Actual Remote Control: A Universal Remote Control using Hand Motions on a Virtual Menu

Dong-Woo Lee, Jeong-Mook Lim, John Sunwoo, Il-Yeon Cho and Cheol-Hoon Lee

Abstract — *Consumer electronic devices are becoming more complex and diverse. In order to provide users with an easy and convenient way of controlling various consumer electronics, it is necessary to use a universal way to control various devices. This requires using a user-friendly and intuitive interface.*

We introduce a wristwatch-type of remote that offers a unified way to control various devices. Total of seven gestures that are based on hand motions are defined. These seven gestures are suitable to the wearable wristwatch-type device. For a variety of systems, these gestures can be used in the same way. Gestures are not designed and analyzed in a discrete manner; instead, they are designed in continuous hand motions and analyzed in gesture commands. Also, a method of modeling a virtual menu from the menu on a consumer electronic device is introduced. With the virtual menu, the user is able to control various devices by using gestures. The virtual menu has to reflect hand motion characteristics and represents functions of electronic appliances. The virtual menu is implemented in XML to represent the basic menu construction and its properties. In order to use virtual menu, we introduce how a user's hand motions can be used in a fast and effective way. Finally we compare our wearable remote control to the conventional remote control through user tests in terms of user convenience and efficiency[1].

Index Terms — remote control, wearable, hand motion recognition, virtual menu.

I. INTRODUCTION

There have been many different kinds of research done to find efficient control method for consumer electronics [1], [6], [7], [9]. Remote control is a representation of controlling electronic devices and has been used widely for decades. However, having several devices results in needing more remote controls. Thus finding an efficient way of controlling multiple devices collectively is becoming important.

Many studies suggest using devices with user friendly voice and gesture commands. Although voice based interactions are known as the most convenient interface, the voice recognition is weak in noisy environments and requires high processing. This makes it difficult to utilize on small embedded devices. Gestural interface can be done using vision (often camera) or motion sensors [8], [9], [12]. However, camera vision for gesture recognition requires fixed cameras in a closed location and is often difficult to maneuver. Moreover, for the real time control, the camera vision based recognition usually requires extensive computing algorithm in a heavy system. Unlike vision, motion sensor based control requires relatively small computing power, and MEMS technology provides a small size sensor that suits to a mobile environment.

To manage a set of diverse electronics, simple and universal method is required. In general, most electronic devices have front control panels, displays, or simple switches. Devices such as audios, refrigerators, and washing machines have front control panels. Televisions and projectors contain a control menu on the screen display. Some electronics such as lamps, curtains and gas valves only require simple switches. Different types of devices operate using different control styles. To find a convenient and easy way to control devices, all kinds of devices need to be considered.

The objective of our research is to resolve the issue mentioned above. Taking advantage of the MEMS based motion sensor, we developed a wearable wristwatch-type of controller. And we proposed a method of controlling various devices in a unified way. For easy control, the virtual menu was developed to resemble the actual menu on electronic devices. We introduce a method of modeling virtual menus from the actual electronic devices.

Major contributions of this work are 1) designing simple and effective hand motion gestures for controlling diverse devices, and 2) introducing a universal method of modeling the menus on various devices.

Section 2 discusses our previous research on gesture band development and its results. Section 3 introduces ARC service architecture and system implementations. Section 4 analyses usability tests by comparing ARC to the conventional remote control. Some of the development issues are discussed in Section 5, followed by conclusions and future research in Section 6.

[1] This work was supported by the IT R&D program of the Korean MKE and IITA (2008-F-048, Wearable Personal Companion for u-computing collaboration)

D.-W. Lee is with the Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: hermes@etri.re.kr).

J.-M. Lim is with the Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: jmlim21@etri.re.kr).

J. Sunwoo is with the Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: bistdude@etri.re.kr).

I.-Y. Cho is with the Electronics and Telecommunications Research Institute, Daejeon, Korea (e-mail: iycho@etri.re.kr).

C.-H. Lee is the corresponding author and is with the Computer Engineering Department, Chungnam National University, Daejeon, Korea (e-mail: clee@cnu.ac.kr).

Manuscript received May 20, 2009

II. Our Previous Work

Previously, we developed a Wearable Pointing and Gesture Band (WPGB) for controlling consumer electronics using gesture commands [1]. It was based on one 3-axis accelerometer sensor in order to recognize forearm gestures. WPGB can be worn on the wrist. This makes it suitable for mobile environments where hands-freeness and portability are necessary. In a recent study, wearable watch-type devices have shown the most accessibility [2]. WPGB enables natural gesture interface which shows many advantages over the conventional universal remote control. However, its disadvantage is that the gestures used are individually mapped with particular functions. For example, in the case of volume or channel control, where the continuous or repetitive control is required, a user has to repeat the same gesture in a discrete manner. Also, extending the gesture command set is difficult, because the recognition rate can diminish and users tend to remember and use only a limited number of gestures [3], [7].

III. Implementations

In order to improve upon the disadvantages of our previous works and other approaches, we have developed an Actual Remote Control (ARC) with better functionalities.

During the ARC development, our major considerations were as follows.

① Designing an accessory type of controller is needed. The controller (ARC) should be as small and lightweight as the conventional remote controls. It should provide hands-free operations and comfort for the mobility.

② The ARC should provide a universal method of controlling, in order to provide users with an easy and convenient way of controlling a variety of electronic devices.

i. Gestures need to be easy and familiar to users. Designing these gestures is needed for the quick and spontaneous controls. Our research considered and designed hand motions.

ii. A way of modeling the menu should be applicable to diverse systems, so that users can control these systems in the same way. In the modeling, it is important to maintain the shape of the original menu which is on the target electronic device. This provides the user with an easy control without the needs of learning them. In our research, a method of creating virtual menu was developed in order to achieve the above.

③The ARC has to be compatible to ordinary consumer electronics.

The ARC is implemented by considering the above. In this section, we introduce the ARC implementations in detail.

A. Overall Architecture of ARC

An ARC is a wearable wristwatch-type remote controller; it can serve as a universal remote control for various consumer electronic devices. As illustrated in Figure 1, our overall service architecture includes a ARC, a Device+ which adds compatibility to ordinary consumer electronics, and target consumer electronics to be controlled. The Device+ is an intermediate prototype for making ordinary electronics controllable via ARC. Future versions of the Device+ are expected to be embedded in the consumer electronics themselves.

B. ARC and Device+ Hardware

ARC hardware consists of a main module (ARM processor, 8M flash, and 4M SRAM), a sensor module (Gyro, Accelerometer, and Piezoelectric sensor) for hand motion tracking, a feedback module (speaker, vibrator, and LED) and a Bluetooth module for communication. To add smoothness and to ensure comfortable wear, 5 rigid PCBs are connected through flexible PCBs as shown in Figure 2. It has better a form factor, design, and battery power than the previously developed WPGB.

Device+ consists of ARM based main module and peripherals. In order for it to bridge ordinary consumer

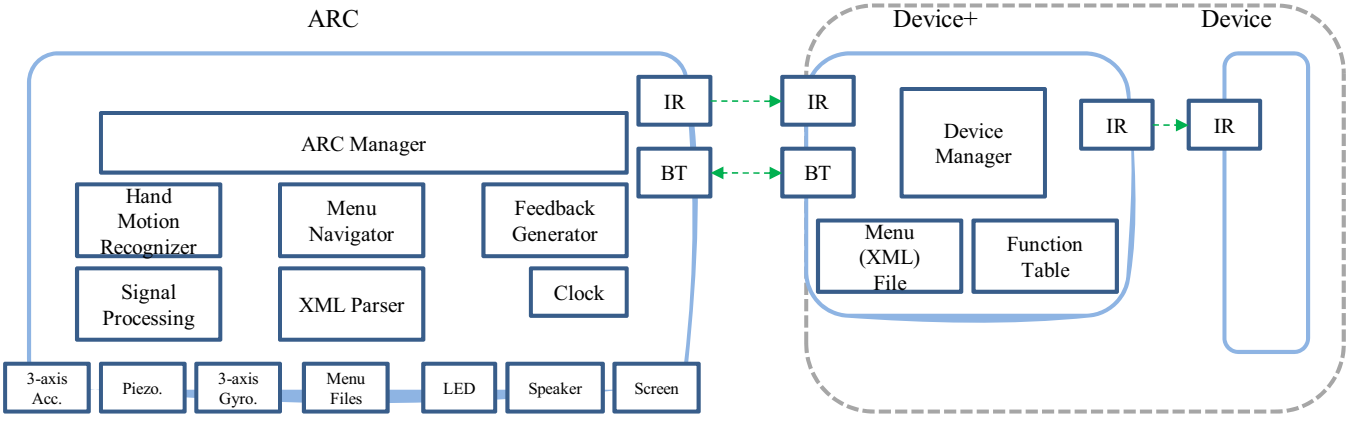

Fig. 1. Overall Service Architecture

electronics to a developed ARC, the Device+ is equipped with IrDA and Bluetooth (BT) communication. Detailed operations are presented later in the paper.

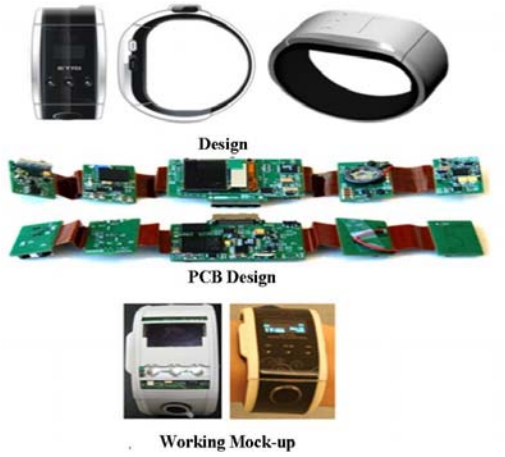

Fig. 2. ARC Hardware

C. *Hand Motion Recognition*

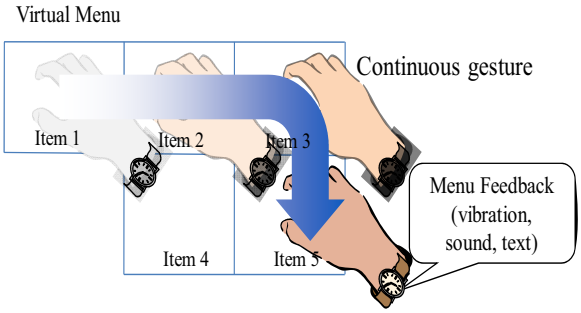

Fig. 3. Hand Motion for Menu Navigation

For menu navigation and selection, we collected patterns of hand motions seen in generic device controls such as adjusting a level bar or dial. Then with the collected patterns, representative gestures are modeled. Finally, we could extract simple arm movements that can make our system user-independent. ARC continuously recognizes four directions (RIGHT, LEFT, UP and DOWN) FingerTapButton (CUE) [1], and wrist rotations (SPIN_LEFT, SPIN_RIGHT). The FingerTapButton recognizes the bone-conduction sound by tapping the thumb and second finger.

ARC recognizes hand motions (or strictly forearm motions) through analyzing gyro and accelerometer sensors. Panning one's hand to the right by 10 degrees generates the RIGHT command. For instance, continuously moving 70 degrees is equivalent to seven RIGHT commands. Sensitivity is adjustable by setting threshold in degrees.

D. *Virtual Menu and Menu Navigation*

Ordinary control panels in electronic appliances have control elements such as a button, dial, or a level bar. Whereas display devices, like a TV, have On-Screen-Display (OSD) type graphical menus. This study is to achieve unified control of various consumer electronics, thus portraying different style of menus in one universal menu is important. To meet the goal, we developed a virtual menu that expresses different menu styles. A virtual menu maps elements in an actual control panel to virtual control elements on a virtual 2-d space. Virtual menu gives the impression that control panel is right in front of users even though actual control panel is at a distance. The shape of virtual menu is well suited to be controlled by simple hand motions.

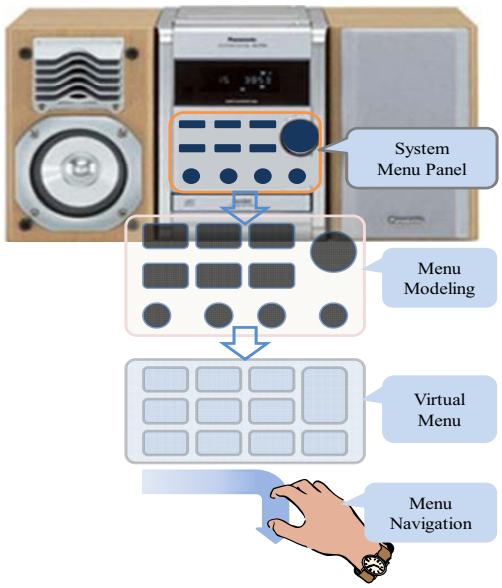

Fig. 4. Virtual Menu Modeling

To control an audio system, Figure 4 illustrates the sequence of how a virtual menu is modeled so that hand motions can be used. The layout of the virtual menu resembles the actual control panel of the electronic device, and by using hand motions, the user can navigate through the menu. If the actual buttons on the audio system have appropriate LED feedback as the user navigates through the virtual menu, it can be more effective.

Fig. 5. Navigating Virtual Menu

Figure 5 shows a user controlling a device with a display. The user can navigate the virtual menu as though the actual control menu is at a reaching distance. To achieve this, the user is also provided with a text and audio feedback.

TABLE I
MenuXML Elements and Attributes

Elements	Description	Attributes (O: optional , M: mandatory)	Events
menuxml	Top element of Menu xml Child element: device+ Note) + means that it has more than one element.	version(O): version value	
device	Device element Child element: menu+	id(M): device ID(MAC Address) name(O): device name model(O): model No.	onload
menu	Element that represents a system menu Child element: item+	id(M): menu id(INT) title(O): menu title type(M): menu type (one of grid, ring, pie) default(O): default menu item index (INT) parent(O): parent menu id	onclick, onleft, onright, onup, ondown, onspinleft, onspinright
item	Button element of menu Child element: N/A	index(M): position of the button in the menu (row and col index) fid(M): function id text(M): button name recursive(O): recursively select corresponding button for particular motion, ex) recursive="left" skip(O): skip corresponding button when navigating nomoveout(O): applicable to buttons on the edge, buttons with this attribute does not support menu-moveout	

E. Using ARC to Control Target Electronics

Communication interfaces between ARC and Device+ are IrDA and Bluetooth (BT). Using directional IrDA enables the user to point and select a particular device he or she wants to control. The *ARC Manager* is a program that runs and handles the FingerTapButton(CUE) differently according to states. CUE operates in 3 different modes as follows:

① CUE is interpreted as DEVICE_SELECT, when none of the devices have been selected and/or established a BT communication channel.

② Once a device is selected and has retrieved a menu, CUE is interpreted as VIRTUAL_MENU_ON. In other words, CUE represents the beginning of menu control via hand motions.

③ While the virtual menu is in ON state, CUE is interpreted as MENU_ITEM_SELECT (i.e., CLICK).

In detail, when the ARC is disconnected, the *ARC Manager* sends an ID (MAC address) of the ARC to the Device+ through IrDA after the user points to the target device and makes a CUE. A BT connection is established after a *Device Manager* in the Device+ receives an ID. Then the *Device Manager* sends its ID back to ARC. After the ARC and the Device+ are paired via BT, data are transmitted through the BT link. The link between the ARC and the Device+ remains active until the user selects another device.

The ARC downloads the virtual menu when needed from selected electronic appliances. As the ARC is receiving the ID of Device+, the *ARC Manager* determines whether it already has the corresponding menu by looking at the device ID received from the Device+. If the menu is not present in the ARC, then it requests a menu to the Device+ for download. After the download is completed, the user can navigate through the menu by moving his or her arm in four directions continuously, as if one is touching a 2-D menu in the air.

The Device+ receives control messages (through a BT link) from the ARC. The *Device Manager* in the Device+ has a function table for the target electronic device. The function table is a list of control messages that correspond to an actual IrDA code which needs to be delivered to a target electronic device. As it receives the control message from the ARC, it compares the control message with the function table and finds which IrDA code needs to be sent to the target device. Finally the Device+ sends the IrDA code to the target device (for instance, a TV) to make the target device follow an action (Fig. 1.).

After a user selects a device and the ARC downloads a menu for the device, the user can start controlling the device by conducting a basic operation which can be expressed as follows:

CUE(VIRTUAL_MENU_ON, load default menu)
□→Navigation
□→CUE(CLICK, VIRTUAL_MENU_OFF)

In the above operation, the Navigation includes menu transitions. By using the CUE as the start and end of the

operation, meaningful gestures are easily distinguished from the noise.

F. MenuXML

A virtual menu is an artificial menu that is controlled by a user's hand motion. By using the virtual menu, an electronic appliances' front control panel or on-screen display (OSD) are controlled. So the virtual menu design needs to be able to reflect hand motions well, and also needs to have a high compatibility between heterogeneous systems to control diverse electronic appliances.

We implemented MenuXML to describe the virtual menu with XML to satisfy these requirements. MenuXML has advantages that can systematically represent basic menu construction and properties. It is well suited for continuous hand motions that are different from discrete gesture commands. To reflect these hand motion properties, MenuXML supports several attributes to represent hand motion properties. These properties include continuous menu navigating, recursive selection, skipping menu item, exiting menu region, etc.

Some functions are suitable with continuous hand motion control, for instance, volume or channel control. We can design an event of one directional continuous hand motion with a *recursive* attribute. To exit and disable the virtual menu region, a user can naturally drop his or her arm in a downward motion. It is also possible to prevent exiting the virtual menu by dragging the menu in one direction. This can be done by giving a *nomoveout* attribute on the menu item.

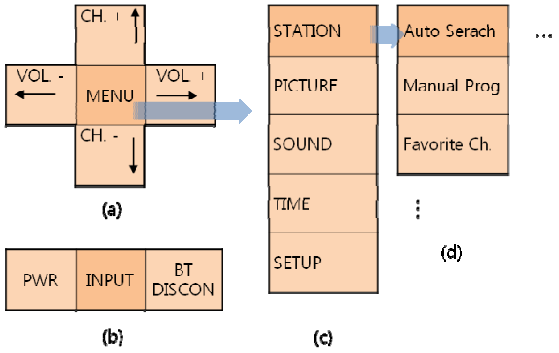

Fig. 6. TV Menu (a) shows the volume and channel control menu, (b) shows input menu, (c) shows the submenu of MENU on (a), (d) shows the submenu of STATION on (c)

Table I shows the elements and attributes of the MenuXML. It defines several events such as ONCLICK, ONLOAD, ONLEFT, ONRIGHT, ONUP, ONDOWN, ONSPINLEFT and ONSPINRIGHT. They are used to make more complex operations, such as menu transitions and modifications. Those complex operations need to utilize attributes of the element (for example, menu item enable or disable, changing default menu item, etc.). The events are described with a script.

Figure 6 shows how the actual TV OSD menu is modeled before describing it in MenuXML.

Figure 7 shows an example of MenuXML where the TV menu has been modeled from Figure 6.

```
<?xml version="1.0" encoding="ksc5601"?>
<menuxml version="1.0">
<device id="101112131415" name="TV" model="TopSync" onload="load:0">
<!-- Short-cut Menu -->
<menu id="0" title="Top Menu" type="grid" parent="" onspinleft="load:1" default="3">
<item index="1,2" fid="1" text="CH. UP" recursive="up"/>
<item index="2,1" fid="2" text="VOL. DOWN" recursive="left"/>
<item index="2,2" fid="3" text="MENU" skip="" onclick="load:2"/>
<item index="2,3" fid="4" text="VOL. UP" recursive="right"/>
<item index="3,2" fid="5" text="CH. DOWN" recursive="down"/>
</menu>
<menu id="1" title="Input" type="grid" parent="" default="2">
<item index="1,1" fid="1" text="POWER"/>
<item index="1,2" fid="2" text="INPUT" onclick="load:30"/>
<item index="1,3" fid="3" text="DISCONNECT"/>
</menu>
<!— menu depth 1-->
<menu id="2" title="Main Menu" type="grid" parent="" default="1">
<item index="1,1" fid="1" text="STATION" onright="load:3" onclick="load:3"/>
<item index="2,1" fid="2" text="PICTURE" onright="load:4" onclick="load:4"/>
<item index="3,1" fid="3" text="SOUND" onright="load:5" onclick="load:5"/>
<item index="4,1" fid="4" text="TIME" onright="load:6" onclick="load:6"/>
<item index="5,1" fid="5" text="PIP" onright="load:7" onclick="load:7"/>
<item index="6,1" fid="6" text="SETUP" onright="load:8" onclick="load:8"/>
</menu>
<!— menu depth 2 -->
<menu id="3" title="STATION" type="grid" parent="" default="1">
<item index="1,1" fid="1" text="Auto Search" onleft="goparent" onright="load:9" onclick="load:9"/>
<item index="2,1" fid="2" text="Manual Prog" onleft="goparent" onright="load:10" onclick="load:10"/>
<item index="3,1" fid="3" text="Favorite Ch." onleft="goparent" onright="load:11" onclick="load:11"/>
</menu>
<menu>
...
</menu>
…
</device>
</menuxml>
```

Fig. 7. MenuXML structure and example

IV. Experiment and Results

An experiment was conducted in order to evaluate a conventional remote control, WPGB, and an ARC. Twelve participants took part in the experiment: eight males, and four females with an average age of 34. There were two objectives in this experiment. The first objective was to compare the task completion time on given tasks, and the second was to evaluate usability of each controller. Each participant had 5 minutes to practice using the ARC and the WPGB, respectively.

In order to compare the time which takes to complete each given task, we have defined the objectives in a total of twelve tasks. The defined tasks are shown in Table II. Each task comprises of basic combinations of TV controls.

TABLE II
Tasks Defined for User Test

	No.	Task
Simple Task	1	2 channels up
	2	2 volumes down
	3	2 channels up and 2 volumes up
	4	4 channels up and 4 volumes up
	5	2 channels down and 2 volumes down
	6	4 channels down and 4 volumes down
Complex Task	7	2 brightnesses up
	8	2 contrasts up
	9	2 brightnesses down and 2 contrasts down
	10	4 brightnesses up and 4 contrasts up
	11	2 brightnesses up and 4 contrasts down
	12	4 brightnesses down and 2 contrasts up

Figure 8 shows the resulting completion time for simple tasks and complex tasks.

Fig. 8. Average Task Completion Time for Simple and Complex Tasks

The result showed that the WPGB showed the worst performance. This is due to the characteristic of its discrete gesture commands. The ARC showed more improved results in the task completion time.

In Task 7 for instance, the remote control required ten commands (i.e., button presses), expressed as:

No. 7 Task[Remote] = Menu → Down → Select (or Right) → Down x3 → Select (or Right) → Right x2 → Select

For ARC Task 7 is combinations of 11 gesture commands. They are continuous hand motions of the following:

No. 7 Task[ARC] = Cue → Cue → Down → Cue → Down x3 → Cue → Right x2 → Cue

The ARC required more commands than the remote control. However, having the continuous hand motions such as [Down x3] or [Right x2] that consists of repetitive movements (or commands) it gives the user a feeling of making only one gesture command. In other word, this can be an advantage to the user because the 11 commands could be narrowed down to simply 8 commands.

In the WPGB case, in order to compare its performance with others, gestures for complex tasks (brightness and contrast control) was added with SPIN_LEFT and SPIN_RIGHT, respectively as a hot-key gesture. For task 7 for instance, it required four combinations of discrete gestures, as follows:

No.7 Task[WPGB] = SPIN_LEFT → RIGHTx2→SELECT

TABLE III
Comparisons of Number of Commands Used for Task and Average Time Spent per Command

Task No.	Remote	ARC	WPGB
1	2	3	2
2	2	3	2
3	4	5	4
4	8	9	8
5	4	5	4
6	8	9	8
7	10	11	4
8	9	10	4
9	15 or 20	21	8
10	19 or 24	25	12
11	17 or 22	23	10
12	17 or 22	23	10
Total commands	115 or 135	147	76
Ave Time of 1 Command	0.83299	1.025	3.489

Note) When using the remote control, task 9, 10, 11, 12 can be done in two paths; a longer and shorter path. Five participants operated the remote control in the shorter path in the actual experiment.

As listed in Table III, all twelve tasks required 115 (or 135), 147, and 76 for the remote control, ARC and WPGB, respectively. The average time required for one command was 0.833sec, 1.025sec and 3.489sec for the remote control, ARC, and WPGB. This result showed that the amount of time spent for the ARC users and the remote control users was close in time (in factor of 1.23). Whereas, the time required with the WPGB took 4.188 times more than that of the remote control.

V. DISCUSSION

We are currently using directional IrDA in order to provide the user with the capability of intuitive pointing toward devices. However, if multiple devices are placed closely and adjacently, there is a potential problem of selecting unwanted devices. The IrDA has a 30 degrees radiation angle in general and the radiation angle is adjustable by redesigning the refraction cone of IrDA. Still, the IrDA is an invisible light that changing the radiation angle gives tradeoffs between convenience and accuracy. To solve this problem, we can consider using other light mediums such as lasers, which are better in terms of straightness and visibility for user feedback.

In the experiment on Section 4 where the WPGB was compared, we assigned special gesture commands (hot-key gestures) to enter the brightness and contrast control mode. This had to be done in order to compare the WPGB in the same condition as the others. Without adding the gestures for brightness and contrast control, the

WPGB does not have the ability to control the brightness and contrast, thus it would be inappropriate to compare it with others. We could add more hotkey gestures for every individual function, for example: increase in brightness, decrease in brightness, increase in contrast, and decrease in contrast. However, as mentioned in Section 2, limitations exist on adding more gestures, while maintaining an acceptable recognition rate. This approach is disadvantageous for the user because as the number of gesture commands increases, the more difficult it becomes for the user to remember all the gesture commands.

The experiment was done using an ordinary TV, so a factory setting for the menu would be designed and optimized on the remote control. If the menu could be redesigned for an ARC that uses hand motions, the ARC's performance would be enhanced. For example, a TV menu has 1x4 or 1x3 menu sizes in different levels. However, if the menu has a 5x5 structure and minimum sub menu layers, then the menu would be able to show more functions simultaneously. This would improve the ARC's performance.

Using the ARC or remote control is more flexible than using the WPGB when mis-operations occur. This is because when a user makes a mistake on the menu control, the ARC or remote control user can easily go back to their intended menu elements, while on a WPGB the user can not. This is due to fact that the WPGB is a single discrete gesture command based device, which implies that using the ARC can be more versatile than using a WPGB.

VI. Conclusions and Future Work

We developed a wristwatch-type wearable remote control (ARC) that runs and interacts with a virtual menu through natural hand motions. XML is used to design the virtual menu. Through the experiment, we have evaluated task completion times and usability for each device. Considering the fact that the ARC provides an easy and unified control through natural hand motions, the experimental results show a potential for the use of ARC in the future.

Future researches will analyze more situations using age groups in mobile environments and improve the function of FingerTapButton. The FingerTapButton uses a piezo-electric microphone and provides an accessible and convenient CUE method. However, depending on the type of user and the conditions in which it is worn, it can give false-true (or vice versa) results which can lead into erroneous operations. The performance of the FingerTapButton directly affects the overall ARC performance. Thus we are currently doing extensive research to improve the FingerTapButton.

References

[1] I.-Y. Cho, J. Sunwoo, H.-T. Jeong, Y.-K. Son, H.-J. Ahn, D.-W. Lee, D.-W. Han, C.-H. Lee, "A Distributed Wearable System Based on Multimodal Fusion," *Proc. International Conference on Embedded Software and Systems*, 2007, pp. 369-378.

[2] D. Ashbrook, J. Clawson, K. Lyons, N. Patel, T. Startner, "Quickdraw: The Impact of Mobility and On-Body Placement on Device Access Time," *Proc. ACM SIGCHI conference on Human factors in computing systems,* 2008, pp. 219-222.

[3] T. Baudel and M. beaudouin-Lafon, "Charade: Remote Control of Objects Using Free-hand Gestures," *Communications of the ACM*, vol. 36, 1993, pp. 28-35.

[4] B. Rime and L. Schiaratura, "Gesutre and Speech," *Fundamentals of Nonverbal Behavior, Cambridge University Press,* 1999, pp. 239 – 281

[5] G. Kortuem, Z. Segall and M. Bauer, "Context-Aware, Adaptive Wearable Computers as Remote Interfaces to 'Intelligent' Environments," *Proc. IEEE International Symposium on Wearable Computers*, 2000, pp. 58-65.

[6] M. Gandy, T. Starner, J. Auxier, D. Ashbrook, "The Gesture Pendant: A Self-illuminating, Wearable, Infrared Computer Vision System for Home Automation Control and Medical Monitoring," *Proc. IEEE International Symposium on Wearable Computers*, 2000, pp. 87-94.

[7] K. Tsukada and M. Yasumura, "Ubi-Finger: Gesture input device for mobile use," *Proc. of 5th Asia Pacific Conference on Computer Human Interaction*, 2002, vol. 1, pp.388-400.

[8] J. Zaletelj, J. Perhavc, and J. F. Tasic, "Vision-based human-computer interface using hand gestures," *Proc. of Eight International Workshop on Image Analysis for Multimedia Interactive Services*, 2007

[9] K. Ouchi, N. Esaka, Y. Tamura, M. Hirahara, and M. Doi, "MagicWand: An Intuitive Gesture Remote Control for Home Appliances," *Proc. of AMT 2005*, pp.274.

[10] J. Rekimoto, "GestureWrist and GesturePad: Unobtrusive Wearble Interaction Devices", *Proc. IEEE International Symposium on Wearable Computers*, 2001, pp. 21-27.

[11] M. Fukumoto and Y. Tonomura, "Body Coupled FingerRing: Wireless Wearable Keyboard," *Proc. CHI*, 1997, pp. 147-154.

[12] A. Feldman, E. M. Tapia, S. Sadi, P. Maes and C. Schmandt, "ReachMedia: On-the-move Interaction with Everyday Objects," *Proc. IEEE International Symposium on Wearable Computers*, 2005, pp. 52-59.

[13] J. Kela, P. Korpipaa, J. Mantyjarvi, S. Kallio, G. Savino, L. Jozzo and S. D. Marca, "Accelerometer-based Gesture Control for a Design Environment," *Personal and Ubiquitous Computing*, Volume 10. Issue. 5. Springer Verlag, 2006, pp. 285-299.

[14] D. Ahlstrom, R. Alexandrowicz, M. Hits, "Improving Menu Interaction: a Comparison of Standard, Force Enhanced and Jumping Menus," *Proc. CHI*, 2006, pp. 1067-1075.

Dong-Woo Lee received his B.S. and M.S. degrees in Electronics Engineering from Kyungpook National University, Daegu, Korea, in 1995 and 1997 respectively. From 1997 to 2000, he worked in CNS Team for Hyundai Electronics in Kyungi-do, Korea, as a member of engineering staff, where he developed Car Navigation System and digital map. Since 2001, he is working in a wearable computing research team as a senior member of research staff at Electronics and Telecommunications Research Institute (ETRI), where he is developing wearable computer systems. He is interested in HCI for wearable computers and textile-based user interfaces.

Jeong-Mook Lim received his B.S. and M.S. degrees in Computer Science from Chungnam National University, in 1998 and 2000 respectively. From 2000 to 2001, he worked on Intelligent Transportation System Team for LG Information and Communications as a researching staff. From 2001, he is working at Electronics and Telecommunications Research Institute, where he developed operation and management system for several access networks including A-ON, E-PON, G-PON, Since 2007, he is interested in developing unconventional user interfaces for ubiquitous computing environments. Now he is developing user interface devices and methods for wearable computers.

John Sunwoo (M'99) received his B.S. and M.S. degrees in Electrical Engineering from Auburn University, in 2003 and 2005 respectively. Since 2005, he is working at Electronics and Telecommunications Research Institute, where he is developing wearable computing systems. He is interested in designing a wearable BAN communication controller for e-textiles.

Il-Yeon Cho received his B.S. and M.S. degrees in Industrial Engineering from Sungkyunkwan University, Seoul, Korea, in 1991 and 1993 respectively, and received his PhD degree in Computer Engineering from Chungnam National University, Daejeon, Korea, in 2007. Since 1993, he is working at Electronics and Telecommunications Research Institute (ETRI), as a senior research scientist. From 1995 to 1996, he was a visiting research scientist in OSF Research Institute, US and conducted a collaborative research. Currently he is a head of a wearable computing research team at ETRI. He is interested in developing wearable computers and embedded systems.

Cheol-Hoon Lee (F'97) received the BS degree in Electronics Engineering from Seoul National University, Seoul, Korea, in 1983, and the MS and PhD degrees in Electrical Engineering from the Korea Advanced Institute of Science and Technology, Seoul, Korea, in 1988 and 1992, respectively.

From 1983 to 1994, he worked for Samsung Electronics Company in Seoul, Korea, as a researcher. From 1994 to 1995 and from 2004 to 2005, he was with the University of Michigan, Ann Arbor, as a research scientist at the Real-Time Computing Laboratory. Since 1995, he has been a professor in the Department of Computer Engineering, Chungnam National University, Daejeon, Korea. His research interests include parallel processing, operating systems, real-time and fault-tolerant computing, and microprocessor design.