A Non-visual Interface for Tasks Requiring Rapid Recognition and Response

An RC Helicopter Control System for Blind People

Kazunori Minatani[1] and Tetsuya Watanabe[2]

[1] National Center for University Entrance Examinations, Tokyo, Japan
minatani@rd.dnc.ac.jp
[2] University of Niigata, Faculty of Engineering, Niigata, Japan
t2.nabe@eng.niigata-u.ac.jp

Abstract. To implement a user interface for blind people, auditory and tactile outputs have mainly been used. However, an auditory interface is ineffective for tasks that require the rapid recognition that vision enables. Thus, this paper presents a method to achieve rapid recognition with a non-visual user interface. This user interface is implemented to achieve a prototype fully controllable system of an RC helicopter for blind people by using a braille display as a tactile output device. This paper also explains the system integration software, named brl-drone, and hardware components of that system including the AR. Drone. The AR. Drone is a controlled helicopter that uses an auxiliary magnetic sensor and a game controller to solve the problems that arise when a braille display is used as a tactile indicating device.

Keywords: Blind People, Non-visual Interface, RC Helicopter, Braille Display, AR. Drone

1 Background and Objective

To implement a user interface for blind people, auditory and tactile outputs have mainly been used. Information can be obtained faster by listening to speech (auditory recognition) than by reading braille (tactile recognition). Nevertheless, intrinsically, speech is constructed from a time series of phonemes, so it requires certain definite time.

This fact does not totally change even if blind people can understand something said at a faster speech rate than sighted people can [1]. Thus, an auditory interface is ineffective for tasks that require the rapid recognition that vision enables. We tried to solve this problem by using a braille display as a tactile output device.

Our primary purpose is to achieve a fully controllable system of an RC helicopter for blind people. An event called the Jump to Science Summer Camp 2010 [2] was held to increase blind students' interest in science. We hosted a workshop in which they tried to control an RC helicopter. Blind students who joined that workshop enjoyed this attempt to learn how to fly helicopters. Such attempts may also be signifi-

cant if the report of a sound localization deficit in early-blind people [3] is taken into account. The students seemed to be very satisfied with the workshop.

That workshop was a purely educational activity, not a kind of experiment in a controlled environment. However, the author recognized problems that must be solved by developing a proper user interface. The workshop's attempts had a critical limitation. Even without sight, a person can detect the position of a flying RC helicopter relative to himself/herself by the sound generated by its rotors. The position of the flying RC helicopter can be known, but the direction because an RC helicopter makes the same sound no matter which direction it faces. Generally, an RC helicopter's control systems are designed to command horizontal moves in the relative direction of the fuselage's head (i.e. turn left or right, move forward in the current direction, etc.). Thus without sight, horizontal moves are impossible to command. The control is limited to only vertical moves (go up or down).

We try to overcome this limitation by indicating the direction of a helicopter's head on a braille display. In our view shaped by the experience of the summer camp, the direction of a helicopter must be recognized immediately, so using speech (an information media constructed from a time series of phonemes) requires too much time. Furthermore, one's auditory sense should be allowed to concentrate on the sound generated from a helicopter's rotors.

Another problem with controlling an RC helicopter arises when a braille display is used as a tactile indicating device. If information that must be recognized rapidly appears on a braille display, the user must put his/her fingers on it. RC helicopters, which are sold as toys, are usually designed to be controlled by a pair of joysticks. This user interface is designed on the assumption that both hands will grasp the joysticks. This is incompatible with our interface using a braille display. For our interface, the commands generated by at least one joystick must be input another way. Therefore, we adopted a method to use the user's own head as a joystick-like device.

2 Development Platform

Fig. 1 illustrates the system's components and the relationship between them.

2.1 Controlled Helicopter and its Attached Additional Sensor

We choose the AR. Drone from the Parrot [4] as the target. Generally, RC helicopters have no programmability, so they are unfit for our development. On the other hand, the AR. Drone has sufficient potential. It was originally designed to be controlled with iOS devices (iPhone, iPad, and iPod touch) through a Wi-Fi connection. Users can not only send commands to an AR. Drone but also receive data of sensors and images captured by cameras on the AR. Drone. Commands and data are exchanged as UDP packets on a Wi-Fi connection established between iOS devices and the AR. Drone. This means devices with a TCP-IP stack, such as all modern PC operating systems, can play the role of a third-party control system of an AR. Drone.

The AR. Drone has an accelerometer and a gyro, the data from which enables pitch, roll, and yaw to be calculated. However, the gyro used as an MEMS sensor tends to make certain errors that lead to miscalculation of the yaw value. For our purpose, the direction of an AR. Drone's head must be calculated precisely. For this calculation, the yaw value is critical data.

Thus, this error is not tolerable. To avoid undesirable characteristics of the gyro sensor and acquire more accurate information on the direction of an AR. Drone's head, we decided to attach a magnetic sensor to the AR. Drone to obtain richer information from it during its fly. As a magnetic sensor we adopted the WAA-010 9 axis sensor [5].

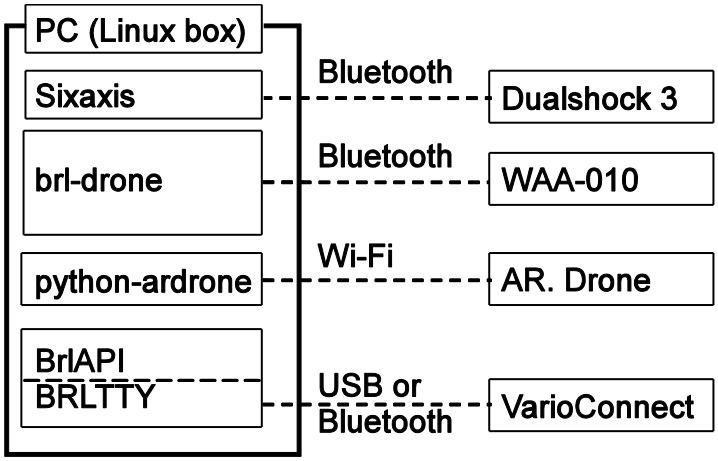

Fig. 1. System configuration.

2.2 Controlling Platform

Base platform and peripherals: Linux Operating System (Debian 6.0 Squeeze) on a PC acts as the core of controlling platform. The PC must have Wi-Fi, Bluetooth, and USB interfaces for connecting to the AR. Drone, WAA-010, and peripherals (explained below).

As already mentioned, a braille display is required. This time, we choose the VarioConnect 40 braille display from BAUM [6]. Reasons for this choice are:

1. It has one joystick-like device named Navistick on the front center of the braille display that can be used as an input device to command an AR. Drone.
2. BAUM's communication protocol between a PC and a braille display allows more than two cursor routing keys to be sent simultaneously [7]. This feature may make our system's user interface more user-friendly.

A head-tracking sensor is needed. As explained above, we planned to use the user's own head as the second joystick required to control an AR. Drone so that the position of the head can be tracked. We decided to use the PlayStation 3's game controller,

named Dualshock 3 [8], from Sony Computer Entertainment as the sensor. Fortunately, the base shape of that controller can be placed on a cap.

Thanks to the Dualshock 3's accelerometer, if the user wears a cap attached to the Dualshock 3 controller, by how much and in what direction the user should tilt his/her head can be detected. This controller supports connection to the PC through not only USB but also Bluetooth, so there are no wires to obstruct the user's body and head movement.

System integration software: Elements explained above are integrated by the appli-cation software running on a Linux PC. This software is written in Python scripting language. It is tentatively named brl-drone.

Brl-drone uses some open source software. It uses python-ardrone [9] for sending commands to and receiving data from an AR. Drone. To control a braille display, Brl-drone uses the Python binding of BrlAPI [10], which is a part of the BRLTTY [11]. Here, we focus on using the VarioConnect as a braille display, but thanks to the BrlAPI, it is possible to support braille displays that can be used with the BRLTTY with the least amount of effort.

The software that receives the Dualshock 3's sensor data is implemented in an in-dependent process. Sixaxis [12] is a user space daemon that supports Bluetooth con-nection between Sixaxis, which is early version of the PlayStation 3 game controller, and Linux. On the basis of Sixaxis's source code, we developed a user space daemon to read the Dualshock 3's sensor data through Bluetooth. Using UNIX's named pipe, this daemon sends Dualshock 3 sensors' values to brl-drone.

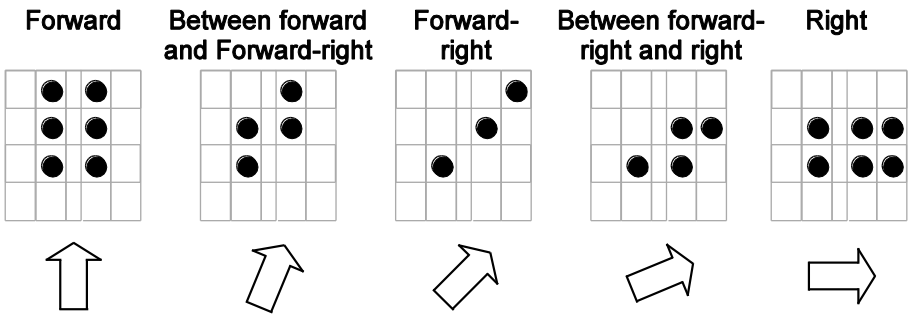

Fig. 2. Tactile symbols showing the direction.

User interface: Apart from the user's head with Dualshock 3, interaction between the user and the system is put all together on a braille display.

Information about an AR. Drone's current state is indicated using braille. Such in-formation includes the battery usage, the altitude, the pitch-rolls of an AR. Drone and user's head, and the direction of an AR. Drone's head. The battery usage (percentile) and altitude (centimeter) are displayed as digits, while the others are represented as icons that occupy two cells each.

The symbol of the direction of an AR. Drone's head, which we judged the most im-portant value, is put at the center of a braille display. On a braille cell array, that place is nearest to the Navistick, which may be often manipulated by the user. The direction

of an AR. Drone's head is represented as symbols such as those shown in Fig. 2. 360 degrees are divided into 16 directions and each direction is mapped to a unique symbol.

All controlling commands to an AR. Drone can be input with buttons on the braille display. Such commands include takeoff, move forward/backward/left/right/down/up, turn left/right, and change speed.

For reasons of practicality, move forward/backward/left/right are also assigned to commands generated by the user's head. Like Linux console's PC cursor, the Vario-Connect's Navistick can generate only one direction key code at a time. During a flight, it is expected to move in a medium direction (i.e. forward-left, backward-right). By calculating Dualshock 3's sensor value, the user's head can command the AR. Drone to move in any arbitrary direction.

3 Conclusions and Future Works

Brl-drone is guaranteed to show information about a controlled AR. Drone in 0.25 seconds. We consider this rapidity to be not only fit for tactile recognition but also sufficient to control RC helicopters. Preliminary demonstration videos are uploaded to a video sharing site [13]. Actual demonstration flights will be performed at the ICCHP conference venue if circumstances permit.

Carrying out subjective experiments can hopefully produce good results. However, this evaluation has particularly troublesome characteristics because controlling an RC helicopter is itself a hard task even for sighted persons. Events like the Jump to Science Summer Camp 2010 mentioned above should be utilized for evaluation.

In January 2012, the Parrot announced to release the AR. Drone 2.0 which will be a successor of the AR. Drone. The AR. Drone 2.0 will have own magnetic sensor, and it may be accessible through an SDK. If such an SDK will be released, our system can be simplified.

References

1. Asakawa, C. et al: Maximum Listening Speeds for the Blind. In: Proc. Conf. of Int. Community for Auditory Display 2003, pp.276-279 (2003)
2. Jump to Science, http://www.jump2science.org (in Japanese).
3. Zwiers, M. P. , Van Opstal, A. J., Cruysberg, J. R. M.: A Spatial Hearing Deficit in Early-Blind Humans. Journal of Neuroscience, 21, pp.1-5 (2001)
4. AR.Drone.com, http://ardrone.parrot.com/
5. Wireless Technologies Inc., http://www.wireless-t.jp/support.html (in Japanese)
6. BAUM Retec AG, http://www.baum.de/cms/en/braille/
7. BAUM Retec AG, http://www.openbraille.org/documents/VarioConnect-comm-prot-public-V6.pdf
8. DUALSHOCK3 wireless controller, http://uk.playstation.com/ps3/peripherals/detail/item113530/DUALSHOCK%C2%AE3-wireless-controller/

9. venthur / python-ardrone GitHub, http://github.com/venthur/python-ardrone
10. BrlAPI: let applications write braille !, http://brl.thefreecat.org/
11. BRLTTY, http://mielke.cc/brltty/
12. PdaXrom embedded, http://www.pdaxrom.org/index.php/PS3_sixaxis_bluetooth_uinput_driver
13. http://www.youtube.com/watch?v=C30QafU88x0, http://www.youtube.com/watch?v=oRQUtlRgJUU (in Japanese)