A Distributed Wearable System based on Multimodal Fusion

Il-Yeon Cho[1], John Sunwoo[1], Hyun-Tae Jeong[1], Yong-Ki Son[1], Hee-Joong Ahn[1], Dong-Woo Lee[1], Dong-Won Han[1], and Cheol-Hoon Lee[2]

[1]Digital Home Research Division,
Electronics and Telecommunications Research Institute, Daejeon, Korea
{iycho, bistdude, htjeong, handcourage, hjahn, hermes, dwhan}@etri.re.kr
[2]System Software Laboratory, Department of Computer Engineering
Chungnam National University, Daejeon, Korea
clee@cnu.ac.kr

Abstract. Wearable computer can be worn anytime with the support of unrestricted communications and variety of services which provides maximum capability of information use. Key challenges in developing such wearable computers are level of comfort that users do not feel what they wear, easy and intuitive user interface and power management technique. This paper suggests a wearable system that consists of a wristwatch-type gesture recognition device and a personal mobile gateway that functional input/output modules can be freely plugged in and taken out. We describe our techniques implemented during our wearable system development: 1) multimodal fusion engine that recognizes voice and gesture simultaneously, 2) power management technique and 3) gesture recognition engine. Finally, we evaluate the performance of our multimodal fusion engine, and show the power consumption measurement data of our system built with the power management technique.

1 Introduction

People these days do not rely on PCs anymore as wire(less) internet spreads with the trend of computers, communications, and electronics appliances merging together into one. Instead, there is an increasing demand for new information terminal devices which can connect to the network anytime and anywhere with a method that's most familiar and convenient. These information terminal devices enable us to use variety of information freely and conveniently. This phenomenon tells us a factor which influences the success or failure in this digital industry; how many diverse tasks we do is not a factor, instead, it is how easy and simple we do the tasks with a user-friendly and intuitive interface. Computer, fashion and clothing industry are merging together into one in order to satisfy the needs of user interface, and these trials are putting ahead the appearance of new-concept information terminal devices.

Wearable computers provide efficient services with various functions by combining functions of personal devices that are scattered and overlapped [1]. Sony developed GestureWrist that recognizes user's hand gesture as an input [2]. U. Anlinker studied trade-offs between the required computing and the allocation of communication resources during the wearable system design process where the wearable system has input/output module distributed on human body [3].

In this paper we suggest a distributed wearable system that is constructed with the WPGB (Wearable Pointing and Gesture Band) and WPS (Wearable Personal Station). WPGB is a small wristwatch type device loaded with a compact gesture engine supporting a low power operation and WPS is loaded with the multimodal fusion engine that analyzes the gesture recognition result from WPGB and voice recognition result from the WPS itself so that the user's intended command can take place.

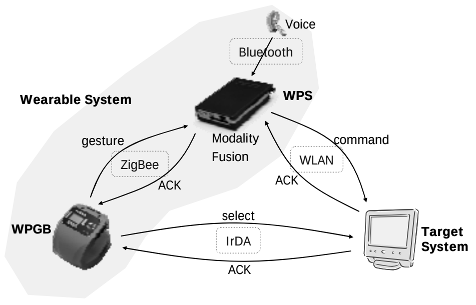


Fig. 1. Design of Wearable System

2 System Architecture

This section describes our proposed system design concept with the structure and functions of hardware and software.

2.1 Design Considerations

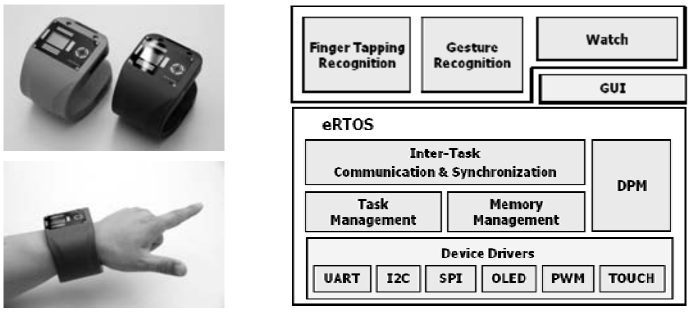
Fig. 1 illustrated the construction of suggesting system. As the name stands, the main function of WPGB is to select the object that needs to be controlled and send commands using arm gestures. Default function when not selecting the other objects is to control the wearable system on the user. It is designed to handle applications such as selecting and controlling one of the devices near the user, and transferring current contents or services to other devices freely. Applications such as content and service transfer are possible between the two WPGB users. WPGB communicates to WPS through Zigbee communication and uses IrDA for selecting devices to be controlled.

Multimodal fusion engine runs in WPS so that it can merge a gesture recognition result from WPGB and voice recognition result from the WPS and recognize to one final command. WPS has a Text-To-Speech (TTS) module as well to give user feedbacks according to recognition results.

2.2 WPGB (Wearable Pointing and Gesture Band)

We designed WPGB to have small form factor and power consumption. For the processor we used i.MX21 [4] that supports voltage scaling and frequency scaling. Multi-Chip Package (MCP) memory and uni-colored OLED are used to build WPGB. We made the power supply circuit that changes the system power level dynamically through our application program.

There is a Kionix KXP84 3-axis accelerometer sensor [5] and a piezo-electronic sensor equipped in the WPGB in order to recognize gestures and distinguish the starting point of each gesture command using finger tapping recognition, respectively. Additionally to give the user feedbacks, there are small-sized vibration motor, buzzer and an OLED. Fig. 2 shows our WPGB prototype.



(a) Prototype (b) Software Structure

Fig. 2. Wearable Pointing and Gesture Band

Operating Systems that runs WPGB is ETRI-RTOS (eRTOS). eRTOS (see Fig. 2(b) has basis from the compact, low power operating system called iRTOS [6]. Current kernel size of the eRTOS is around 25KBytes and it operates well in the low capacity RAM/ROM systems.

2.3 WPS (Wearable Personal Station)

WPS is a main platform of our system suggested through this work. It has a performance of an average PDA system with the size and shape that is convenient to carry. It provides functions for a personal mobile gateway. In order to provide such functions, it supports WLAN, Bluetooth, Zigbee communication, and has dimension of 80mm x 54mm x 19mm. It runs on PXA272 [7] processor and supports both embedded Linux and WinCE operating system. It is designed to be geared and used with various peripheral modules in a selective way depending on the services. Fig. 3 illustrates the WPS module which can change its functionality and system configuration by pairing with various output modules.

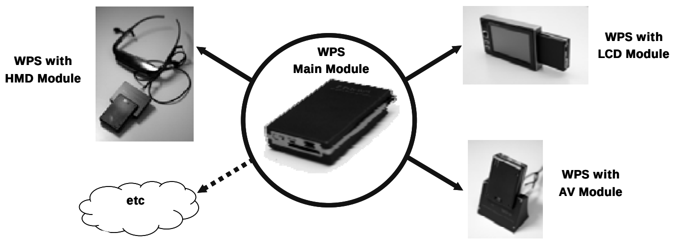


Fig. 3. Expandability of WPS

3 Core Technologies

This section describes core technologies that are implemented in our system suggested through this paper. The core technologies are multimodal fusion engine, low power scheme, gesture recognition engine and gesture segmentation cue method.

3.1 Multimodal Fusion Engine

We used PowerASR voice recognition engine from HCILAB for the WPS voice recognition. PowerASR is the voice recognition engine developed for the embedded system use that is user-independent and supports the recognition of variable vocabulary (maximum of 200 words in a database) [8].

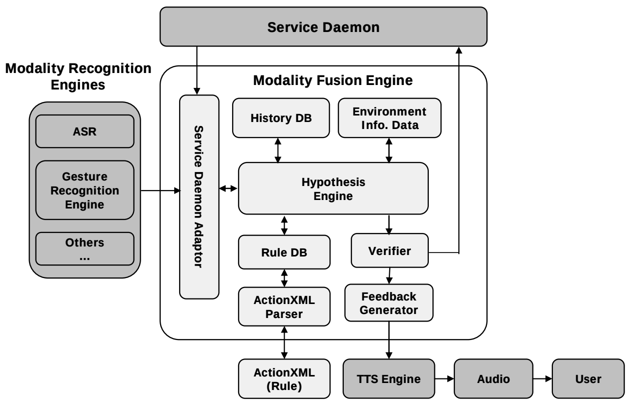


Fig. 4. Structure of Multimodal Fusion Engine

Advantages of the multimodal input system compare to the uni-modal system are known as the reduced uncertainty and improved input accuracy in noisy environment [9]. Methods of multimodal fusion are divided into two in overall: input feature level fusion and semantic level fusion [10]. Our study chose semantic fusion strategy since it is easy to add a new modality, and is convenient in maintaining. Software architecture of our multimodal fusion engine is shown in Fig. 4.

Operation sequence of the multimodal fusion engine is as follows. Fusion engine extracts rules from the *ActionXML* file using the *ActionXML Parser* then saves the rules in *Rule database*. User and system events that are recorded and time-stamped in *Service Daemon* and *Modality recognition engine* are sent to *Hypothesis engine* through the *Service Daemon Adaptor*. *Hypothesis engine* deduces what user intended to command by referencing *System Environment Information Data*, *History DB and Rule DB* where the past deduced results are recorded. Deduced results in this manner are verified in *Verifier* and are sent to *Service Daemon*. *Verifier* finds an error and notifies user via *Feedback Generator*.

Voice recognizer returns two candidates of recognition result, and gesture recognizer returns one that is the most likely to what the user intended to commend. Combinations of the two (voice + gesture) recognition results are assigned to the commands according to user's needs and registered in *ActionXML* file. The following example shows how the two commands in different modalities are combined into one in order to achieve mutual complementary benefits. We get the mutual complementary benefits when the voice and gesture commands are defined for the same purpose. For instance, both the voice command "NEXT" and gesture command "[RIGHT]" can be mapped to the "→" key in the keyboard. It can be useful in a noisy environment where the voice recognition results are often poor.

Within the mutual complementary policy described above, the following cases will be considered "recognition fail" by the multimodal fusion engine and there will be no resultant action.

- Both the voice and gesture recognizer fail to give recognition results.
- Voice and gesture recognizer give different recognition results.

In contrast, the following are the cases of "recognition success".

- Both the voice and gesture recognizer give same recognition results.
- Only one of the two recognizer gives a successful recognition result.

Excluding the "recognition fail" case shown above, user can define the combinations of voice and gesture in the *ActionXML* file and assign them to the final commands user wants. Or, the user can find the cases of "recognition success" shown above and register them in the *ActionXML* file. Combinations that are not registered in the *ActionXML* file are processed as a recognition failure. However, for the particular combinations of voice and gesture that are often recognized wrong, we can overcome this to a certain level by registering them to *ActionXML* file. The following is an example of overcoming such case.

If the voice recognizer frequently misses to recognize the user's voice command "NEXT" and confuses as "TEXT", then the user can register "TEXT" as same as "NEXT" in ActionXML file so that the result of "TEXT" is recognized by the fusion engine as "NEXT". In this case, user may not want to assign "TEXT" command.

Our multimodal fusion engine suggested through this work is designed light enough to be operated together with the voice recognizer, speech synthesizer, and gesture recognizer within an embedded system such as our WPS system. It is implemented to reduce the deduction (reasoning) procedure which takes a considerable amount of time in the fusion engine by letting the user to define multimodal fusion rules into *ActionXML* file directly. Finally it results in the load reduction of our fusion engine so that it can be run on a lower performance embedded system that has small resources.

3.2 Dynamic Power Management (DPM)

WPGB hardware supports the following features for DPM application: WPGB uses Freescale's i.MX21 as CPU. The CPU frequency can be configured as 133MHz and 66MHz. System bus frequency can be changed from 16MHz ~ 66MHz. The CPU core voltage can be changed from 1.4V ~ 1.5V.

DPM and policy in WPGB are implemented based on the methods suggested in [11-13] while considering WPGB hardware and software property.

3.3 Gesture Segmentation Cue

Generally, a hand gesture occurs subsequently after the previous hand movements that maybe meaningless. In order to correctly analyze a user's intended gesture command during such continuous hand movements, recognition engine has to know the starting point and ending point of the gesture. This kind of gesture segmentation issues are being a problem [14]. In our early development stage, we solve this gesture segmentation problem by using a button that can be worn on a finger as illustrated in Fig. 5(a) so that user can press the button only when making a meaningful gesture. However, this system can be inconvenient for everyday life. In order to overcome this inconvenience, we developed *FingerTapButton* that recognizes the bone-conduction sound by touching a thumb and second finger and uses it as a hand gesture segmentation cue, which can avoid the use of mechanical buttons. WPGB only monitors data from the time when the user taps his or her fingers until the user finishes gesture command and stays still. Fig. 5(b) illustrates the basic principle of *FingerTapButton*.

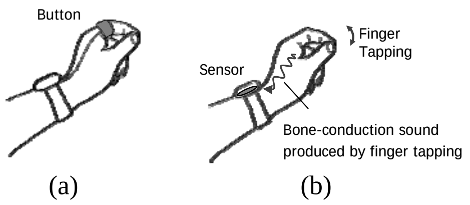
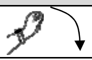
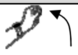
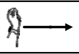
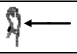
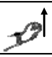
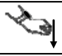
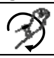
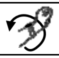


Fig. 5. Basic principle of *FingerTapButton*

3.4 Gesture Recognition Engine

We define 8 commands that are sufficient to control general multimedia appliances. Each command is mapped to forearm gestures by considering our intuitive gestures used in the real world.

Table 1. Defined gesture table.

Commands	Gestures	Commands	Gestures
Device Selection		Device Cancel	
Right/Next		Left/Previous	
Volume up		Volume down	
Play		Stop	

The x, y and z sensor data will be sampled only when the user makes meaningful gestures starting with *FingerTapButton*. Each gesture data is normalized to have the fixed height of data range, and sub-sampled to filter out the noise and reduce the data size to lessen the load when the engine analyzes the input. The preprocessed data is analyzed and characterized in terms of 1) the maximum and minimum values of the acceleration along each axis, 2) where they occur in time-vs.-acceleration plots and 3) quantitative comparison of them in order to find parameters for the software recognition engine so that it can recognize each command. In addition, as the command set increased, more geometric characteristics were considered such as the starting/end value and vertex (local maxima/minima) locations of each input vector. This method of extracting characteristic information and classifying them with the rule-based recognition engine is used to distinguish gesture commands [15].

4 System Evaluation

This section describes the use of our multimodal fusion engine that is implemented in our suggesting wearable system.

4.1 Recognition Enhancement through Multimodal Fusion

As mentioned previously, advantage of multimodal fusion engine is its improved recognition rate compare to that of individual recognition engine. In order to measure the performance of our fusion engine, we used 6 gestures and 66 voice commands. There were 5 test subjects (all males) and each subject was asked to make each combinational voice and gesture command for 10 times.

Fig. 6 shows the comparison of voice only recognition rate, gesture only recognition rate, and voice + gesture recognition. As shown in the Fig. 6, recognition rate using the fusion engine is better than using the voice or gesture recognition engine alone.

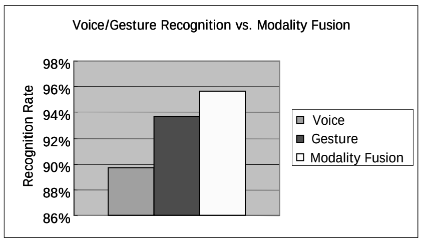


Fig. 6. Recognition rate of Voice/Gesture vs. multimodal fusion

4.2 Power Consumption of WPGB

WPGB hardware is designed to minimize the power consumption, and the operating system runs the WPGB supports a power supply control interface for each component. In order to measure the power consumption, we measured the current of the system using Agilent 34411A digital multi-meter (in 2000 samples/sec) with the supply voltage of 3.7V from Agilent E3648A power supply.

Table 2. DPM Policy

WPGB Task	Operating State	WPGB DPM Policy		
		Default	Idle Scaling	Load Scaling
FINGERTAP	TASK	133/66@1.5V[1]	133/66@1.5V	88/44@1.45V
GESTURE	TASK+1	133/66@1.5V	133/66@1.5V	133/66@1.5V
ICONVIEW[2]	TASK-1	133/66@1.5V	133/66@1.5V	66/33@1.425V
IDLE	IDLE	133/66@1.5V	66/16@1.4V	66/16@1.4V

Considering the CPU requirements of the WPGB tasks, we decided the operating state and applied DPM policy defined in section 3.2. Table 2 shows the WPGB DPM policy of the four tasks.

Fig. 7 shows the measurement data for each policy. Portion noted as ① in the figure is when the WPGB recognizes the starting point of a gesture using the *Finger-TapButton* and gives vibration feedbacks through the vibration motor, ② is when it collects the accelerometer sensor data and recognizes the gesture, and ③ is when the gesture results are being displayed on OLED.

Average current and power consumptions are shown in Table 3. As illustrated in the table, we achieved an improvement of 30% in power saving under Idle Scaling policy when compared to non-DPM system where the Idle Scaling policy has a power saving mode only in the system idle state. Under Load Scaling policy which has power saving modes in both the idle state and task operation, there is almost an improvement of 33% in power saving.

[1] CPU frequency / system bus frequency @ CPU voltage
[2] Display of recognition results on the OLED

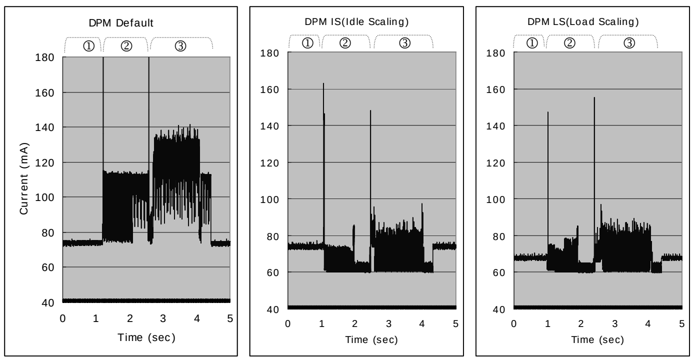


Fig. 7. Current Profile with DPM Policies

When considering the capacity of the battery (300mAh) that powers the WPGB, it has around 3 hours of system run time when DPM default policy is applied, however by applying the DPM Load Scaling policy its run time increases by an hour.

Table 3. Power Consumption of WPGB

Policy	Average Current Consumption (mA)	Average Power Consumption (mW)	Power Saving (%)[3]
DPM Default	98.50	364.46	-
DPM IS	68.11	252.01	30.85
DPM LS	65.63	242.83	33.37

5 Conclusions

We presented our prototype that consists of a wristwatch type Wearable Pointing and Gesture Band (WPGB) and a small sized Wearable Personal Station (WPS) that the functional input/output modules can be freely plugged in and taken out. WPGB is loaded with a compact low power gesture recognition engine in order to support gesture command based user interface in the wearable computing environment. WPS has a mass storage with a communication gateway and has multimodal fusion capability.

By measuring and analyzing the performance of our multimodal fusion engine, we showed that using both voice and gesture input in a mutual supportive way results in a better recognition rate compared to when using a single input modality. We also showed that using a DPM technique gives us power savings by measuring and analyzing the power consumption rate in our WPGB system where it is operated by the DPM enabled eRTOS.

[3] Compare to DPM Default

References

1. T. Starner: The challenges of wearable computing Part 1. IEEE Micro Vol. 21. No. 4. July (2001) 44-52
2. J. Rekimoto: GestureWrist and Gesture Pad: Unobtrusive Wearable Interaction Devices. Proc. IEEE International Symposium on Wearable Computers (2001) 21-27
3. U. Anliker et al.: A systematic approach to the design of distributed wearable systems. IEEE Trans. Computers, Vol.53. No.8. (2004) 1017-1033
4. i.MX21 Applications Processor Reference Manual, MC9238MX21RM, Rev.2 (2005)
5. ±2g Tri-Axis Digital Accelerometer Specifications, Part Number: KXP84-2050, Rev 1, Kionix Inc (2006)
6. H.S. Park, *et al.*: Design of Open Architecture Real-Time OS Kernel. KISS, Vol. 2. (2002) 418-420
7. Intel PXA27x Processor Family Developer's Manual (2006)
8. http://www.hcilab.co.kr
9. A. Corradini, M. Mehta, N.O. Bernsen, and J.C. Martin: Multimodal input fusion in human-computer interaction. Proc. the NATO-ASI Conference on Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management (2003) 18-29
10. P.R. Cohen, M. Johnston, , D.R. McGee, S.L. Oviatt, J. Pittman, I. Smith, and J. Clow: Quickset: Multimodal Interaction for Distributed Applications. Proc. the 5th International Multimedia Conference, ACM Press (1997) 31-40
11. B. Brock and K. Rajamani: Dynamic Power Management for Embedded Systems. Proc. IEEE International SOC Conference (2003) 416-419
12. IBM and MontaVista Software: Dynamic Power Management for Embedded systems. http://www.research.ibm.com/arl/projects/dpm.html (2002)
13. CE linux forum: Power Management Specification _R2", http://tree.celinuxforum.org/CelfPubWiki/Power ManagementSpecification_5fR2 (2004)
14. P.A. Harling and A.D. N. Edwards: Hand tension as a gesture segmentation cue. Proc. Gesture Workshop on Progress in Gestural Interaction (1996) 75-88
15. J. LaViola: A Survey of Hand Posture and Gesture Recognition Techniques and Technology. Technical Report CS99-11. Dept. of Computer Science, Brown University. Providence, Rhode Island (1999)