U-TOPIA: Campus-wide Advanced Ubiquitous Computing Environment

박규호, 이주평, 유종운, 임승호, 박기웅, 최현진, 원광연
Kyu Ho Park, Jupyung Lee, Jong-Woon Yoo, Seung-Ho Lim,
Ki-Woong Park, Hyun-Jin Choi and Kwangyun Wohn

Korea Advanced Institute of Science and Technology
Daejeon, 305-701, Korea
kpark@ee.kaist.ac.kr, {jplee, jwyoo, shlim, woongbak, hjchoi}@core.kaist.ac.kr, wohn@vr.kaist.ac.kr

Abstract

U-TOPIA, introduced in this paper, is a campus-wide advanced ubiquitous computing environment. Research in UTOPIA spans various components each of which is essential to realize U-TOPIA: from user device hardware / software, user interface, communication technology, indoor / outdoor testbed, middleware to practical applications. We designed and implemented a wearable computer from the scratch and make use of it as a main user device inside U-TOPIA. In addition to this, as a new user interface, we developed a wireless gesture recognition device, called i-Throw. For data communication and location tracking in U-TOPIA, campus-wide indoor and outdoor testbed was installed. To keep up with highly variable dynamic U-TOPIA environment, a new extensible middleware, called u-ware, was developed. Finally, as a practical application for U-TOPIA, we implemented a ubiquitous testbed room where multiple users interact with various ubiquitous devices or other users in a user-friendly manner. Integrating these components all together, we show that U-TOPIA can be a realistic role model to improve current paradigm of ubiquitous computing environment one step forward within a few years.

1. Introduction

In recent years, the rapid progress of ubiquitous and pervasive computing technology, fueled by either a government-led or a company-led effort to encourage the research, has led to the emergence of various 'smart' places powered by ubiquitous computing technology, ranging from smart room and smart campus to smart city.

In 2005, our team launched a government-funded project aimed at realizing a campus-wide advanced ubiquitous computing environment until the end of 2007. The ubiquitous computing environment was named as *U-TOPIA*, where 'U' stands for 'ubiquitous' and 'TOPIA' stands for 'place' in Greek.

Research in U-TOPIA spans various components each of which is essential to realize U-TOPIA: from user device hardware / software, user interface, communication technology, indoor / outdoor testbed, middleware to practical applications. We designed and implemented a wearable computer from the scratch and make use of it as a main user device inside U-TOPIA. In addition to this, as a new user interface, we developed a wireless gesture recognition device, called i-Throw. A novel algorithm to minimize the interference between different communication interfaces on same 2.4GHz ISM band was suggested. For data communication and location tracking in U-TOPIA, campus-wide indoor and outdoor testbed was installed. To keep up with highly variable dynamic U-TOPIA environment, a new extensible middleware, called u-ware, was developed. Finally, as a practical application for U-TOPIA, we implemented a ubiquitous testbed room where multiple users interact with various ubiquitous devices or other users in a user-friendly manner.

All these efforts are closely related with realizing UTOPIA in our campus.

2. U-TOPIA: Basic Components

Research in U-TOPIA spans various components each of which is essential to realize U-TOPIA: from user device hardware / software, user interface, communication technology, indoor / outdoor testbed, middleware to practical applications. We have tried to push each research field a step forward to meet the ambitious goal, realizing a campus-wide advanced ubiquitous computing environment.

2.1. User Device

In U-TOPIA, a mobile user device is necessary to provide a user with plentiful ubiquitous computing resources. A mobile device should be light-weight, easy to carry, easy to use and it should have aesthetic appearance and social acceptance.

We chose to design and implement a wearable computer from the scratch and make use of it as a main user device inside U-TOPIA. A wearable computer platform, in contrast to either laptop PC or handheld device, allows a user to carry the computing device in a comfortable and natural manner, because clothes have been already a essential component of our daily lives. Large surface area of clothes can be utilized for various purposes and thus, I/O interface does not have to be located in only a small-sized computing device. Moreover, the wearable computer makes it easier to measure and gather bio signal data such as temperature and heart rate by integrating body-attached sensors and computing devices upon a same clothes interface.

2.2. User Interface

No matter how plentiful ubiquitous computing resources are, it means nothing from the perspective of a user if a user cannot access the resource easily via a well-organized user interface. User interface should be easy to learn, easy to use and user-friendly and environment-friendly. It is obvious that traditional user interfaces, such as keyboard, mouse and touch screen, are not suitable for a primary user interface inside U-TOPIA. In U-TOPIA, since plentiful resources locate outside a user, rather than inside a user device, a brand-new user interface is required to manipulate various outside resources in a user-friendly manner. To meet this requirement, we developed a wireless gesture recognition device, called i-Throw. Using this device, a user can express one's intention easily by using one's spatial movement and hand gesture.

2.3. Communication Technology

In U-TOPIA, we support three communication interfaces: WLAN, Bluetooth and ZigBee. WLAN is used for campus-wide network services such as service discovery, security transaction and program download, and Bluetooth is for wireless headset, mouse and keyboard. ZigBee is adapted for low-power short message transfer or indoor location tracking.

There are several studies on the interference and coexistence problems among communication devices that use the 2.4GHz ISM band. Up to now, many coexistence models were presented in terms of WLAN and others. In case of the interference between WLAN and Bluetooth, various problems were discussed in [1]. However, the coexistence problems between WLAN and ZigBee are not widely addressed yet. The study about coexistence problems between WLAN and ZigBee is very important because in a near future, ZigBee is expected to be widely applied in various communication environment, such as wireless sensor networks, personal area networks and body area networks. To minimize the interference between communication devices, we have designed and implemented the dynamic channel allocation algorithm in device driver and application layers that use communication interface devices. For that, we investigate the frame error rate(FER) for each communication device, and dynamically allocate the appropriate channel of ZigBee devices considering the channel status. A more detailed explanation is provided in [4].

2.4. Indoor and Outdoor Testbed

Since U-TOPIA aims for a campus-wide environment, it is essential to build a large-scale testbed inside which various services can be operated. Two important components of a target testbed are a communication infrastructure and a location-tracking infrastructure.

Communication infrastructure lays the groundwork for ubiquitous computing. Communication inside U-TOPIA is made possible by wireless mesh network that is installed inside our campus. Figure 1 illustrates the wireless mesh network that is actually installed in U-TOPIA.

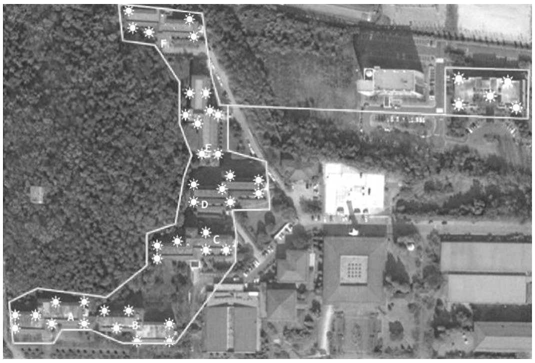


Figure 1. Wireless Mesh Network in U-TOPIA

Location-tracking infrastructure is necessary for a location-based service. In outdoor environment, we made use of widely-used GPS information to track the location of moving objects. In indoor environment, we basically used Zigbee signal strength-based location tracking mechanism. To do this, we installed enough number of ZigBee sensor nodes inside two selected buildings inside U-TOPIA. The ZigBee sensor node broadcasts beacons periodically, which uses 2.4GHz as a physical channel. The moving user receives beacon signals from the ZigBee communication interface. When the user receives multiple beacons from the multiple sensor nodes, the users who have received the beacons can identify his location by calculating each Received Signal Strength Indicator(RSSI) value from each sensor node.

During the measurement, however, we found that the resolution of location sensing using this mechanism was not sufficient for our target application. Thus, we also utilized UWB-based location tracking device[7] whose typical accuracy is 6 inches(15cm). Due to the high cost of this solution, UWB-based location tracking device have installed in only two rooms inside U-TOPIA.

2.5. Middleware

In U-TOPIA, we assume a situation where thousands of users move here and there, interact with each other or ubiquitous computing environment, share information with authorized other users, access to diverse devices for diverse purposes, run various location-based applications. In this situation, an extensible middleware framework is necessary to keep up with highly variable dynamic environment. We have been working with middleware team and they developed a extensible middleware, called u-ware[5]. u-ware is composed of light-weight service discovery protocol, distributed information sharing, context manager and instance service loader, all of which are useful to manage dynamic data and develop new application utilizing various ubiquitous resources.

2.6. Application

Finding an attractive and useful target application have been one of primary concerns for ubiquitous computing researchers. In U-TOPIA, a good application should be practical and closely related with students' campus activity. A good application should also consider ubiquitous resources distributed in U-TOPIA. Moreover, a good application should be able to demonstrate the significance of aforementioned components effectively. We took 'user-friendly interaction with ubiquitous devices using i-Throw' as a target application. This application assumes a ubiquitous testbed room, that is full of various ubiquitous devices and sensors. In this room, a user tries to control various devices and execute various operations on one's own. As the number of controllable devices and supported operations increases, it becomes extremely difficult for a user to supervise this room properly. We attempt to resolve this problem by introducing i-Throw device. This device makes it much easier to supervise this room, so that even a person who trained for only a few minutes can perform various jobs correctly. As it becomes common that in a class room, laboratory, library or conference room there are many computing devices, this application is likely to be closely related with campus activity. In addition, this application is closely related with aforementioned components: user device, user interface, communication technology, testbed and middleware. Thus, this application can be seen as appropriate for U-TOPIA.

3. User Device and User Interface

In this section, among six components of U-TOPIA, the user device and the user interface are explained in detail.

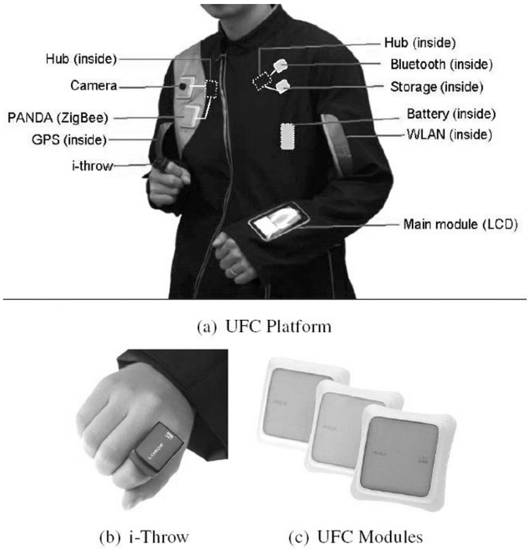
3.1. User Device: UFC

In last section, we mentioned that we utilized a wearable computer as a main device inside U-TOPIA. Our wearable computer is called Ubiquitous Fashionable Computer (UFC) [8], which is named based on our special emphasis on its wearability, aesthetic design and close interaction with ubiquitous environment.

The UFC system consists of a portable wearable computer with various communication interfaces and user interfaces, and ubiquitous environment. The basic design concept of UFC wearable computer is modularity and extensibility.

Our UFC consists of several module parts: main module including CPU and Memory, communication modules including various communication interfaces, and user interface modules with I/O interfaces. In the main module, the core of the UFC system is an ARM based Intel XScale processor: the PXA270. Main features of this processor include clock scaling and dynamic voltage scaling up to 624MHz. With this, power management of wearable computer can prolong the life time of the platform. Main memory of UFC is 256MB, which is relatively large capacity for a mobile device. However, with this large capacity, we can support a wide range of services such as audio and video transmission, and java virtual machine based middleware services.

The implemented UFC platform is shown in Figure 2. UFC modules are distributed on a garment, considering the distribution of weight and aesthetic design. Moreover, each UFC module can be attached and detached easily on a garment, allowing users to construct one's own UFC platform. Since we utilized a standard USB protocol to communicate between the main module and various UFC modules, due to the hotswap capability of USB devices, each UFC module can be attached and detached while the system is running.



(a) UFC Platform

(b) i-Throw (c) UFC Modules

Figure 2. UFC Platform Design and Implementation

The success of wearable computer relies on not only wearability, but also the aesthetic appearance and social acceptance. We tried to find the solution to fulfill the requirements by repeating the prototyping bodystorming progresses. We defined the target users as young university students and drew design concepts by analyzing their activities in everyday life and fashion trend. In addition, we have made effort for each part of the UFC platform to look like familiar fashionable components: for example, an attachable/detachable module is comparable to a button of clothing and an i-Throw device is comparable to a ring. Also, some wireless module can be worn as a form of a necklace. Moreover, as an extreme case, we made a ZigBee earring that has only a ZigBee transceiver and several LEDs, which is tiny enough to be worn as a form of an earring.

Operating system running on UFC is GNU/embedded Linux with kernel 2.6. Linux 2.6 with ARM processor shows more deemed feasible performance in real time embedded system than lower versions. Efficient middleware platform is implemented with UFC to provide various helpful services with low overhead and power.

3.2 User Interface: i-Throw

In last section, we mentioned that a wireless gesture recognition device, called i-Throw, was developed as a primary user interface. This device is small enough to be worn on one's finger like a ring. It has a three-axis accelerometer and a three-axis magneto-resistive sensor for recognizing a gesture and the direction of the finger. It also has a ZigBee transceiver for transmitting the recognized gesture information to the UFC platform. Figure 1-(b) shows its appearance.

The gesture recognition performed by the i-Throw device consists of two stages: *feature extraction* stage and *testing* stage. The feature extraction stage is a preprocessing stage to find reference features of each gesture. A feature f is represented by a 4 dimensional vector as follows:

$$f = (A_{THx}, A_{THy}, A_{THz}, T_H) \quad (1)$$

, where A_{THx}, A_{THy} and A_{THz} are the acceleration thresholds of each axis and T_H is time duration threshold.

In the feature extraction stage, we should find appropriate thresholds for each possible input gesture. Due to the limitation of space, we omit the detailed explanation of the feature extraction stage. Table 1 shows the extracted features of various input gestures.

In the testing stage, i-Throw device compares the output of the accelerometer with each reference feature for over T_H seconds. If one of the features is matched, then i-Throw transmits the recognized gesture to the UFC platform via ZigBee interface.

The gesture recognition algorithm is designed to be simple enough to run on a microcontroller inside the i-Throw by extracting the minimum set of required features and using threshold-based simple features.

Table 1. Features of various gestures

Gestures	Features
Throwing	(2g+,X,2g+,150msec)
Increasing	(X, 0.5g+, X, 500msec)
Decreasing	(X, 0.5g−, X, 500msec)
scrolling up	(0.5g−, X, X, 500msec)
scrolling down	(0.5g+, X, X, 500msec)
selecting	(X, X, 0.5g−, 70msec) &
	(X, X, 1.5g+, 70msec)
scanning	(1.7g+, X, X, 100msec) &
	(X, 1.4g+, X, 100msec)

We summarized and illustrated the gesture sets that the i-Throw recognizes in Figure 3. Other possible gestures, scrolling up/down and canceling, are intentionally omitted here.

Every time a UFC user points to a device, the UFC platform displays the selected target device upon its screen. This feedback information helps the UFC user find the correct target device. Similarly, a scanning gesture allows the user to investigate controllable devices inside the room. This scanning operation is similar to the operation of moving a mouse pointer across several icons in a typical PC desktop environment.

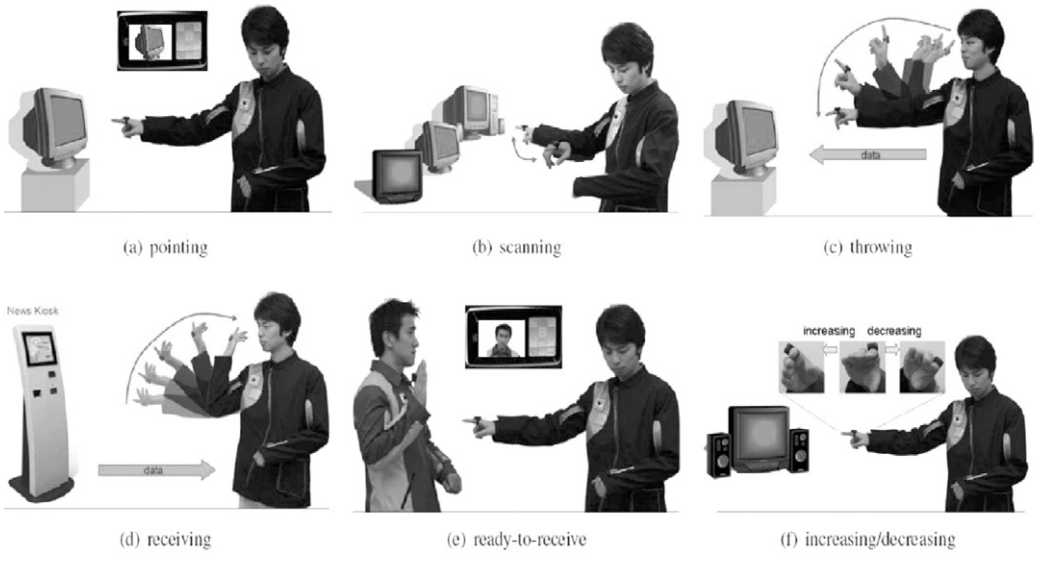
'Ready-to-receive' gesture is necessary for a UFC user to express one's intention to receive other UFC users' objects. When one user makes a pointing or scanning gesture, only limited users who make the 'ready-to-receive' gesture can be selected.

4. Target Application: User-friendly Interaction with Ubiquitous Environment using i-Throw

As mentioned in Section 2.6, we took 'user-friendly interaction with ubiquitous devices using i-Throw' as a target application. To execute this application, we have implemented a ubiquitous testbed room where multiple UFC users interact with various ubiquitous devices or other UFC users. Figure 4 illustrates the concept of the ubiquitous testbed room . In addition, we have implemented a practical application that runs upon the UFC platform and the ubiquitous devices, which makes it possible to exchange the various objects and control ubiquitous devices very easily.

4.1 Motivation

Due to its small form factor, most portable devices, including our UFC platform, have only small-sized display and limited input devices. The



(a) pointing (b) scanning (c) throwing

(d) receiving (e) ready-to-receive (f) increasing/decreasing

Figure 3. Gesture Sets of i-Throw

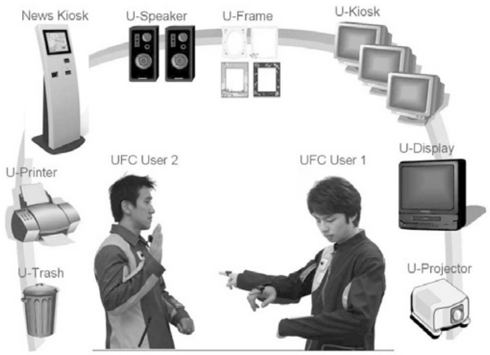


Figure 4. The Concept of the Ubiquitous Testbed Room

UFC platform has 2.5″ LCD display and 12 input buttons, which are definitely insufficient to monitor the status of a UFC main module and various peripheral modules, control the modules, and send a user's intention to the UFC platform.

This problem is exacerbated when a UFC user tries to control various ubiquitous devices using one's UFC platform: as the number of controllable ubiquitous devices increases, it becomes more inconvenient to find one among them and exchange information with it, due to the small-sized display and limited input devices of the UFC platform. Efficient utilization of a small-sized display and intelligent mapping of various commands on the input buttons can partially solve this problem. However, such an approach usually makes it difficult to learn how to use the device, which degrades the usability of the UFC platform. One recent workshop underscored that usability is one of the primary challenges in a next-generation "smart" room, that is full of various ubiquitous devices[2].

We attempt to resolve this problem by making full use of spatial resources inside the testbed room: given that various ubiquitous devices are spatially distributed inside the testbed room, a UFC user can express one's intention easily by using one's spatial movement and gesture. For example, let us assume that one UFC user takes a picture and intends to put it on a public display so that other people can see the picture he takes. From the perspective of the user, the most natural way of reflecting one's intention on the environment is pointing his finger at the public display and throwing one's picture at the public display. If this kind of a user-friendly spatial gesture interface is supported, the limitation of the I/O resources of the UFC platform can be overcome by fully utilizing abundant spatial resources.

For the UFC platform to support such a gesture interface, the following components are necessary:

* *gesture recognition device* recognizes the target device that a UFC user is pointing at and the gesture such as 'throwing' and 'receiving'.

* *location tracking device* keeps track of a UFC user's location. This is necessary because finding the target device that a UFC user is pointing at is dependent on the absolute location of the UFC user. We utilized UWB-based location tracking device[7] whose typical accuracy is 6 inches(15cm).

* *location server* gathers and manages the location information of both UFC users and ubiquitous devices. When one UFC user points at a specific device, the recognized gesture information is sent to the location server and it finally decides what the target device is.

* *service discovery platform*: For a UFC platform to exchange information with one ubiquitous device, the UFC platform should be able to obtain the interface through which the communication is made possible. The interface includes IP address, port

number and simple properties of the device. We have been working with middleware team and they developed ubiquitous service discovery (USD) protocol as part of KUSP (KAIST Ubiquitous Service Platform)[5]. USD protocol was originally based on UPnP[6], which is widespread as a service discovery, and this protocol is simplified to avoid XML parsing overhead. In this study, the USD protocol and KUSP was used as a service discovery platform.

* *application that runs upon a UFC platform* infers a UFC user's intention based on a gesture, a target device and previous operations and conducts a corresponding operation.

Among these necessary components, in Section 2, we already explained first four. Thus, in this section, we focus on the application that runs upon a UFC platform.

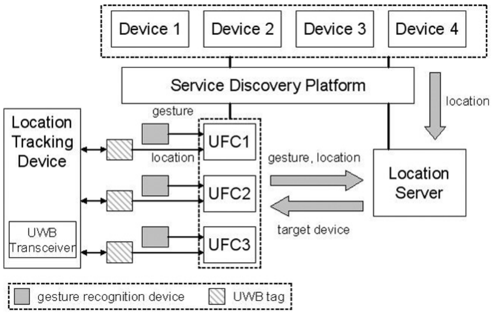


Figure 5. Overall architecture

Overall architecture is illustrated in Figure 5. Each UFC user holds both a gesture recognition device and a UWB tag. By receiving messages from location tracking device, each UFC can identify its own location. The location server gathers information about current gesture and location of each UFC user. In addition to this, the location server also manages information about the location of each ubiquitous device.

Once a UFC user points to a specific device, because the location server knows the location of both the user and the device, the location server can identify the fact that the user points to the device. After this identification, a new interface between two can be established with the help of the service discovery platform.

4.2 Detection of Target Device

Our target detection algorithm is based on Cone selection which is used in virtual computing environments[3]. A cone is cast from i-Throw and a set of devices that intersect with it are chosen. Additionally, we have modified that typical cone selection algorithm to vary the area of the cone adaptively to improve the overall target detection accuracy. To do this, the orientation of i-Throw and the position of the UFC user and devices should be known. The location information is gathered and managed by the location server, as mentioned in Section 3.1. And the orientation of i-Throw can be obtained by combining the accelerometer and magnetic sensor outputs. The accelerometer is used for tilt compensation. By using the orientation and position information, target detection can be performed properly.

We define the object sets which can be sent and received inside our ubiquitous testbed room as follows:

* *music* : mp3 format, music objects can be played either by a UFC platform or public speaker,

called *u−speaker*

∗*news* : html format, news objects can be obtained from news kiosk

∗*photo* : jpg format, photo objects are generated when UFC users take picture

Next, we further define the target device sets which consist of possible target devices in the ubiquitous testbed and the characteristics of each one, which are summarized in Table 2.

Table 2. Target Device Sets

Target device	Supported objects	Flow of objects
u-display u-projector	news, photo	input, output
news kiosk	news	output
u-printer	news,photo	input
u-trash	news,photo,music	input
u-speaker	music	input, output
UFC	news,photo,music	input

Among these devices, the *news kiosk* automatically gathers recent news from internet web sites and displays each one, that is refreshed every 10 seconds. When one UFC user sees the interesting news upon the news kiosk, he or she can obtain the news by making 'receiving' gesture towards the news kiosk.

The *u−trash* functions as a symbol of 'deleting a file'. Similarly to the natural way of using an actual trash, throwing something that is useless anymore at the trash, if one UFC user makes a throwing gesture towards the u−trash, the current object will be deleted automatically. The u−trash device is different from other devices in that it is not an electric device; it acts only as a marker standing for a particular operation and thus actual operation is not conducted inside it. This example gives us insight as to how to fully utilize spatial resources inside the room. If various markers whose symbolic meaning can be easily interpreted are added inside the room and each corresponding operation is efficiently conducted, the spatial gesture interface allows UFC users to conduct various operations in a user−friendly manner.

Table 2 points out that some target devices do not support all kinds of objects: music objects can be supported by the device which is capable of plays the music such as u−speaker or UFCs and the news kiosk only supports the news object. Table 2 also shows that some devices only allow either an input channel or an output channel: a u−trash only opens an input channel, while a news kiosk only opens an output channel.

4.3 UFC Operation Sets

Until now, we have summarized the gesture sets, the object sets and the target device sets that our ubiquitous testbed supports. When a UFC user makes one gesture among the given gesture sets, the UFC platform should be able to decide which object and which operation needs to be processed. That decision depends on the type of a target device and the type of a recent operation. For example, if one UFC user takes a picture('taking a picture' operation) and made a 'throwing' gesture towards the u−printer, it is highly likely that the user intends to print the photo he has just took. On the other hand, if the user reads a news using his UFC terminal('reading a news' operation) and makes a 'throwing' gesture towards the u−printer, the user may intend to print the news he has just

Table 3. UFC Operation Sets

operation	selected object	condition			change of mrso sets
		target_device	gesture	etc.	
increase volume	none	u-speaker	increasing	none	none
decrease volume	none	u-speaker	decreasing	none	none
delete a file	mrso	u-trash	throwing	none	mrso=none
send a file	mrso	u-display	throwing	mrso_type = photo or news	none
		u-printer		mrso_type = photo or news	none
		UFC		none	none
	mrso_music	u-speaker		none	mrso_type=music mrso=mrso_music
read news	1)	news kiosk	receiving	none	mrso_type=news mrso=1)
	other.mrso	none	ready-to-receive	other.mrso_type==news and other.target_device==this and other.gesture==throwing	mrso_type=news mrso=other.mrso
listen to music	1)	u-speaker	receiving	none	mrso_type=music mrso=1)
	other.mrso	none	ready-to-receive	other.mrso_type==music and other.target_device==this and other.gesture==throwing	mrso_type=music mrso=other.mrso
view a photo	1)	u-display	receiving	none	mrso_type=photo mrso=1)
	other.mrso	none	ready-to-receive	other.mrso_type==photo and other.target_device==this and other.gesture==throwing	mrso_type=photo mrso=other.mrso

* '1)' indicates that the corresponding object depends on the status of the selected target device.

read rather than photos or other objects. These examples show that the UFC platform has to keep track of the recent operations, more specifically, recently selected objects. To do this, we define the followings: mrso indicates the most recently selected objects among various music, news and photo objects. mrso music, mrso news and mrso photo indicate the most recently selected music, news and photo objects, respectively.

Finally, Table 3 summarizes the UFC operation sets which were actually utilized in our demonstration. In this table, '1)' indicates that the corresponding object depends on the status of the selected target device. The demonstration video clip is available in [8].

5. Conclusion

U-TOPIA, introduced in this paper, is a campus-wide advanced ubiquitous computing environment. We summarized six essential components to realize U-TOPIA paradigm in our campus: user device, user interface, communication technology, indoor / outdoor testbed, middleware and practical applications. Since 2005, we have tried to push each research field a step forward to meet the ambitious goal, realizing a campus-wide advanced ubiquitous computing environment.

Specifically, we designed and implemented a wearable computer from the scratch and make use of it as a main user device inside U-TOPIA. In addition to this, as a new user interface, we developed a wireless gesture recognition device, called i-Throw. An novel algorithm to minimize the interference between different communication interfaces on same 2.4GHz ISM band was suggested. For data communication and location tracking in U-TOPIA, campus-wide indoor and outdoor testbed was installed. To keep up with highly variable dynamic U-TOPIA environment, a new extensible middleware, called u-ware, was developed. Finally, as a practical application for U-TOPIA, we implemented a ubiquitous testbed room where multiple users interact with various ubiquitous devices or other users in a user-friendly manner. All these efforts are closely related with realizing U-TOPIA in our campus.

We believe that U-TOPIA can be a realistic role model to improve current paradigm of ubiquitous computing environment within a few years. As a future work, we will attempt to apply our indoor application to outdoor testbed, so that a user can extract suitable information from outdoor components, such as building, billboard, large display and vehicles.

▌References

[1] IEEE802.15.2 Specification, Coexistence of Wireless Personal Area Networks with Wireless Devices Operating in Unlicensed Frequency Bands, Aug. 28, 2003.

[2] M. Back, S. Lahlow, R. Ballagas, S. Letsithichai, M. Inagaki, K. Horikira and J. Huang, "Usable Ubiquitous Computing in Next-genration Conference Rooms: Design, Evaluation, and Architecture," *UbiComp 2006 Workshop*, Sep. 2006.

[3] J. Liang and M. Green, "JDCAD: A Highly Interactive 3D Modeling System," *Computers&Graphics*, 18(4), 1994, 499-506.

[4] M. Kang, J. Chong, H. Hyun, S. Kim, B. Jung and D. Sung, "Adaptive Interference-Aware Multi-Channel Clustering Algorithm in a ZigBee Network in the presence of WLAN Interference," *IEEE International Symposium on Wireless Pervasive Computing*, Puerto Rice, Feb. 2007.

[5] Y. Song, S. Moon, G. Shim and D. Park, "Mu-ware: A Middleware Framework for Wearable Computer and Ubiquitous Computing Environment," *Middleware Support for Pervasive Computing Workshop at the 5th Conference on Pervasive Computing & Communications (PerCom 2007)*, New York, Mar. 2007.

[6] UPnP Forum, UPnP Device Architecture 1.0, Version 1.0.1, Dec. 2003.

[7] Ubisense, http://www.ubisense.net

[8] KAIST UFC Project, http://core.kaist.ac.kr/UFC