A glove-based gesture interface for wearable computing applications

Holger Kenn[1] Friedrich Van Megen[2] and Robert Sugar[2]

[1] TZI
Universität Bremen
kenn@tzi.de,
http://www.wearable-computing.de/
[2] Microsoft EMIC
friedrich.vanmegen@microsoft.com,
http://www.microsoft.com/

Abstract. This paper describes applications of a gesture-based user interface device and its integration into application software. It describes the glove input device used, the gesture recognition software and the integration of the device into application software with the help of a context framework. The system has been used in a number of demonstration applications ranging from desktop applications to the control of a mobile robot. We present preliminary results from user feedback given at a public demonstration of the system.

1 Introduction

User interface research in wearable computing distinguishes itself from other areas of human computer interface research by considering the interaction with the computer to be secondary, i.e., that the user of a wearable computer is primarily concerned with a task in the physical world surrounding him and that the wearable computer is supporting the user in his primary task. In desktop computing applications however, the primary task of the user is considered to be executed by interaction with the computer, so the focus of attention for the user lies on the human computer interface. For wearable user interfaces, this has several consequences including the fact that WIMP (Windows Icons Menus Pointer) interfaces often cannot be used although the typical user is highly familiar with such user interfaces from previous encounters with the ubiquitous paradigm of desktop computing. However, familiarity with an user interface is an important factor toward user acceptance, so one of the goals of wearable computing is to develop novel user interface mechanisms that, inspite of their novelty still give the user an instant familiarity with the user interface concept much like the "'Desktop"' metaphor and the "'point and click"' mouse interface exploits familiar concepts of a physical desk in desktop computing.

In wearable computing, the design of user interface devices is still an important area of research. Input methods such as chording keyboards [1], voice input, direct manipulation [2] and tracking of hand motion [3] have been successfully implemented and used in wearable applications. Seen from an application perspective, two important properties of an input method for wearable computing is their impact on the primary

task of the user and the social acceptance of their application. Ff the primary task of the user involves voice communication, a user interface using voice commands impedes the primary task. In a quiet environment such as a library, the use of a voice command interface might have low impact on the primary task but might be socially unacceptable. The negligence of these factors in the design of applications typically leads to low end-user acceptance [4].

Using keyboard-like input devices is often socially acceptable as their use is typically quiet and they can be build in a small form factor and integrated into the normal clothing of the user[5]. However, they often have a significant impact on manual primary task as they impede the use of the users hands. A drastic example is the use of portable wireless keyboards[6] that are typically held in one hand and typed on with the other hand which leaves no option to even hold another object while using the keyboard. In tasks such as aircraft maintenance [7, 8], the use of such an interface may be acceptable as the input of data typically is performed after the actual maintenance task to document changes, but in the general case, the interaction with the wearable computer should not limit the use of the users hands for other purposes.

Glove-based user interface devices have been used in the past for the interaction with wearable computers. These have the advantage that in many industrial applications, the users are required to wear gloves, so the impact on the primary task in non-interaction situations is small. In the implementation described by Boronowsky[2], the user interface uses a direct manipulation technique for menu selection. This has the drawback that when interacting, the user has to observe the head-mounted display to see what menu entry is currently highlighted and, by turning his hand, select another one or confirm the selected entry. This has an effect similar to the use of a pointer-based WIMP interface: it becomes unusable when the "'feedback loop"' between the user input device, the screen, the eye of the user and the hand of the user is interrupted. It also has an impact on the learnability of the user interface, the user cannot learn a sequence of actions he can execute "'without looking"' like it is possible with sequences of key presses on a keyboard. This latter feature may not be important for casual users but is a significant limitation for professional users as it limits the input speed gain achievable by user training.

Many gesture-based user interfaces have been studied, designed and implemented before. Examples for gesture input devices in the context of wearable computing are are the GestureWrist [9] and FreeDigiter [10], but similar input devices and systems have been designed for many other applications, ranging from design environments[11] to intelligent spaces[12].

In this paper, we describe an implementation of a glove-based gesture interface that represents a first step toward a user interface with learnable gesture sequences that can be executed without direct feedback through an output device. The device can be used not only for wearable applications, we demonstrate the use to control desktop applications and physical real-world artifacts. We present the architecture of the system and a preliminary qualitative evaluation based on user experience.

2 System Description

The gesture recognition system consists of a wireless input device integrated into a glove. The device is connected to the computer of the user, where a software system performs signal analysis and gesture identification. The output of this process is made available to applications via the context interface. Upon the recognition of a gesture, the software system generates events that can be routed to legacy applications and to wearable user interfaces. An overview of the system is given in Figure 1.

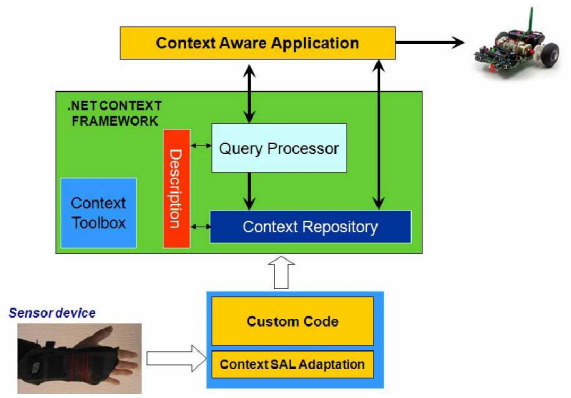

Fig. 1. System diagram

2.1 Input Device Hardware

The input device hardware consists of a textile glove with integrated electronics. Design constraints for the device were wireless operation for an extended period of time and minimal impact on the user's hand manipulation capability. The textile glove has been designed in an iterative process in which several prototypes were manufactured and tested with users for fit and ease of use. A description of the design process can be found in [13, 14]. In order to fulfill the design constraints, a small, lightweight and energy-efficient sensor platform has been developed, the so-called SCIPIO platform [15]. Together with its power supply and additional sensors, it forms the onboard electronics of the glove. For gesture recognition, the glove has been equipped with a 3d acceleration sensor device [16] that is used to measure both acceleration and pose in relation to the vector of gravity.

2.2 Signal analysis and gesture identification

For our experiments, we have defined a number of simple gestures that are both easy to detect with simple sensors and are easy to learn from a user perspective. We chose simple gestures that are based on the inclination of the hand in pitch, i.e., fingers pointing upwards or downwards, and roll, i.e., hand turned left or right.

Fig. 2. The textile glove

The gesture detection system used is based on the fact that it is easy to identify the direction of the vector of gravity by applying a low-pass digital filter to the signal of the sensor. Both simple 6dB per octave exponential average and 12dB per octave Butterworth filters have been successfully used for this purpose. The result of the filtering process gives three components of the gravity vector in the pose space of the glove device. Pitch and Roll of the device are then calculated by computing the angle between the gravity vector and the X and Y axises of the sensor device. Note that this type of sensor cannot be used to observe yaw, i.e. rotations around the Z axis.

Pitch and Roll are then used to detect gestures. For this, a gesture start position has been defined, the hand position in which the thumb of the user points upwards. The gesture software detects when the hand is in the start position. Then the pose of the hand is tracked for four simple gestures, i.e. reaching four thresholds in pitch and roll angles. These gestures can also be combined to a sequence, e.g. turning left and then pointing the fingers upwards, so that a total number of four simple and eight combined gestures can be detected. Additionally, the two buttons on the glove device can be used as gesture modifier keys or individual events.

The use of a gesture start position also makes it possible to perform the gesture detection continuously with few false positive detections of the gesture system, provided that the gesture start position is defined according to the requirements of the application.

2.3 Context Interface

The previous section described the low level signal processing tasks required to filter the raw sensor data and how gestures are recognized from this data, thus, transitioning from raw sensor input to contextual information. The authors designed a context collection and distribution framework allowing providers to easily store contextual information in a central repository and allowing client access through a rich programming interface.

The context repository is the central place where references to contextual information are maintained. It allows context providers to register new context sources and clients to discover context information. The query processor translates complex queries to queries on the context repository and allows clients to register event notifications that are triggered when new context information matching the query is available. The

description component manages the interfaces to context, i.e., it maintains the semantic meaning rather than the data needed to represent a piece of contextual information. Context sources are either custom, or use parts of the provided context toolbox to apply reasoning algorithms on the data provided by the underlying sensor layer. Context processing starts with a context provider acting as the bridge to the sensor layer. The provider maintains the set of recognized sensors and exposes both, its context interpretation, i.e., a semantic meaning of the underlying data and context representation, i.e., the data required to represent an actual contextual value. In the described scenario, the context provider exposes three acceleration sensors. Each sensor represents one axis. Additionally, a gesture context is exposed that reflects the latest gesture recognized from the acceleration input. Six gestures are recognized, Left/Right, Up/Down, and Enter/Exit. This data is registered with a context repository which acts as the interface to client applications. It allows clients to query for data, override existing contextual data, and to add additional interpretations for a given context entity. The later can be used by a client for example to override the recognized gesture in cases where the automatic detection failed. A context aware application will then query the context repository to discover available contextual information and will subscribe changes to contextual data found. The client application then can update its behavior (or graphical representation) based on contextual changes.

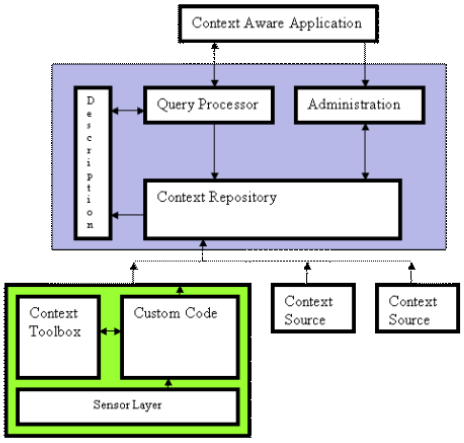

Fig. 3. The context framework

2.4 Legacy Application Interface

A client application that is not context aware and which cannot be adapted to use contextual information natively, can only take advantage from the contextual information indirectly, through a context aware client acting as a proxy to the legacy application.

The proxy would act as a shim between the user input and the application interpreting gestures and translating them to mouse clicks, keyboard presses or other standard windowing events. The application will than act according to the windowing event received without any previous knowledge of the alternative input device used. The authors evaluated different types of legacy applications with respect of being controllable through gestures without any modification done on the original application. The goal was to evaluate their usefulness in the absent of a classic input device. The goal was to navigate through the application only with the help of gestures and, additionally, pitch information from the acceleration sensors. Three applications where chosen for this test:

1. One Application allowing users to move/zoom/select parts of a map. The entire map size was bigger than the available screen. Another application of the same type was evaluated allowing the user to select a point of interest from a map. The points of interest where distributed randomly on the map. The user was only able to navigate with 4 basic gestures through the set (up/down left/right) and to select an entry explicitly by ”Enter”ering the point (which was mapped to the ”enter/exit” gestures).
2. One Application allowing users to navigate within a presentation in a way comparable to a remote control. There were no constraints in terms of timeouts in which the user would have to make a decision to which element to move next.
3. One Application allowing a user to control a toy robot by moving it forward/backward, and to turn left/right. The application logic was such that doing the same gesture twice (or more often) increased the action speed while performing the counter gesture stopped both engines (and thus stopped the robot). The interesting question in this scenario was how a user performs gestures in a time constraint situation - as the robot would potentially hit a wall/the floor.

3 Results

Based on the implemented demonstration applications, we present a number of findings from user feedback. The applications were presented on a public exhibit at the VDE Kongress 2006 in Aachen. Although they do not represent an in-depth survey of user experience, they indicate important steps toward future development.

3.1 Controlling a mapping application

While navigating within a large map was possible using the glove, users had problems with the navigation on several levels:

1. Gestures were not accepted as they proved to be unnatural in the sense that users tried ”move” parts of the map with their hand rather than accepting the ”move a single step in one direction” metaphor.
2. Users misinterpreted the term ”navigating” the map. They expected that the glove would enable them to somehow navigate (the application used in this test is normally used to calculate navigation routes for cars) and wanted to select a start/end point for a route - something not possible with the restricted gesture set.

3. Because the detection of gestures was not perfect, users tended to navigate too far in a direction. For the application had no time dependent functionality this did not result in errors, user were still disappointed when they had to navigate back.

3.2 Controlling a presentation application

Navigating within a presentation application was accepted by the users after a brief introduction to the gestures.

1. Advancing through the slides was intuitive. One slide forward was achieved by performing a ”right” gesture while one slide back was mapped to the ”left” gesture. However, starting and ending a presentation took long because of an unnatural mapping to the available gestures (enter/exit).
2. One problem was due to the null-gesture class. It was relatively hard for the users to avoid unwanted gestures in this scenario mainly because it is common to gesticulate during a presentation with the hands. The used hardware did not allow reliable filtering of unwanted gestures (because the available sensors did not allow an unambiguous detection of a gesture start event) and as a result, unwanted transitions between slides were common.

Gesture-based control of a mobile robot One of the demonstration scenarios includes a physical robot controlled by user gestures. Similarly to The Glove, the Toy Robot offers Bluetooth Serial connection with a simple AT command-based control protocol. As the gesture recognition engine is implemented on a personal computer the glove cannot alone control the robot. Rather, the glove sends raw acceleration data to the computer that interprets this as gestures and forwards the related command (set engine speed) to the robot. The used during the demo was an off-the-shelf ASURO Robot Kit[17] with the following characteristics:

1. Data transfer through Infrared (modified to Bluetooth)
2. 2 light sensors underneath
3. 6 touch sensors in the front
4. 2 independent motors
5. 2 odometers sensors for wheel speed measurements
6. 2 red LEDs on the back
7. Tricolor status LED
8. ATmega8L microcontroller (programmable in C)
9. SDK and software libraries to develop custom code for the robot

The only change we made to the hardware is replacing the IrDA module with a serial-line-enabled Bluetooth module to increase reliability and range of the wireless communication

Controlling a robot was different from the previous application scenario as that it required the user to control a physically moving object by gestures.

1. The main obstacle here was that a user could not ”just wait” to figure out a gesture because the robot would continue its movement while the user performed the gestures. This was mainly a problem during the initial training phase of the user.

2. Users liked the way the robot was controlled and intuitively watched the robot instead of looking at the glove. Gestures for left and right where mostly performed correct while up/down gestures where often misinterpreted by the system (recognized up instead of down). The reason was that users did not wear the glove in the correct way, thus generating different acceleration patterns.
3. Because of misinterpretations of gestures, users relatively often had to stop the test and re-set the robot to free space. As the system would intermittently detect gestures when this was not intended, the robot started moving around quite often.

4 Analysis and Discussion

Concluding from the user feedback, we find that the current gesture input system is probably least suited for the mapping application, as much interaction is needed. The current gesture recognition lacks the necessary robustness and the mapping of gestures is considered unintuitive by the users, a direct manipulation would probably be better suited. This was evaluated in a second demonstrator where acceleration information was transformed directly into map movements, repeating the movement until the glove was moved back to the initial position. Users responded better to this type of navigation but still did not feel comfortable.

The control of a slide presentation application received a more positive feedback by the users, however, users did move their hands unintentionally, leading to false gesture detections and thus unintended slide changes.

The control of the mobile robot was the most critical application in respect of false gesture detections, as physical damage can occur if the wrong command is sent to the robot. However, users liked to use gestures as input and the use of the gesture control seemed natural for this purpose. However, gesture recognition accuracy was not considered sufficient, i.e., the robot had to be stopped manually several times to avoid hardware damages. Concluding, we think that for mobile robot control, the robustness of the gesture detection has to be improved and the mobile robot should be equipped with additional sensors to protect it from hardware damage resulting from unintended gesture commands.

During the tests, we encountered the problem of gesture misinterpretation several times. For example, when the user was performing certain hand movements (typing on the keyboard, walking, waving with hand etc.), these were interpreted as gestures although the user had no intention to perform a gesture at that moment. The problem is akin to the Monty Python sketch titled ”Biggles Dictates a Letter” [18] where it was not clear whether Biggles was actually dictating or merely speaking to the secretary. In the sketch, the problem was solved with an external signal (a Moose’s antler on Biggles’ head) to switch between dictation mode and speaking mode. Similarly we could define two additional gestures, ”Start Gestures” and ”Stop Gestures” to switch gesture recognition on and off. The ”Start Gestures” gesture should be unique enough not to be accidentally performed during everyday movements but easy enough to perform for cases where switches could be frequent.

Acknowledgements

The textile glove used for the gesture interface has been designed by Sabrina Tanner from Lösungsmittel GmbH, Vienna, Austria. The signal processing software for gesture recognition has been implemented by Michael Dippold and Jörn Reimerders at TZI as part of a joint research project funded by Microsoft EMIC. This work has been partly funded by the European Commission through IST Project wearIT@work: Empowering the Mobile Worker by wearable Computing (No. IP 004216-2004). The authors wish to acknowledge the European Commission for their support.

References

1. Handykey Corporation: Twiddler2 wearable coording keyboard, http://www.handykey.com (2005)
2. Boronowsky, M., Nicolai, T., Schlieder, C., Schmidt, A.: Winspect: A case study for wearable computing-supported inspection tasks. In: Fifth International Symposium on Wearable Computers (ISWC '01). Volume 5., IEEE Computer Society, IEEE Computer Society (2001) 163–164
3. de la Hamette, P., Lukowicz, P., Tröster, G., Svoboda, T.: Fingermouse: A wearable hand tracking system. In: Fourth International Conference on Ubiquitous Computing (UbiComp '02), IEEE Computer Society, IEEE Computer Society (2002) 15–16
4. Rgge, I.: Einsatzpotenziale, Nutzungsprobleme und Lsungsanstze mobil tragbarer Informations- und Kommunikationstechnologien. PhD thesis, Universitt Bremen (2006)
5. Möhring, U., Gimpel, S., Neudeck, A., Scheibner, W., Zschenderlein, D.: Conductive, sensorial and luminiscent features in textile structires. In Herzog, O., Kenn, H., Lawo, M., Lukowicz, P., Troester, G., eds.: 3rd IFAWC. Volume IFAWC 3rd international forum on applied wearable computing 2006., VDE Verlag (2006)
6. FrogPad Inc.: Frogpad wearable keyboard, http://www.frogpad.com (2005)
7. Nicolai, T., Sindt, T., Kenn, H., Witt, H.: Case study of wearable computing for aircraft maintenance. In: 2nd International Forum on Applied Wearable Computing (IFAWC) 2005, VDE Verlag. (2005) pp. 97–110
8. Nicolai, T., Sindt, T., Witt, H., Reimerdes, J., Kenn, H.: Wearable computing for aircraft maintenance: Simplifying the user interface. In Herzog, O., Kenn, H., Lawo, M., Lukowicz, P., Troester, G., eds.: 3rd IFAWC. Volume IFAWC 3rd international forum on applied wearable computing 2006. (2006) VDE Verlag.
9. Rekimoto, J.: Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. In: Fifth International Symposium on Wearable Computers (ISWC '01). Volume 5., IEEE Computer Society, IEEE Computer Society (2001) 21–27
10. Metzger, C., Anderson, M., Starner, T.: Freedigiter: A contact-free device for gesture control. In: ISWC. (2004) 18–21
11. Kela, J., Korpipää, P., Mntyjärvi, J., Kallo, S., Savino, G., Jozzo, L., Marca, S.D.: Acceleratometer-based gesture control for a design environment. Journal of Personal Ubiquitous Computing **10** (2006) 285–299
12. Wilson, A., Shaefer, S.: Between u and i: Xwand: Ui for intelligen spaces. In: Proceedings of the conference on human factors in computing systems: CHI 2003. (2003) 545–552
13. Lawo, M.: Ein drahtloser eingabehandschuh fr augmented reality anwendungen. In Gausemeier, J., Grafe, M., eds.: 5. Paderborner Workshop Augmented & Virtual Reality in der Produktentstehung. (2006) 199–208

14. Lawo, M., Witt, H., kenn, H., Nicolai, T., Leibrandt, R.: A Glove for Seamless Computer Interaction - Understand the WINSPECT. In Kenn, H., Glotzbach, U., Herzog, O., eds.: The Smart Glove Workshop. Number 33 in TZI Report (2006)
15. Witt, H., Nicolai, T., Kenn, H.: Designing a wearable user interface for hands-free interaction in maintenance applications. In: IEEE International Conference on Pervasive Computing and Communications (PerCom), Pisa, Italy (2006)
16. Freescale Inc.: Mma7260q fact sheet (2005) avaliable at http://www.freescale.com/files/sensors/doc/fact_sheet/MMA7260QFS.pdf.
17. Gruber, R., Grewe, J.: Mehr Spa mit Asuro. AREXX INTELLIGENCE CENTRE (2005)
18. Chapman, G., Cleese, J., Idle, E., Jones, T., Palin, M., Gilliam, T.: Monty python's flying circus: Salad days: Biggles dictates a letter. TV-Series 1969-1974 (1972) according to http://orangecow.org/pythonet/sketches/biggles.htm.