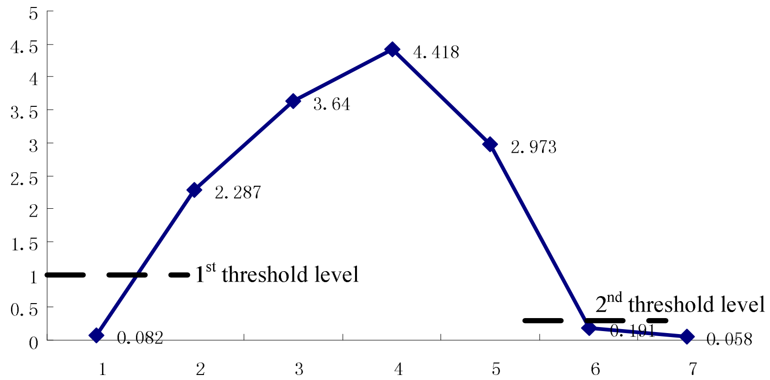


FIGURE 4.5. Standard deviations vs. Time

This is a simple example to explain algorithm we implemented. Finally, we have array (2,-1,2) (4,-2,3) (-0.1,0.2,0.1) (0,0.1,0). As you can see (-0.1,0.2,0.1) (0,0.1,0) is relative stable and should not be considered as motion data, but in reality, motion sequence should last longer than this example. So, small amount of noisy data will not affect accuracy in theory. This array will be submitted to HMM engine module.

4.3.2 HMM engine module

This module takes the responsibility to learn and recognize data series submitted from segmentation module. Since HMM is a supervised algorithm, this module supports both training and recognition. Two open source components are used to develop this module.

- ✧ **GART:** Gesture and Activity Recognition Toolkit (GART)[21] is an on going project developed by Contextual Computing Group in Georgia Institute of Technology. It is a toolkit to allow for rapid prototyping of gesture-based applications. Basically, it is a wrapper of CU-HTK.
- ✧ **CU-HTK:** The Hidden Markov Model Toolkit (HTK)[22] is a portable toolkit for building and manipulating hidden markov models. It is developed by the Machine Intelligence Laboratory in Cambridge University and Microsoft. HTK is primarily used for speech recognition research.

See Figure 4.6, in training session, segmented sample data will be stored to library on disk. Basically, the library is a set of XML files which hold all gesture samples, their names and configuration arguments. When all desired gestures are collected and stored to library, CU-HTK can be triggered to analysis and learn these samples. Learned experience will be stored as a set of files on disk as well.

For each gesture user should provided at least 5 samples. Each sample should last 0.5 second at least. In theory the more samples user provides the higher accuracy can be achieved.

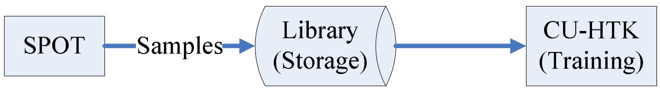


FIGURE 4.6. Training work flow

After training session, user can use HMM engine to recognize gestures. See Figure 4.7, the SPOT sends segmented sample on the fly to CU-HTK which in turn sends a result to all of its listeners (applications).

FIGURE 4.7. Recognition work flow

As we discussed in 3.3.3, HMM has its advantages and disadvantages. As a recognition engine advantages are:

1. There is no limitation of number of gestures;
2. Users can customize their own gestures. Gestures are highly user oriented.

As a gesture based interface disadvantages are:

1. Users have to waste lots of time to train their own gestures;
2. The trained system is user specific, the recognition rate may fall dramatically when other user try to use this system.

To minimize these drawbacks, we predefined some gestures in the system, so user can directly use this system out of box. Gesture based applications that we built will be discussed in the following section.

4.4 Gesture based applications

4.4.1 Gesture based SVG virtual phone book

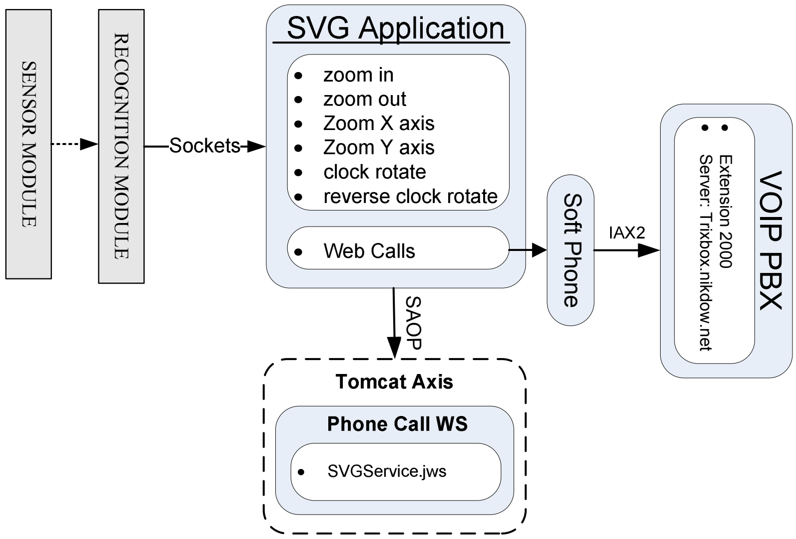


FIGURE 4.8. Virtual phone book architecture

Gesture based SVG Browser

SVG is a standard format used for describing vector graphics in XML. The reasons for choosing SVG are: First SVG as an output platform is accessibility. Users can, for example, zoom into or out of a display easily than for a more conventional view. Another advantage for using SVG is that relationship between SVG and Cascading Style Sheets (CSS) are quite close, which means application can use W3C's recommendation way for styling.

Batik[27] is a Java-based toolkit for applications or applets that want to use images in the Scalable Vector Graphics (SVG) format. It provides various functions that use the SVG format for display. The package is called SVGraphics2D.

Batik's API gives developers a set of core modules that can be used together or individually to handle custom SVG elements or support specific SVG solutions. In our implementation, we use Batik to achieve gesture based functionalities, such as, zoom in, zoom out, zoom X axis, zoom Y axis, clock rotate, and reverse clock rotate.

In Virtual Phone Book application, there are totally six gestures defined, we trained these gestures and stored them into a directory called *PhoneApp*, then inside application logic, each pre-trained gesture is linked to an application function. For example, if you clockwise move Sun SPOT to draw a quarter circle means clockwise rotate the phone book image, if you push Sun SPOT in front of you means call the person relate to the current phone image (detail of VOIP call will be discussed in next section), if you move Sun SPOT to draw a circle means change to next image etc. Once gesture and application function mapping is done, when user wave Sun SPOT, the data pattern generated for particular gesture will pass to recognition engine which compare the receiving data to training date, if the recognition engine find a match pattern, it will pass the *Pattern String* to Virtual Phone Book application use socket communication, then corresponding function defined in application will be executed.

Asterisk based VOIP call

Asterisk[29] is the world's leading open source PBX, telephony engine, and telephony applications toolkit. The reason we choose Asterisk is because it is open source software and it offering flexibility unheard of in the world of proprietary communications.

The next step is choosing SIP extension or AIX extension. SIP (Session Initiation Protocol) is one such protocol which has been the subject of extensive research over the past few years. More recently, IAX (InterAsterisk Exchange Protocol) has emerged as a new VoIP protocol which is steadily gaining credence among the open source community. After careful research, we finally choose IAX as communication protocol by the reason of IAX are simplicity, NAT-friendliness, efficiency and robustness.

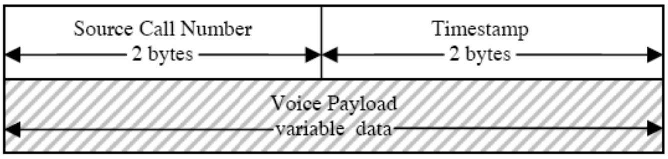


FIGURE 4.9. IAX mini-frame packet

After we install asterisk, then create an extension 2000 for test use.

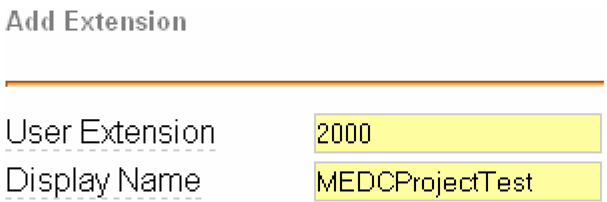


FIGURE 4.10. Create an extension in FreePBX

Next step is creating a trunk for outgoing route, we choose Faktortel as VSP provider, details of IAX2 trunk setting is shown in Figure 4.10.

```
allow=ilbc
auth=md5
context=from-trunk
disallow=all
host=iax.faktortel.com.au
qualify=3000
```

FIGURE 4.11. Faktortel trunk set up

The Final step is creating an outbound route for outgoing calls. The name of the route is FaktortelOutbound, and use 0 as prefix for all outgoing calls. After finish all the steps above, the asterisk based VOIP PBX is ready for making calls.

In Virtual Phone Book we are use web URL format to make web calls, the format is like "italkto:// phoneNumber /2000:2000@PBX server location", then this particular URL will sent to iaxLite soft phone, through soft phone call the real PSTN landline or mobile number.

In order to control remote laptop or work station to make calls, we are running axis based web service on remote computers, locally we create proxy for remote web services, then Virtual Phone Book using these proxies to send call command in Soap format, and execute in remote computers.

4.4.2 Control virtual earth with multiple sensors

In virtual earth application, we use accelerometers on Sun SPOTs to capture hand posture. Tilt is then translated to normal input devices signal in OS. OS takes the responsibility to control NASA World Wind 1.4[14].

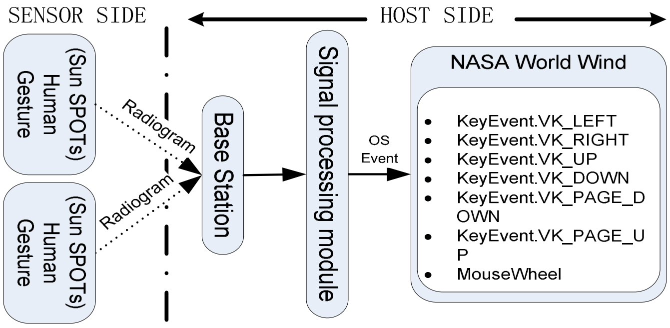


FIGURE 4.12. Virtual Earth architecture

As you can see in the Virtual Earth architecture shown in Figure 4.l2, two sensor nodes send data stream continuously, the base station receives the data and judge where the data coming from. Then, signal processing module translates events to keyboard or mouse actions.

The reason for choosing NASA World Wind is because the application gives user rich 3D interactive experiences, just as if you were really there. World Wind lets you zoom from satellite altitude into any place on Earth. So with Sun SPOTs multiple sensor nodes, user can control the virtual earth remotely, which makes NASA World Wind even absorbing. If user imaging these two sensor nodes as plane’s controller, with the right movement the application will give users a feeling like they are flying over the top of earth. User can not have this experience with keyboard and mouse.

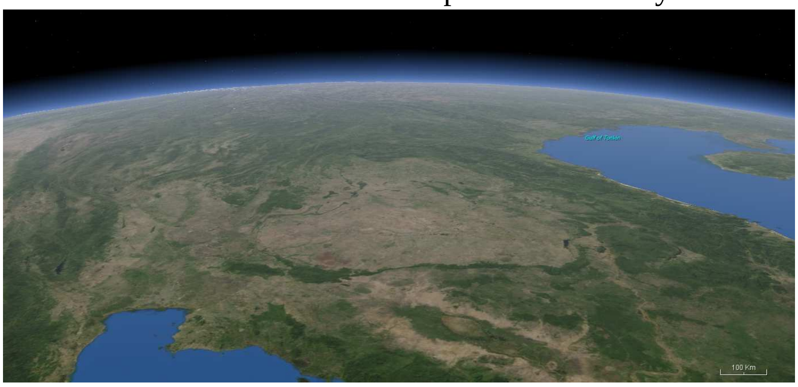


FIGURE 4.13. Fly over the earth

4.5 Other functions

In this project, we also integrated some other powerful components. Both users and developers get benefits from these functions.

First, our application can output all reading to CSV file. This CSV file can be imported to database for future research.

Second, we use experimental vector based Java Nimbus[25] visual style in our application. Since we believe, people mainly use gesture based application in front of big screen with high resolution. Compare with pixel based visual theme nowadays, vector based application could be gleamier in future.

Third, JFreeChart[26] is used to develop a real time data analyzer to make data stream human readable. Once the base station receive data stream, it will be displayed in this chart in real time, see Figure 4.14. With the help of this chart, we can visually analysis data patterns.

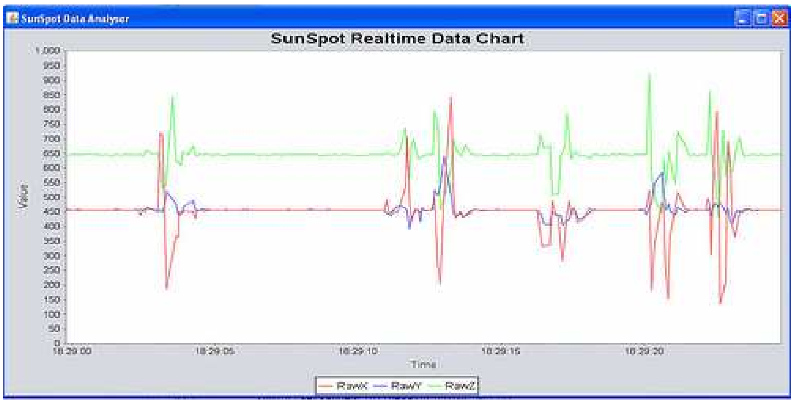


FIGURE 4.14. Sun SPOT real time data chart

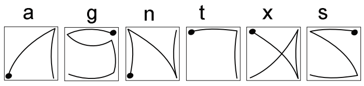
5. Experiments and Results

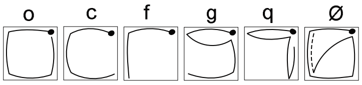
Systematic tests are required in order to evaluate the accuracy of the gesture training and recognition system. We will give the experiment methodology in first section. Results and analysis will be discussed in rest sections.

5.1 Methodology of experiment

We choose some characters from EdgeWrite[23] which is a unistroke text entry method developed by University of Washington. Its benefits include increased physical stability, tactility, accuracy, and the ability to function with very minimal sensing. It is originally designed for PDA, but we use it in our free space gesture recognition system.

Instead of test whole character set in this experiment, we carefully selected some characters which might be more suitable in our free space gesture application. Here below are two sets of gesture:

- ✧ Gesture set S1: 

- ✧ Gesture set S2: 

In S1, the shape and track of each gesture is different from each other. Recognizer should be able to identify these characters easily. In S2, 6 gestures are similar to each other. Compare to S1, the recognition accuracy of S2 should be lower. We want to know system's performance on similar gestures.

We defined the test cases. Now, we need to train the system based on these test cases. We trained two gesture sets respectively. Each gesture set, we trained 2 versions.

- ✧ Version 1 (V1): 5 samples per gesture. User specific, trained by an expert
- ✧ Version 2 (V2): 10 samples per gesture. Non user specific (relative), trained by two experts

For V1, each gesture in gesture set has 5 samples. For V2, each gesture in gesture set has 10 samples. Compare with V1, V2 contains more training samples, it should get better accuracy.

Each version contains both S1 and S2 as training data. Finally, we have 4 total training data sets.

	S1	**S2**
V1	N1	N3
V2	N2	N4

TABLE 5.1. Sample matrix

We will perform 10 samples for each gesture to evaluate the accuracy of the system.

5.2 Performance analysis of recognition module

In the first experiment, we want to test the performance of HMM engine. The data stream is manually segmented through the switch on SPOT. It works similar to Nintendo Wii game console which force user to hold button B before gesture is finished. Table 5.2 depicts the summary accuracy. As we discussed before, it is very easy to get high accuracy when manual segmentation is used. Table 5.3 to Table 5.6 depicts the detail accuracy information on each test.

	S1	**S2**
V1	85%	75%
V2	96%	78%

TABLE 5.2. Summary of accuracy

In confusion matrices below, row means the actual gesture we performed; column means the recognized gesture by system. For example, in Table 5.3, cell **nn** (shadowed area) is 70%. It means 70% of gesture **n** has been correctly recognized by system. On the other hand, look at **na** and **ng**, 20% and 10% of gesture **n** has been incorrectly recognized as **a** and **g** respectively.

From experiments, we can see that the HMM engine works very well in this case. Even with minimum 5 training samples we can get roughly 80% accuracy. With the help of 5 more training data, it reaches 96%. Compare two gesture sets, the accuracy of S1 is higher than S2. It is because gestures in S2 are similar to each other.

85%	**a**	**g**	**n**	**t**	**x**	**s**
a	**70%**	30%	0	0	0	0
g	0	**100%**	0	0	0	0
n	20%	10%	**70%**	0	0	0
t	0	20%	0	**80%**	0	0
x	0	10%	0	0	**90%**	0
s	0	0	0	0	0	**100%**

TABLE 5.3. Confusion matrix of N1

96%	**a**	**g**	**n**	**t**	**x**	**s**
a	**100%**	0	0	0	0	0
g	0	**100%**	0	0	0	0
n	0	0	**100%**	0	0	0
t	0	0	0	**100%**	0	0
x	10%	0	0	0	**90%**	0
s	0	10%	0	0	0	**90%**

TABLE 5.4. Confusion matrix of N2

75%	**o**	**c**	**f**	**g**	**q**	**Ø**
o	**70%**	0	0	0	20%	10%
c	0	**30%**	0	50%	0	10%
f	0	0	**90%**	10%	0	0
g	0	0	0	**100%**	0	0
q	10%	0	0	40%	**60%**	0
Ø	0	10%	0	0	0	**100%**

TABLE 5.5. Confusion matrix of N3

78%	**o**	**c**	**f**	**g**	**q**	**Ø**
o	**80%**	0	0	0	20%	0
c	0	**90%**	10%	0	0	0
f	0	0	**90%**	0	10%	0
g	0	0	0	**40%**	60%	0
q	10%	0	0	20%	**80%**	0
Ø	0	10%	0	0	10%	**90%**

TABLE 5.6. Confusion matrix of N4

5.3 Performance analysis of segmentation module

As we explained in Section 3.2, in order to recognize natural human gesture, both recognition and auto segmentation are important. We focus on the combination performance of auto segmentation technique and HMM in the second experiment.

As we predicted, the accuracy is much lower than the accuracy when manually segment is used. From Table 5.7, we can see that the system we developed can not be used in practical. For the worst case, we get only 17% accuracy. Please refer to Table 5.10, the recognizer is totally confused that it can not tell any difference between gestures. The recognizer considers all gestures are **q**. The situation did not change too much when we change to richer library V2. Gesture **o**, **c** and **g** still can not be recognized in Table 5.11. The conclusion is threshold based algorithm we implemented is not very good. It can not handle segmentation work properly.

	S1	**S2**
V1	32%	17%
V2	43%	33%

TABLE 5.7. Summary of accuracy

However, the recognizer get benefit from gesture set S1 which looks different from each other. Although the recognition rate of gesture **g** and **x** still very low, the final accuracy is better than S2. Table 5.8 to Table 5.11 depicts the detail accuracy information on each test.

32%	**a**	**g**	**n**	**t**	**x**	**s**
a	**50%**	30%	0	0	40%	10%
g	40%	**0%**	0	0	0	60%
n	80%	0	**10%**	0	0	10%
t	10%	20%	0	**0%**	10%	80%
x	0	10%	0	0	**30%**	70%
s	0	0	0	0	0	**100%**

TABLE 5.8. Confusion matrix of N1

43%	**a**	**g**	**n**	**t**	**x**	**s**
a	**40%**	20%	40%	0	0	0
g	0	**10%**	0	0	0	90%
n	10%	10%	**40%**	0	40%	0
t	10%	0	0	**70%**	0	20%
x	0	10%	30%	0	**0%**	60%
s	0	0	0	0	0	**100%**

TABLE 5.9. Confusion matrix of N2

17%	o	c	f	g	q	Ø
o	**0%**	0	0	0	100%	0
c	0	**0%**	0	0	100%	0
f	0	0	**0%**	0	100%	0
g	0	0	0	**0%**	100%	0
q	0	0	0	0	**100%**	0
Ø	0	0	0	0	100%	**0%**

TABLE 5.10. Confusion matrix of N3

33%	o	c	f	g	q	Ø
o	**0%**	10%	40%	0	30%	20%
c	0	**0%**	70%	0	30%	0
f	0	0	**80%**	0	20%	0
g	0	0	20%	**0%**	80%	0
q	0	0	10%	0	**90%**	0
Ø	0	40%	0	0	30%	**30%**

TABLE 5.11. Confusion matrix of N4

Another thing we should notice is all these training and test gestures are performed by developers. If a new user tries to use this system, in order to keep high accuracy, user has to spend large amount of time to learn how to correctly perform a recognizable gesture, otherwise train their own gesture set is desired.

This experiment tells us the truth to design a useful and practical gesture interface. We need to follow[24]:

- **Do usability testing on the system.** Don't just rely on the designer's intuition.
- **Do avoid temporal segmentation if feasible.** At least with the current state of the art, segmentation of gestures can be quite difficult according to literature review.
- **Don't tire the user.** When a user is forced to make frequent, awkward, or precise gestures, the user can become fatigued quickly. For example, holding one's arm in the air to make repeated hand gestures becomes tiring very quickly.
- **Don't make the gestures to be recognized too similar.** For ease of classification and to help the user.
- **Don't require precise motion.** Especially when motioning in space with no tactile feedback, it is difficult to make highly accurate or repeatable gestures.
- **Don't increase the user's cognitive load.** Having to remember the whats, wheres, and hows of a gestural interface can make it oppressive to the user. The system's gestures should be as intuitive and simple as possible. The learning curve for a gesture interface is more difficult than for a mouse and menu interface, since it requires recall rather than just recognition among a list.

6. Future Work and Conclusion

In this project, we implemented our own gesture recognition system based on sensor technology and machine learning. We achieved highest 96% accuracy when data sequence is manually segmented. Auto segmentation algorithm we implemented is not as good as we expected. In future, to develop a usable system, how to correctly and efficiently segment data stream is the key issue we need to solve.

We learned a lot from this project, and got inspired. In our opinion, with proper extension, the system we built can do more things. Since the software part really does not concern what kind of data it receives. It only learns different patterns of data, so theoretically all data captured from sensors no matter what type they are can be fed to recognition engine to get trained and to recognize patterns latter.

We did not integrate multiple wireless nodes in this project. First, wireless communication may interference with each other. We believe, if ALOHA is used, it will introduce more delays which will tremendously affect the usability of this near real time application. Second, in real time application how to precisely synchronize time between multiple nodes is also hard to solve. So to take a consistent snap shot of state from all distributed sensors at a point of time is hard to achieve.

Another important issue is we believe that data mining on sensor network system and data stream system will become a popular research area. Based on the characteristics of massively distributed WSN, it is very difficult for human to identify complex patterns from thousands of readings. As far as we know, majority of WSN applications nowadays, do the data analysis off line. Complex events or series events can not be detected easily. In future, data mining algorithm should be distributed on each sensor node to identify events intelligently. We believe the autonomous and pervasive WSN and BSN applications will hugely change our way of living in foreseeable future.

7. Reference

[1] Richard Harper, Tom Rodden, Yvonne Rogers and Abigail Sellen, Being Human: Human-Computer Interaction in the year 2020, Microsoft Research Ltd, 2008

[2] Tablet PC. (2008, June 6). In Wikipedia, The Free Encyclopedia. Retrieved 09:51, June 7, 2008, from http://en.wikipedia.org/w/index.php?title=Tablet_PC&oldid=217509098

[3] Digital paper. (2008, May 6). In Wikipedia, The Free Encyclopedia. Retrieved 09:53, June 7, 2008, from http://en.wikipedia.org/w/index.php?title=Digital_paper&oldid=210660639

[4] Multi-touch. (2008, June 4). In Wikipedia, The Free Encyclopedia. Retrieved 09:54, June 7, 2008, from http://en.wikipedia.org/w/index.php?title=Multi-touch&oldid=217140238

[5] LaViola, J. J. 1999 *A Survey of Hand Posture and Gesture Recognition Techniques and Technology*. Technical Report. UMI Order Number: CS-99-11., Brown University.

[6] Brashear, H., Henderson, V., Park, K., Hamilton, H., Lee, S., and Starner, T. 2006. American sign language recognition in game development for deaf children. In *Proceedings of the 8th international ACM SIGACCESS Conference on Computers and Accessibility* (Portland, Oregon, USA, October 23 - 25, 2006). Assets '06. ACM, New York, NY, 79-86. DOI= http://doi.acm.org/10.1145/1168987.1169002

[7] Lee, Y.-H. and C.-Y. Tsai (2007). "Taiwan sign language (TSL) recognition based on 3D data and neural networks." Expert Systems with Applications (in press)

[8] Allen Yang, Roozbeh Jarafi, Philip Kuryloski, Sameer Iyengar, Shankar Sastry, and Ruzena Bajcsy, Distributed segmentation and classification of human actions using a wearable motion sensor network. *Workshop on Human Communicative Behavior Analysis*, CVPR 2008.

[9] Mantyjarvi, Jani, Juha Kela, Panu Korpipaa, and Sanna Kallio. "Enabling fast and effortless customisation in accelerometer based gesture interaction." MUM 2004.

[10] Sawada, Hideyuki, and Shuji Hashimoto. "Gesture recognition using an acceleration sensor and its application to musical performance control." Electronics and Communications in Japan 80:5, 1997.

[11] WHMS - Wearable Health Monitoring Systems, Electrical and Computer Engineering The University of Alabama in Huntsville, Retrieved 23:33, June 6, 2008 from http://www.ece.uah.edu/~jovanov/whrms/

[12] Body sensor network. (2007, December 18). In Wikipedia, The Free Encyclopedia. Retrieved 10:35, June 7, 2008, from http://en.wikipedia.org/w/index.php?title=Body_sensor_network&oldid=178750768

[13] Wireless sensor network. (2008, May 31). In Wikipedia, The Free Encyclopedia. Retrieved 10:43, June 7, 2008, from http://en.wikipedia.org/w/index.php?title=Wireless_sensor_network&oldid=216226546

[14] World Wind Java SDK, Retrieved 10:43, June 7, 2008, from http://worldwind.arc.nasa.gov/java/index.html

[15] MICA2 Data Sheet, Retrieved 23:33, June 6, 2008 from http://www.ece.uah.edu/~jovanov/whrms/www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf

[16] MICAz Data Sheet, Retrieved 23:33, June 6, 2008 from www.xbow.com/products/product_pdf_files/wireless_pdf/6020-0060-01_a_micaz.pdf

[17] Introduction Tmote Sky, Retrieved 23:33, June 6, 2008 from faculty.uml.edu/yluo/Teaching/AdvCompArch/reports/WeinbergZhangFinalReport.pdf

[18] SunSpotWorld - Home of Project Sun SPOT, Retrieved 23:33, June 6, 2008 from www.sunspotworld.com

[19] Crossbow Imote2.Builder, Retrieved 23:33, June 6, 2008 from www.xbow.jp/Imote2.Builder_kit.pdf

[20] Li, C., Zheng, S. Q., and Prabhakaran, B. 2007. Segmentation and recognition of motion streams by similarity search. *ACM Trans. Multimedia Comput. Commun. Appl.* 3, 3 (Aug. 2007), 16. DOI= http://doi.acm.org/10.1145/1236471.1236475

[21] GART, Retrieved 23:33, June 6, 2008 from http://wiki.cc.gatech.edu/ccg/projects/gt2k/gt2k

[22] HTK Speech Recognition Toolkit, Retrieved 23:33, June 6, 2008 from htk.eng.cam.ac.uk

[23] EdgeWrite Text Entry, Retrieved 23:33, June 6, 2008 from http://depts.washington.edu/ewrite/

[24] Matthew Turk, Microsoft Research, Gesture Recognition, Retrieved 23:33, June 6, 2008 from ilab.cs.ucsb.edu/projects/turk/TurkVEChapter.pdf

[25] Nimbus - A New Look-and-Feel for Swing, Retrieved 23:33, June 6, 2008 from https://nimbus.dev.java.net/

[26] JFreeChart, Retrieved 23:33, June 6, 2008 from http://www.jfree.org/jfreechart/

[27] BATIK GUIDE, [Online], available from: http://xml.apache.org/batik/svggen.html

[28] Sagar. M.S, “An SVG Browser for XML Languages”, Theory and Practice of Computer Graphics, June 2003.

[29] Asterisk GUIDE, [Online], available from: http://www.asterisk.org/

[30] Abbasi.T Prasad.S, Seddigh. N., Lambadaris I., “A comparative study of the SIP and IAX VoIP protocols”, Electrical and Computer Engineering, 2005