UbiGesture: Customizing and Profiling Hand Gestures in Ubiquitous Environment

Ayman Atia, Shin Takahashi, Kazuo Misue, and Jiro Tanaka

Graduate School of Systems and Information Engineering, Department of Computer Science, University of Tsukuba, Japan
ayman@iplab.cs.tsukuba.ac.jp, {shin,misue,jiro}@cs.tsukuba.ac.jp

Abstract. One of the main challenges of interaction in a ubiquitous environment is the use of hand gestures for interacting with day to day applications. This interaction may be negatively affected due to the change in the user's position, interaction device, or the level of social acceptance of a specific hand gesture. We present UbiGesture as architecture for developers and users who frequently change locations while interacting in ubiquitous environments. The architecture enables applications to be operated by using hand gestures. Normal users can customize their own hand gestures when interacting with computers in context-aware ubiquitous environments. UbiGesture is based on combining user preferences, location, input/output devices, applications, and hand gestures into one profile. A prototype implementation application for UbiGesture is presented. Then a subjective and objective primary evaluation for UbiGesture while interacting in different locations with different hand gesture profiles is presented.

Keywords: Ubiquitous environment, Hand gesture profiles, Context aware services.

1 Introduction

Human hand gestures are an intuitive way for humans to express their feelings and to interact with several objects in daily life. Hand gestures can be used as a tool for interacting with different applications like media player, presentation viewer...etc. Interaction can be done through predefined hand gestures; however it is difficult to fit all users or enabling users to customize their own hand gestures. The interaction can be affected negatively if the use's change position, interaction device, or application or all of them together. There are two other parameters that can have direct impact on users so that they change their hand gesture pattern. First is the position of the user while customizing his hand gesture. A predefined hand gesture, such as moving the elbow towards the body in a standing position, will be difficult to do when the user is sitting down on a sofa or sitting at a table because of the limited space. Second is the social implication of the hand gesture; some hand gestures may scare, annoy, or disturb other people. For example, user A is sitting on a sofa beside person B. If user A moves his or her arm towards person B (someone A knows), Person B might become annoyed and interpret the gesture as a minor interruption. A more complex reaction

J.A. Jacko (Ed.): Human-Computer Interaction, Part II, HCII 2009, LNCS 5611, pp. 141–150, 2009.

might occur if there is no relationship between people such as in a public space. Such a gesture may scare the other person. The main idea behind UbiGesture is to provide users with enough appropriate hand gesture profiles for any given situation. The system enables users to define and store their hand gestures using single training hand gesture templates inside a ubiquitous environment context. The context includes the user's location, interaction device, application, and defined application functions. The UbiGesture system helps developers to operate their applications with hand gestures. There is no need for developers to understand the details of hand gesture recognition algorithms or user profiling details. The UbiGesture system was built using inexpensive resources, which benefits the usability and availability of the system.

The rest of this paper is divided into sections discussing related work, UbiGesture architecture, primary evaluation, and conclusion and future work.

2 Related Work

The study of gesture recognition with a presentation viewer application was shown in [1]. They show an active region for starting and ending gesture interaction. Also they point out that gestures can be useful in crowded or noisy situations, such as in a stock exchange or manufacturing environment. Head and hand gestures have been used for limited interactions as demonstrated in [2]. They point out the problem of learning gestures and show the importance of customization. Kurze et al. presented personalization of multimodal applications as a design approach [3]. They focus on implicit and explicit customization of systems according to a user's preferences. Kawsar et al. presented customizing the proactive applications preferences in a ubiquitous environment [4]. They present customization in many levels of artifact, action, interaction, and timing preferences. Customizing a tilt rest hand gesture in eight directions has been discussed in our previous work [5]. We compared interactions with remote displays using three methods and user-customized tilt hand gestures. The users' shows increase in accuracy and time to hit targets when they were allowed to customize their hand gestures.

There are many studies that focus on reusing current applications and interfaces in context-aware environments. Nakajima proposed a model for displaying a GUI on multiple output devices [6]. He/She also developed a way for selecting devices according to a user's location. Ronkainen et al. studied the usability of hand gestures in different ubiquitous environments [7]. They conducted a survey on the social implications of hand gestures in public spaces. Moreover, they present a tap gesture for interacting with mobile devices as a type of socially acceptable hand gesture. They point out that there are gestures that are perceived as being threatening in public spaces. Hand-gesture positions were determined from our previous results gathered by Ayman et al. [8]. They show that the position of hand gestures could affect the accuracy and speed of interaction with an interface for entering text.

The information technology revolution in the last ten years had a direct impact on the size and performance of wireless sensors. Some of these sensors have been used for inferring the context of user and providing ubiquitous services. Recently,

there have been many multifunction sensors that have been embedded in small devices [9] [10]. The problem of these small sensors is their short battery life and their high cost. UbiGesture uses wii remote as an interacting device in ubiquitous environments. The Wii [11] remote controller has been suggested as an interaction device for entertainment purpose and some music and art applications [12] [13].The main advantage of the Wii remote is its embedded 3D-accelerometer and optical sensor that is used for motion and position tracking. The Wii remote acceleration range is ±3 G and has been used in many studies for hand-gesture recognition [14] [15].

3 UbiGesture Architecture

The UbiGesture system is used for profiling hand gestures of users depending on the context parameters they exists in. In this research an inexpensive Bluetooth infrastructure was used because this technology is now embedded in most hand held devices. Sanchez et al. used this technique for commercial advertising based on location [16]. Fig. 2. shows an overview of the UbiGesture system. There is a need for different level of users to operate the UbiGesture system (administrator, developer, and regular users). An administrator's main role is managing resources and granting permissions to each user. A developer enables his or her application to be customized by defining keyboard shortcuts or application programming interfaces (APIs). A regular user has to define his or her gesture profile and store it in the network.

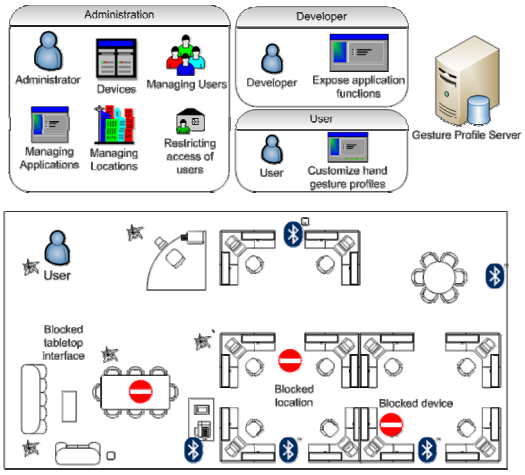

Fig. 1. UbiGesture system overview and Bluetooth infrastructure in office prototype

UbiGesture was built using four main modules each module is responsible for some functions. Fig. **2.** shows the system architecture modules of UbiGesture.

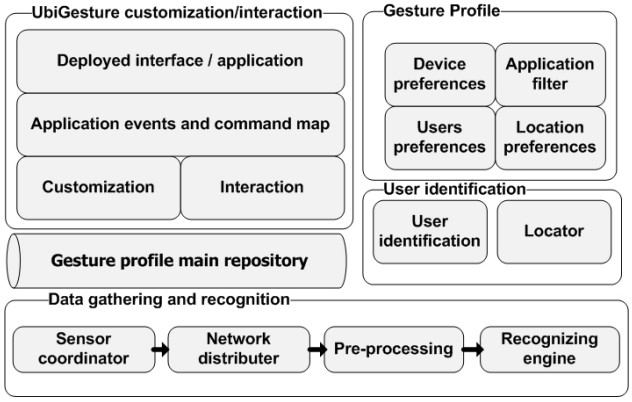

Fig. 2. UbiGesture System architecture

3.1 User Identification Module

UbiGesture assumes that there is a Bluetooth location manager server "Locator" in each location, which was a low-specification desktop PC with a Bluetooth dongle. The locator hosts an application that reads a user's identification by continuously searching for a user's device's media access control (MAC) address within a 30-second interval. Each user may have one or more identification devices such as a hand held mobile device or a Bluetooth sensor. The locator grants access to users for specific devices in one location and it logs the user's entrance and exit time from the location. Whenever users exist in location covered area, it flags a message for all the permitted devices to download the user gesture profile.

User can have access to more than one device in one location; hence, authentication is required "User identification" before a device can be used. Interaction with devices is usually initialized using a keyboard, mouse, or stylus and touch panel. We believe that hand gestures could be similar to someone's unique hand-written signature, by which a user can access a device by entering their device hand gesture signature. In the UbiGesture the system administrator asks the user to define his or her personal hand gesture to control permitted devices. However, sometimes devices could have predefined authentication hand gestures. If the user wants to use a plasma display, for example, he or she first needs to create a specific hand gesture for starting this device; another gesture will be required for a large screen display and so on.

If there are two or more users in one place and both have been granted access to the same device, the User identification module locks the device to be used with only a single user.

3.2 Data Gathering and Recognition

The user has to press the Wii remote's "A" button to start recording a gesture and release the button to stop the recording. Hand gesture *h* can be represented as a sequence of accelerometer sensor readings. A reading at time t can be represented as $A(t)=\{a_{xt},a_{yt},a_{zt}\}$. A sensor coordinator connects the Wii remote, a human-computer interface, and a locator module. The locator module receives readings from the Wii

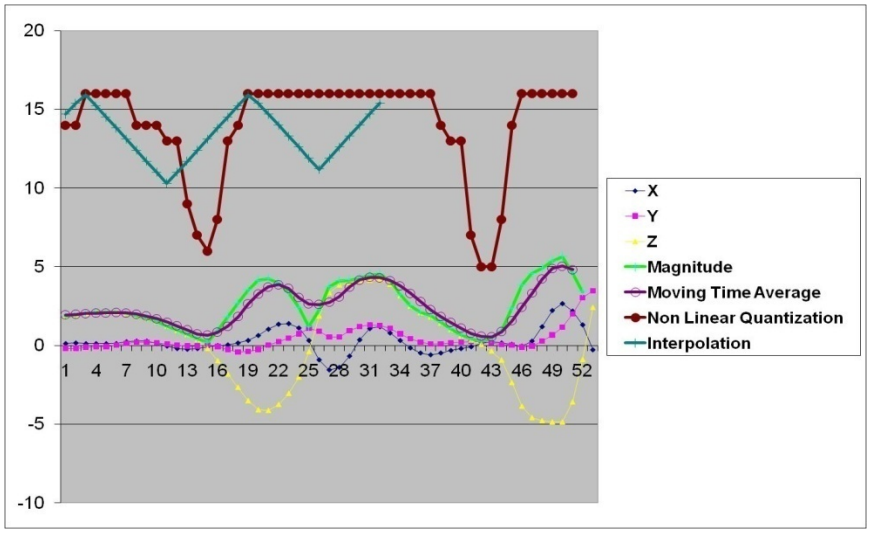

Fig. 3. The analysis of a gesture that entails the shaking of the hand gathered from wii remote

remote and sends them using Transmission Control Protocol/Internet Protocol (Tcp/Ip) to the appropriate authenticated device "Network distributer". On each device there is an application for receiving the data and checking data appropriate receiving order. The collected data pass into three levels of preprocessing for the gathered signal to obtain better recognition of the hand gesture. Fig. **3.** shows the analysis of a gesture that entails the shaking of the hand.

The core of the recognition engine was built using a slightly modified DP-Matching algorithm for recognizing users' hand gestures. The cost function of the algorithm has been calculated as Euclidean distances between two 3d vectors, as shown in Equation 1, where $p_{\{x,y,z\}}$ is one of the accelerometer readings of a user's hand gesture and $q_{\{x,y,z\}}$ is the accelerometer readings inside the list of stored template hand gestures.

$$\text{Cost}= \sqrt[2]{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \quad (1)$$

The minimum value between a user's hand gesture and all stored template gestures is calculated. The result value should be less than the threshold value, which depends on the sensitivity of the application.

3.3 UbiGesture Customization/Interaction

We have designed two prototype applications, one for customizing hand gestures and the other for using defined hand gestures with developer's applications. The command map translates the gesture commands into application understandable events. In the current version of UbiGesture, a hand gesture is translated into keyboard-shortcut sequences. Each keyboard shortcut sequence is mapped to an interface or application function. The application developer has to list all application functions and features in

the main system repository. The interface extracts the features and exposes them for user customization. The customization interface always compares the entered gesture with the stored gestures to avoid conflicts and alarming the user. The interaction interface compares the user's captured hand gesture with all the templates, returns the minimum distance, and compares it with the threshold value. If the minimum distance is below the threshold value, it will execute the command; otherwise, it will ignore the gesture.

3.4 Gesture Profile

A gesture profile is recorded in the main repository of UbiGesture. This record is composed of the user, location, device, application, application functions and recorded hand gesture. The gesture profile is the key record in this research, and whenever the user changes his or her context, the appropriate gesture profile will be downloaded on the device of interaction. From this profile, the system extracts the rest of the information such as the user's preferences and hand gestures. The profile is stored in the network for further access from different locations.

4 Primary Evaluation

For a primary evaluation, we ran experiments with three volunteers. All volunteers were between the ages of 20 and 25, and all were computer science specialists. The customization of hand gestures, which depends on location, its effect on the error recognition rate, and interaction speed, was tested. In addition, we studied the effect of imposing position and social parameters on users interacting with predefined hand gestures. A test bed was established using two Bluetooth location-discovery PCs in two different rooms. The first room (Room A) had a large display screen and a three-seater sofa. The second room (Room B) had a large display screen only (see Fig. 4.).

Volunteers used their cellular phones as an identification device. A prototype application for image manipulation was developed using Microsoft C\# and Windows Presentation Foundation. Table 1 shows the application's 20 functions. Predefined hand gestures for those functions were set to cover complex, simple, different

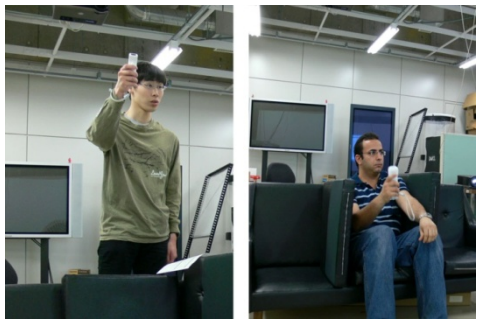

Fig. 4. User positions in Room B (left) and Room A (right)

Table 1. Application functions and gesture patterns

Function Name	Gesture pattern	Function Name	Gesture pattern	Function Name	Gesture pattern
Zoom in all pictures		Zoom camera out		Move camera up	
Zoom out all pictures		Take picture		Move camera down	
Start PTZ camera		Rotate image		Zoom camera in	
Show image	Double Shake	Crop image		Browse down	Shake +
Close application		Black white		Browse left	Shake +
Move camera right		Save captured image, close		Browse right	Shake +
Move camera left		Browse up	Shake +		

geometrical shapes and entailed handshake gestures. These different shapes enable users to imagine what hand gesture he or she can customize using his or her hands.

4.1 Experimental Setup

The administrator granted permission for volunteers to access Rooms A and B, access the big display screen in both rooms, and run the photo application on both devices. The administrator also set a default authentication gesture to start the devices. The developer made all the applications accessible by keyboard shortcuts. For example, to open an image in full view, volunteers select an image by pressing the keyboard arrows and "v". To zoom in, he presses "z", to zoom out, "o", and so on. The Administrator stored the application functions and their corresponding keyboard shortcuts in the main repository. The volunteers were asked to perform two sequences of hand gestures. The first sequence was to browse images, view in full view, rotate, crop, and add black and white effects. The second sequence was to capture an image using a pan tilt zoom (PTZ) camera and save it.

The first experiment involved a full sequence of pre-defined hand gestures. The volunteers were asked to do the experiment in two positions; standing and sitting on the sofa. We put a border between each sofa seat to simulate the presence of another person. The volunteers should not touch this border during the experiments.

The second experiment was to test the UbiGesture on the customization of hand gestures per location. A volunteer customized his hand gestures for browsing images in a sitting position in Room A, (Customization Scenario 1). Then we asked the volunteer to customize his hand gestures for capturing a picture in a standing position in Room B (Customization Scenario 2).

4.2 Results and Discussion

The volunteers were asked to enter 22 gestures in sequence (one session), and they had to make three sessions per experiment, for a total of 198 hand gestures. We measured the average number of hand gestures to finish scenario 1 and 2 (Fig. **5.**5a) and the average interaction time (Fig. **5.**5b).The results show that using customized hand gestures reduces the interaction time and number of hand gestures by an average of 47 percent.

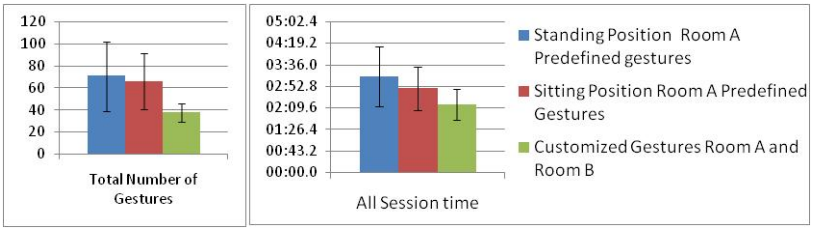

Fig. 5. (a) Average total number of gestures per session, (b) Average time to finish session

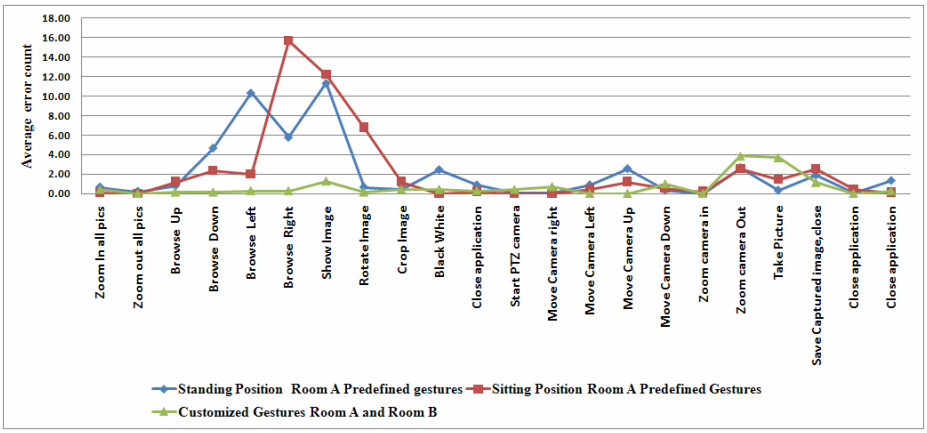

Fig. 6. Average error count of hand gesture sequence

Table 2. Volunteers' satisfaction results for UbiGesture system

Question	Average
Was it easy to interact using predefined hand gestures in a standing position?	1.67
Was it easy to interact using predefined hand gestures in a sitting position?	2.33
Was the customization easy?	2.67
Was it easy to interact using customized hand gestures in a sitting position?	4.33
How many times did you touch the sofa border?	4

The average time needed to customize the hand gestures for Customization Scenarios 1 and 2 was 3 minutes and 20 seconds. When using the predefined hand gestures in the sitting position and interacting with the application, we observed that the volunteers hit the borders or left the seat many times to gesture. In real situations, these gestures may annoy others close by. Also it was remarked that the volunteer sometimes failed to perform the predefined hand gestured towards lower body. Fig.6 shows the average error counts to execute the sequence of hand gestures. The peek seen in Average error count of hand gesture sequence6 for gesture "Browse Right" using a predefined hand gesture and in a sitting position, explains how hard it was for the volunteer to gesture.

The volunteer tried to avoid touching the border nearest to him, so he failed in completing the gesture. When the volunteer was able to customize this gesture, the error rate and time dramatically decreased. When the volunteers completed all the experiments, we asked them to answer a short questionnaire by giving a score from 1 to 5 (5 meaning very good). The results are listed in **Table 2**. We received comments from the volunteers about the need for a GUI feedback showing the pattern of the customized gestures. Sometimes the volunteers forgot the pattern of the gesture they made and requested to re-customize the gesture.

5 Conclusion and Future Work

We developed the UbiGesture system, which profiles a user's gestures according to specific parameters. The system was built using inexpensive resources making it more available to more users. The system also enables developers to add hand gestures to their applications in a few easy steps. The end users can customize their hand-gesture profiles depending on the context, which gives them more flexibility and intuitive interaction. Moreover, we also proposed a solution for starting devices in a shared-resource ubiquitous environment. Using two-handed gestures is a more intuitive way in human interaction. We need to evaluate the usability of UbiGesture by running different scenarios and test beds. We proposed user adapted-systems; however, we need to address concerns about system-adapted hand gestures in which a system recommends a suitable gesture for users depending on the context.

References

1. Baudel, T., Beaudouin-Lafon, M.: Charade: remote control of objects using free-hand gestures. Commun. ACM 36, 28–35 (1993)
2. Keates, S., Robinson, P.: The use of gestures in multimodal input, pp. 35–42 (1998)
3. Kurze, M.: Personalization in multimodal interfaces, pp. 23–26 (2007)
4. Kawsar, F., Nakajima, T.: Persona: a portable tool for augmenting proactive applications with multimodal personalization support, pp. 160–168 (2007)
5. Ayman, A., Takahashi, S., Tanaka, J.: Coin Size Wireless Sensor Interface for Interaction with Remote Displays, pp. 733–742 (2007)
6. Nakajima, T.: How to reuse existing interactive applications in ubiquitous computing environments?, pp. 1127–1133 (2006)

7. Ronkainen, S., Hakkila, J., Kaleva, S., Colley, A., Linjama, J.: Tap input as an embedded interaction method for mobile devices, pp. 263–270 (2007)
8. Ayman, A., Takahashi, S., Sato, D., Tanaka, J.: Evaluating interaction with Popie using coin size wireless sensor, pp. 4-17–4-18 (2008)
9. SunSpot, http://www.sunspotworld.com/
10. Ubisense, http://www.ubisense.net/
11. Wii Remote, http://www.nintendo.com/wii
12. Bruegge, B., Teschner, C., Lachenmaier, P., Fenzl, E., Schmidt, D., Bierbaum, S.: Pinocchio: conducting a virtual symphony orchestra, pp. 294–295 (2007)
13. Lee, H.-J., Kim, H., Gupta, G., Mazalek, A.: WiiArts: creating collaborative art experience with WiiRemote interaction, pp. 33–36 (2008)
14. Shirai, A., Geslin, E., Richir, S.: WiiMedia: motion analysis methods and applications using a consumer video game controller, pp. 133–140 (2007)
15. Schlomer, T., Poppinga, B., Henze, N., Boll, S.: Gesture recognition with a Wii controller, pp. 11–14 (2008)
16. Sanchez, J.-M., Cano, J.-C., Calafate, C., Manzoni, P.: BlueMall: a bluetooth-based advertisement system for commercial areas, pp. 17–22 (2008)