International Journal of Control and Automation
Vol.7, No.3 (2014), pp.113-124
http://dx.doi.org/10.14257/ijca.2014.7.3.12

A Kinect Sensor based Windows Control Interface

Sang-Hyuk Lee[1] and Seung-Hyun Oh[2*]

Department of Computer Science, Dongguk University, Gyeongju, South Korea

{nike2001[1], shoh[2]}@dongguk.ac.kr

Abstract

Many new concepts have recently been developed for human–computer interfaces. Xbox, introduced by Microsoft, is a game console. Kinect is a low-cost device that supports multiple speakers, a microphone, a red-green-blue camera, and a depth sensor camera for motion recognition. In the beginning, Kinect was developed for Xbox, but then supported various functions for the Windows operating system (OS). Similar to a Smart TV interface, in this study, we propose an intelligent interface for the Windows OS using Microsoft Kinect. Instead of using a mouse or keyboard, the interface helps humans interact with the computer using voice and hand gesture recognition. To interact with our system, users first register their voice commands in our recognition system, and then utilize those registered commands control the Windows OS. Our proposed system also provides scalability and flexibility and was assessed with a high level of recognition rate and user's satisfaction.

Keywords: *Context-aware, X-Box, Kinect Sensor, PC Control, Motion Recognition, Voice Recognition*

1. Introduction

There have been many human–computer interfaces developed based on human behavior, such as finger/arm movements, tongue movements, etc. Microsoft's Kinect is now widely used as a low-cost device because of its integrated variety of sensors. Kinect [1], shown in Figure 1, comes integrated with a red-green-blue camera, an infrared sensor, a four-array microphone, and a three-axis sensor.

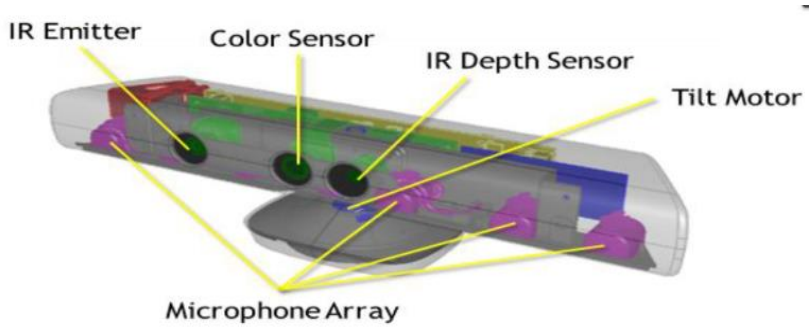

Figure 1. Kinect sensor component

* Corresponding author

ISSN: 2005-4297 IJCA

Microsoft (MS) continuously helps to develop more and more applications for Kinect by offering and supporting the Kinect software developer's kit (SDK). The Kinect SDK supports Color Stream output, Depth Stream output, and Skeleton Stream output for voice recognition and motion tracking. Version 1.7 of the Kinect SDK supports the depth sensor to recognize 3D objects. Kinect Fusion is also available in this version.

In this study, we propose an intelligent interface for the Windows operating system (OS) using Microsoft's Kinect. Instead of using a mouse or keyboard, the interface supports human interaction with a computer using voice and hand gesture recognition. To interact with our system, users stay a few meters in front of the Kinect device, register their voice command in our recognition system, and then use those registered commands to control the Window OS. Or, users can use our exist commands which are already registered by us in the recognition system. Our experiments include the evaluation of speech recognition, the result of recognition rates in the environment with different noise levels and distances, and the results of satisfaction of the program for users who participates in the experiments.

This paper is organized as follows: Section 2 introduces some of the research related to Kinect. Section 3 describes our proposed Kinect interface for controlling the Windows OS. Section 4 covers our implementation, and Section 5 discusses our experiments and results. Finally, conclusions are presented in Section 6.

2. Related Research

2.1 Kinect Sensor-based Motion and Gesture

Various interfaces have attempted to use the Kinect sensor as an input method for gesture, motion, and voice recognition. Some research used depth information and skeleton images to recognize human gestures and motion [2, 3]. Authors have also researched how to use the coordinates of both hands by tracking hand motion and hand gestures in order to implement an interface. Figure 2 shows skeleton coordinates provided by the Kinect depth sensor with the support of the Kinect SDK.

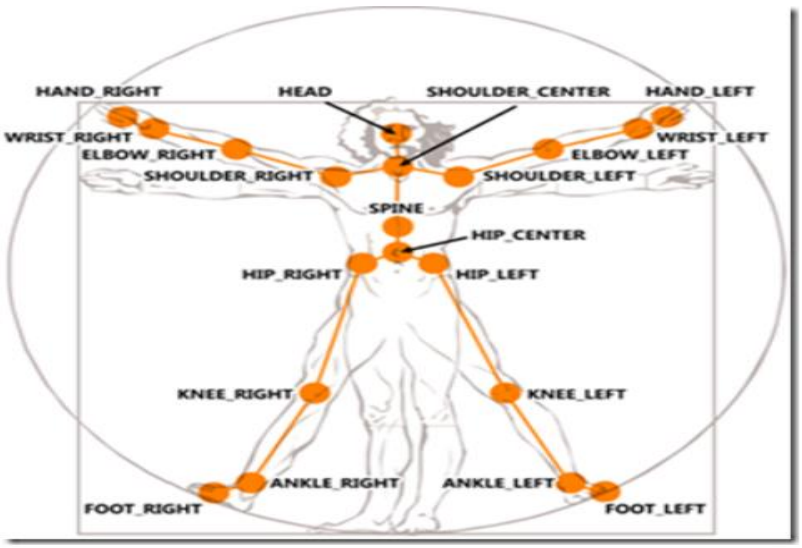

Figure 2. Skeleton joint elements

The Kinect interface uses default skeleton coordinates to implement several types of motion and gesture recognition, such as algorithmic detection, neural networks and some others [4]. An algorithm is used to analyze the gestures of a person and calculate the probability of the next action of that person using a neural network. In addition, gesture recognition can be detected via stored samples of people action. However, Kinect does not support hand recognition, motion recognition, or some kinds of interactions, such as touch, click, double click, dragging, and so on.

2.2 Kinect Sensor-based Voice Recognition

Our own previous research [5] introduced voice recognition as an interface for controlling a computer. That paper implemented a voice recognition system and a feature to support physically challenged users. In this paper, the program executes by handling simple types of input separately. Single words are recognized and commands are pre-defined. Only a developer can add new commands. Users must learn the instructions and commands before start to use the application. The computer will only be controlled by registered commands. In order to implement a voice recognition system, we built speech grammar recognition in the constructor, using desired predefined syntax and words. Figure 3 shows our grammar constructor syntax. In the source code, the developer uses an If-Then structure to control the computer by issuing a defined voice command.

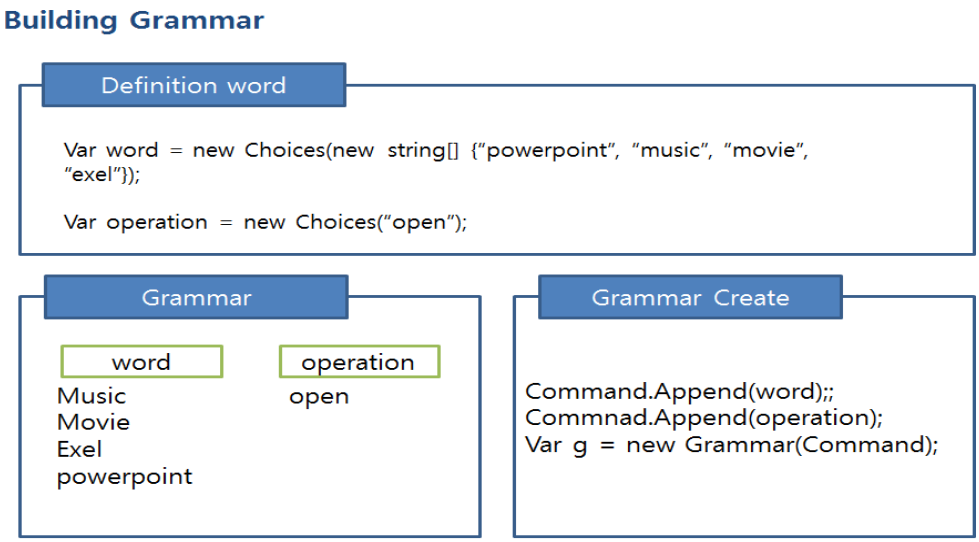

Figure 3. Grammar constructor syntax

2.3 Smart TV in the Living Room Environment

Some researchers proposed a Smart TV system based on the integration of a speech recognition interface (voice recognition), QR (Quick Response) code, and motion recognition (motion detection) technology [6, 7, 8]. Their implementation of voice recognition used Google's open-source library application programming interface and text-to-speech function. The principle behind Google's speech recognition is for people to use their smart phones or other devices as speech input devices. Then, the data is sent to Google servers for recognition. The Google servers have a voice analysis system for most common types of data. This analyzes the received voice data and sends back a list of results. This speech recognition is

used to control the Smart TV, and the information of each device was implemented to be able to check and control devices in the living room.

The Smart TV was implemented to control home appliances through detection of user movements. It implements an interface to control this process using a motion detection sensor. There are some limitations when using only an infrared sensor as input communication to develop remote control. To implement interface for motion and button control, they used the Wii remote controller, which provides a common connection among motion recognition sensors, infrared sensors, and Bluetooth communications capabilities. The advantage of the motion interface is not limited to one kind of event.

Similar to a Smart TV interface without a mouse and keyboard, we propose a Windows interface to control the living room using Kinect's motion, gesture, and voice recognition. This interface supports a wide range of processing to handle events registered by the user. This feature can also support a Smart TV, especially for those who are physically challenged.

3. Kinect Sensor-based Windows Control Interface

In this section, we briefly describe our proposed user interface for the Windows graphical user interface (GUI) based on Kinect motion, gesture, and speech recognition. The configuration of the whole interface is shown in Figure 4. In the Windows GUI, people use the mouse to select one of the menus on the screen to run a program or function. Studying how a Smart TV works, we developed a Kinect-based motion, gesture, and voice recognition interface while running the Windows GUI in the background in order to control the windowing system with hands and voice instead of mouse or keyboard.

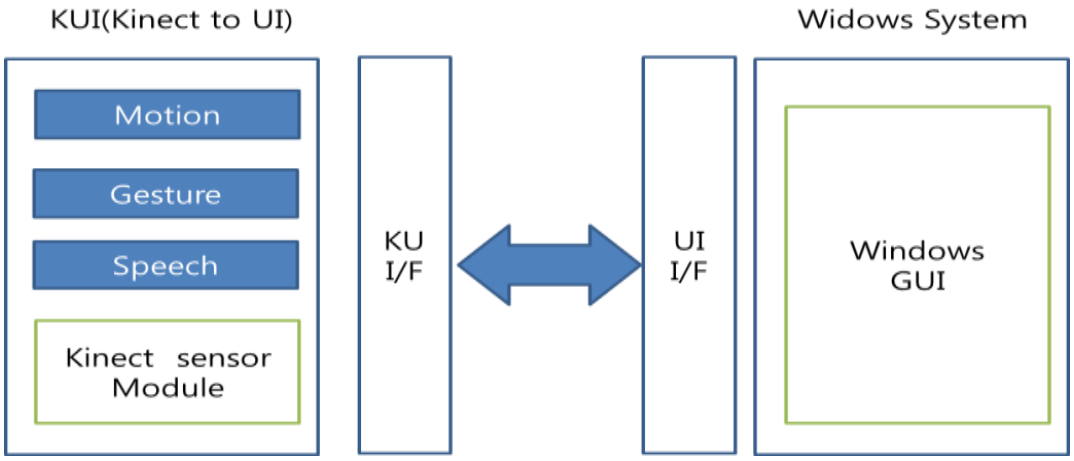

Figure 4. Kinect sensor-based Windows control interface

However, it is difficult to control computers using only voice recognition, for example, select a file, browse the Internet, or switch between two applications. Hence, we propose a more user-friendly interface in which user can not only control the Window OS by voice but also by motion and gestures. Our motion and gesture interfaces are shown in Figure 5.

Kinect detects motions via the user's joints (or skeleton). From the retrieved information, the hand information is detected and extracted to the coordinates of the screen. The hands' coordinates will be converted to the cursor's coordinates of the Windows GUI to control the mouse cursor. The Kinect SDK supports detection of a "grab" interaction, which corresponds to a double click of the mouse in the Windows GUI. The use of motion and gestures will

enable manipulation of the mouse in the Windows GUI such as mouse moving, selection function, and execution of program.

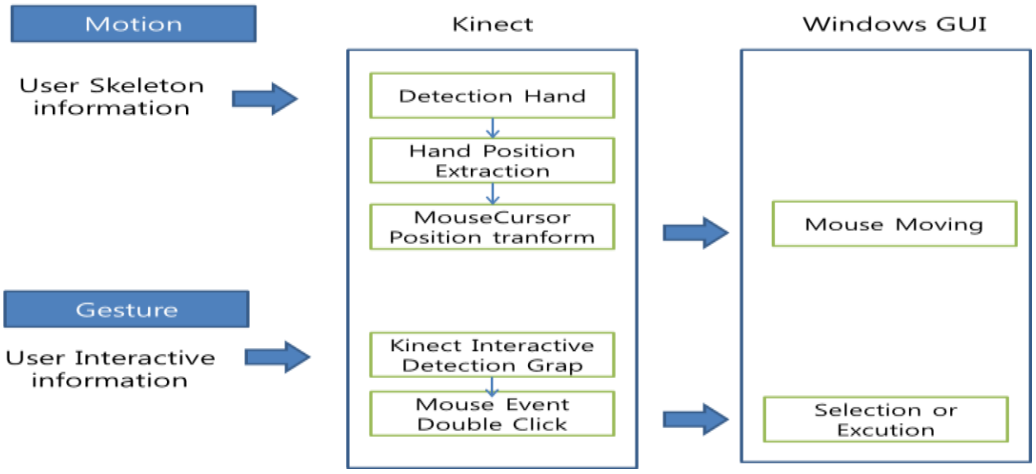

Figure 5. Motion and gesture interfaces

Besides, we also improve the speech re cognition interface in which users can register and use their own command to control the Winwod OS. Hence, the Kinect voice recognition feature provides three modes: command mode, input mode, and user register mode. Figure 6 shows the speech recognition interface. Left side shows our voice register recognition system with using recorded information from Kinect device. Right side shows the corresponding functions in the Window system.

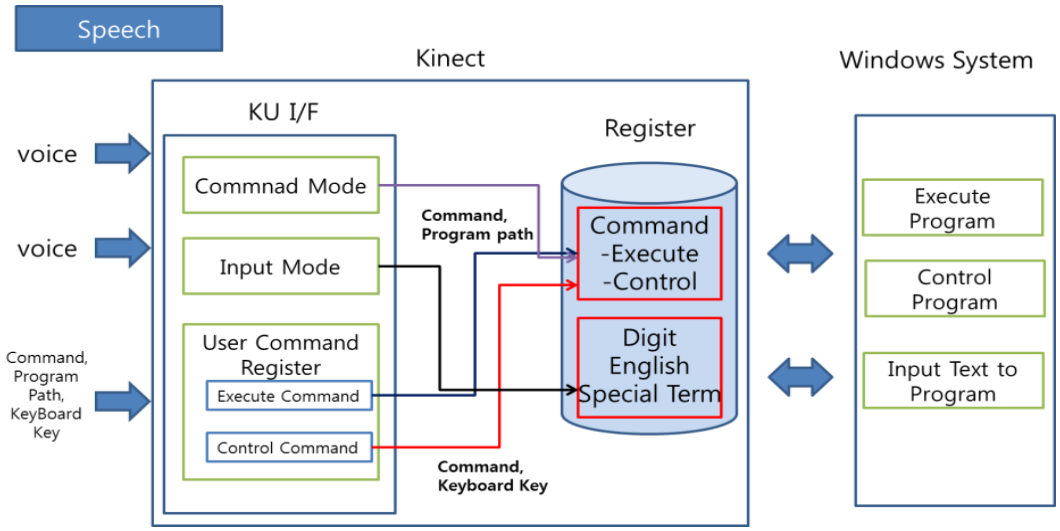

Figure 6. Speech recognition interface

All commands are stored in the “Register”. “Command” mode and the “User Command Register” mode are designated for execution or control in the “Register”. “Input” mode is designated for “Digits, English words, or special characters” in the “Register”. Composing an

email or working with MS Word are examples of input mode. All voice recognition is converted to text and stored as a number, words, and special characters. The ability to run or control a program belongs to the command mode. We implemented some basic commands, such as open the video player, edit a document (copy, paste, save, close, *etc.*), and browse the Internet. Our interface automatically switches between command mode and input mode according to the currently running program. For example, it allows users to enter an Internet address while the Internet browser is running. Or users can use their voice to switch between two modes.

The "User Command Register" mode provides a feature for users to define their own needed commands through the use of voice recognition. This feature is divided into two parts (execute and control) corresponding to the Windows system. The user can run a program with the execute command as registered. Following the instructions of the "User Command Register" mode, users register their own "Execute Command" by inputting the command's name/word and the program's path, then saving it to the register using the command's name program's path. Details will be explained in Section 4.1). If the registration succeeds, the user can execute the program by saying the command's name the command to execute corresponds to the execute function of the Windows OS.

Users can also register a "Control Command" to replace existing functions in a program that are usually invoked with a shortcut key, for example, Save (Ctrl+S), Copy (Ctrl+C), or Help (F1). To register such a command, users input the control command name/word intended to replace the corresponding shortcut key, and then save it to the register with the command's name shortcut key. Details will again be explained in Section 4.1). If the registration succeeds, the user can execute the program by saying the command's name. The control method will be invoked according to the matchup between the command and the shortcut key.

4. Implementation

In order to implement the new user interface, Kinect SDK version 1.7 is used with Visual Studio 2010 C # programming. The developed interface was tested under a Windows 7 32-bit operating system. A Korean speech language pack was used to ensure that the programs run in the background. The details of our implement process are discussed in subsection 4.1 – main screen, 4.2 – execution command register development process, and 4.3 – control command register development process.

4.1 Window control interface implementation

Figure 7 shows the screenshot of our implementation for the Window control program. The screen of the program consists of the features listed in Table 1

In Figure 7, label (1) displays the recognized command that the user have said; label (2) displays the recognized word in input mode; label (3) shows the user is in command mode or input mode.

From (4) to (6) show the control command register. When the command name is being displayed in (6), users can click a button (4) to delete the command from the register. Or, the

user types a new command name in textbox (6), and then clicks a button (5) to add it to the register system.

From (7) to (10) show the execute command register. The user types the command wants to execute in textbox (7), clicks button (8) to browse to the program's path, and then clicks button (10) to add it to the register system. Or, users can click a button (9) to delete an existing executes command from the register system.

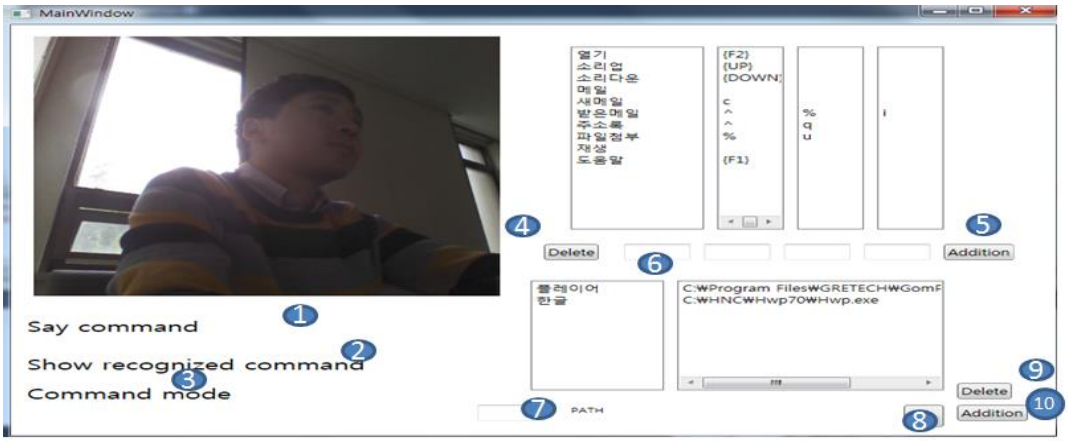

Figure 7. Execution screen

Table 1. Interface features

#	Description	#	Description
1	Display recognized Command	6	Display input control command
2	Display input command	7	Display input execute command
3	Display Mode (Input mode, Command mode)	8	Execute command, and search program's path
4	Delete registered control commands button	9	Delete execute command button
5	Add new control command button	10	Delete execute command button

4.2 Execution Command Register

Figure 8 shows the screenshot of our program and the program is being executed by the voice command register. In the previous voice command interface [5], developers must define all commands and corresponding executable programs by providing a command voice and executing a program's path in the source code. However, this command register interface lets users to add new commands to execute the program wanted. Later, users can use their own command to execute the program. In this example, a user added an execute command "Gom tivi" to our voice recognition system, and called the program by saying the command name.

Figure 8. Execution command register screen

4.3 Control Command Register

Figure 9 shows the screenshot of the Hangul program being run. Now, our command register interface helps to register a new command via voice recognition. For almost all common voice recognition programs, the corresponding commands of hotkeys are manipulated in the source code and by the developer. However, our command register interface allows users to insert more commands for a hotkey. By using this interface, users can extend the operations of the executed program. For example, in this experiment, a user called execution command named "Hangul" to execute the Hangul editor program and called register command named "Help" to open the help menu. The "Help" command had been registered in the voice recognition system before by that user.

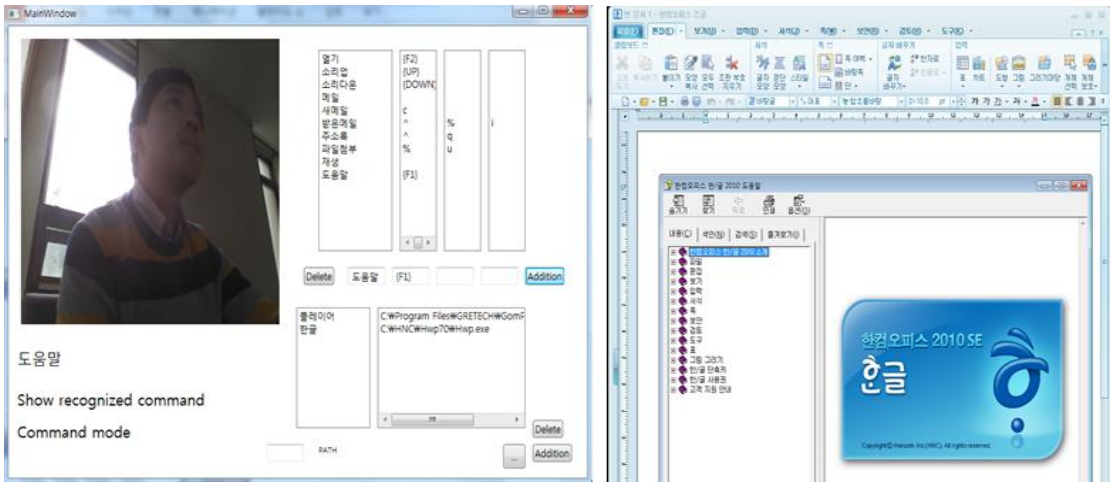

Figure 9. Control command register screen

5. Experiments and Results

5.1 Experiment Environment

Figure 10 shows the environment we used to test our system.

Figure 10. Experiment environment

This room resembles a living room. We conducted measurements with two groups of men and women, each group had two people. They were required to say ten different words five times. The recognition rate was measured and evaluated based on noise and distance between the speaker and the Kinect device. Then, we increased the number of participants for each group to determine satisfaction with our program, with rankings of very good, good, medium, poor, and very poor.

5.2 Measurement Methods: Speech Recognition

In this paper, evaluation of speech recognition followed standard measurement [9] for voice recognition systems are created to test and assess actual service environments. The test method is divided into two aspects using performance metrics and testing.

There are three defined rates identified to measure system performance: recognition rate, rejection rate, and recognition error rate. The recognition rate attempts to recognize speech and output the percentage of recognized speech that matches the spoken commands. The rejection rate and recognition error rate attempt to recognize whole speech (input speech) then compare it to a threshold. The speech is not recognized if its rejection rate is less than, or equal to, the threshold. If the recognition error rate is greater than the threshold, the speech is recognized but is deemed in error. The recognized result is displayed if the speech meets the recognition rate (a speaker's voice is recognized, the word is displayed on the screen). Other output recognition results are displayed with the number of recognition errors over the total number of times speech was supposed to be recognized.

The performance of the entire system was evaluated based on the types of speaker, direction and distance of the speakers, and the environment's noise. Direction and distance performance was evaluated by measuring the distance between a speaker and the Kinect device and exactly where the speaker stood (*i.e.*, in which direction) within the identified range of the Kinect device. Speaker performance was evaluated based on recognition of different types of speaker, such as man, woman, young person, and older person. Noise was evaluated by running the system in environments with different levels of noise

In this paper, we propose an interface that allows people to control the Windows operating system with the help of Kinect motion and speech recognition replacing keyboard and mouse commands. In addition, we provide a user registration system to extent the range of the voice recognition system depending on user selection. We propose a voice recognition system that

verifies the actual direction of, and distance between, the speaker and the Kinect device and the noise of the environment. Also, the recognition system was tested with groups of women and men who were each required to say ten different words five times. Then, each participant was required to examine the system and evaluate it for satisfaction.

5.3 Distance and Noise Recognition Rate

Figure 11 shows the result of recognition rates in the environments with different noise levels and distances. In Figure 11, the blue line indicates a distance of 1 meter between speaker and Kinect device, the red line indicates a distance of 2 meters, and the green line indicates a distance of 3 meters. The results show that recognition rates are not much different when they come to distance. However, they are different when they come to noise. The louder the noise is, the lower the recognition rate is recorded. A quiet environment is about 40dB. An environment playing hard music has a noise level of about 80dB. The graph shows that the recognition rate falls rapidly when levels change from 40dB to 60dB, to 70dB, to 80dB. However, 80dB of noise is not a realistic living room. Hence, it is not applicable to our voice recognition system.

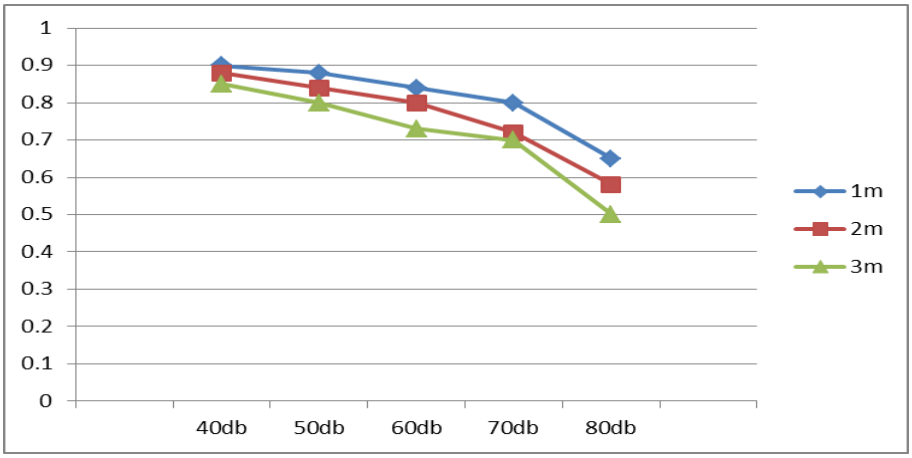

Figure 11. Results of recognition rate

5.4 Program Satisfaction

We also did a survey on how users satisfy our recognition system. The survey includes the user's satisfaction on the system interface, the recognition feature, the register mode, the input mode, and the average. Program satisfaction was evaluated by users who participated in the lab environment experiment. Depending on the features of each participant, satisfaction ranged from 100 percent (considered very good), 80 percent (considered good), 50 percent (medium), 30 percent (poor), and 10 percent (very poor). In Figure 12, the satisfaction of our system is around 78.9 percent, earning a rating of good.

Satisfaction with our user register mode was also good. But input mode showed a slow response. The evaluations from participants show they were satisfied with the interface recognition rate and register command mode, but not the input mode.

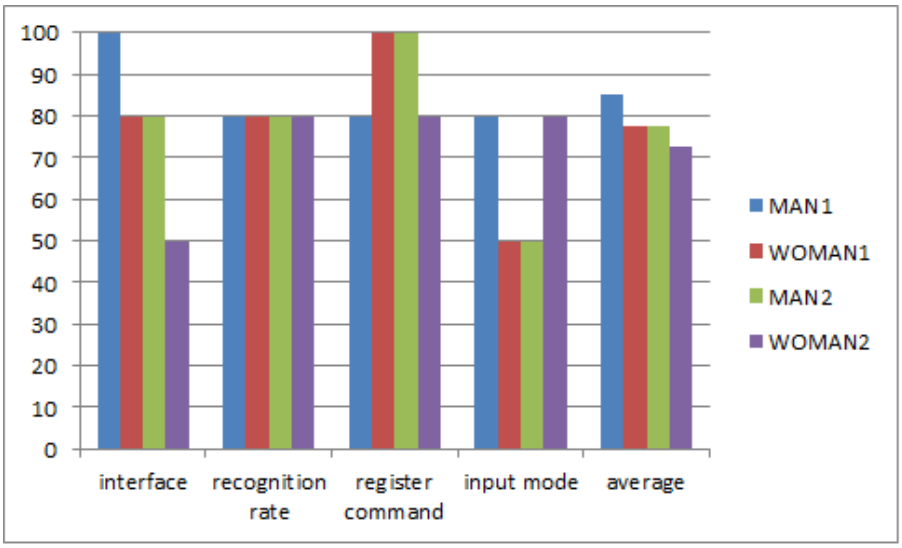

Figure 12. Program satisfaction

6. Conclusions

This study used the Microsoft Kinect sensor with motion and voice recognition to develop an interface to control the Windows operating system. The combination of motion and voice not only replaced the mouse using but also provided an interface for users to add more operations/commands by using a registration method. So, users can control their computer easily depend on their own needs. With Kinect, possible interfaces to control the Window system without using a mouse and keyboard are unlimited.

We also have evaluated our speech recognition rates in the environment with different noise levels and distances, and the results of satisfaction of the program for users who participates in the experiments. To verify our proposed interface, distance and noise were also measured for satisfaction with recognition rates. As a result, a 3 meters distance and 70dB of noise allowed a 70 percent recognition rate. Satisfaction proved that it can be acceptable. However, input mode had a slow response and needs to be further improved.

References

[1] Kinect for Window, http://www.microsoft.com/en-us/kinectforwindows/.
[2] T. Arici, "Introduction to programming with Kinect: Understanding hand / arm / head motion and spoken commands", Signal Processing and Communications Applications Conference (SIU), **(2012),** pp. 18-20.
[3] V. Tam and L. -S. Li, "Integrating the Kinect camera, gesture recognition and mobile devices for interactive discussion", Teaching, Assessment and Learning for Engineering (TALE), IEEE International Conference, **(2012)**, pp. H4C-11-H4C-13.
[4] J. Webb and J. Ashley, "Beginning Kinect Programming with the Microsoft Kinect SDK", Published by Apress**, (2012).**
[5] L. S. Hyuk and O. S. Hyun, "Kinect Sensor based PC Control Interface for Handicapped Users", International Conference on Convergence and its Application (ICCA 2013), vol. 24, (**2013**), pp. 49-52.
[6] S. Choi and T. Kim, "Personalized user interface design for smart TV", Proceedings of Korea Computer Congress 2011 (KCC 2012), vol. 39, no. 1(D), (**2012**), pp. 106-108**.**
[7] Y. -C. Moon, S. -J. Kim, J. Kim and Y. -W. Ko, "Design and Implementation Integrated Interface for Smart TV in Home Automation", Journal of Advanced Information Technology and Convergence, vol. 11, no. 2, (**2013**), pp. 87-94.

[8] M. Omologo, "A prototype of distant-talking interface for control of interactive TV", Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference, (**2010**), pp. 1711-1715.
[9] Korea Telecommunications Technology Associations, "Performance Test method of Speech Recognition Service for Robot", **(2009).**

Authors

Sang-Hyuk Lee

He received the Bachelor of Computer Science from Dongguk University, Gyeongju Campus, South Korea in 2012 and 2014, respectively. His research interests are in underwater wireless sensor routing protocols. He has joined a Kinect project from university during his Master course.

Seung-Hyun Oh

He received the B.S. degree in computer science from Dongguk University, Seoul, Korea, in 1988, and the M.S. and Ph.D. degrees in computer engineering from Dongguk University, Seoul, Korea, in 1998 and 2001, respectively.

He has been a Professor with the Department of Computer Engineering, Dongguk University Gyeongju Campus since 2002. Since 2011, he is serving as Chairperson of the Department of Computer Engineering, Dongguk University Gyeongju Campus. His current research interests include wireless communications and sensor networking system.

