u-Photo: Interacting with Pervasive Services Using Digital Still Images

Genta Suzuki[1], Shun Aoki[1], Takeshi Iwamoto[1], Daisuke Maruyama[1], Takuya Koda[1], Naohiko Kohtake[1], Kazunori Takashio[1], and Hideyuki Tokuda[1,2]

[1] Graduate School of Media and Governance, Keio University
[2] Faculty of Environmental Information, Keio University, 5322 Endo, Fujisawa, Kanagawa 252-8520, Japan
{genta, shunaoki, iwaiwa, marudai, acky, nao, kaz, hxt}@ht.sfc.keio.ac.jp
http://www.ht.sfc.keio.ac.jp/

Abstract. This paper presents u-Photo which is an interactive digital still image including information of pervasive services associated with networked appliances and sensors in pervasive computing environment. U-Photo Tools can generate a u-Photo and provide methods for discovering contextual information about these pervasive services. Users can easily find out this information through the metaphor of 'taking a photograph'; the users use u-Photo by clicking on a physical entity in a digital still image. In addition, u-Photo makes managing information more efficient because the still image has embedded visual information. Using u-Photo and u-Photo Tools, we conducted various demonstrations and performed usability tests. The results of these tests show that u-Photo Tools are easy to learn. We also present that the time that expert u-Photo users take to find the object in piles of u-Photos is shorter than the time it take to find the object in piles of text-based descriptions.

1 Introduction

A principal theme of pervasive computing is the interaction between users and pervasive services. Networked devices such as networked appliances, networked sensors, and other computing-enabled everyday objects are becoming more common. This means that there are not only increasing number of devices but also incleasing the pervasive services associated with these devices.

The increase of these pervasive services now requires intuitive ways to discover and use them. Jini [8], and UPnP [14] provide methods to discover networked appliances' services. Directory services [4][21] also supply information about networked services. However, they don't return enough information about the services. Users need to know not only the virtual and networked information (e.g. host names, URLs, types of services) but also the links between the services' physical entities and virtual information because the pervasive services actuate or sense the physical world. Suppose that there are multiple printers in a location. How can visitors tell the network name of a specific printer? It is inconvenient if systems that supply information about the services do not show the links between network names and physical entities of the printers.

Our approach uses interactive, digital still images for intuitive service discovery and use. U-Photo is a digital still image with information of pervasive services. U-Photo uses the simple action of taking a photograph to discover pervasive services and then uses image-based intuitive GUI to use the services. Users can interact with the services anytime as long as they have the u-Photos.

In this paper, we discuss the basic concept of u-Photo, the design of u-Photo Tools for using of u-Photo, and experiments based on the prototype. The rest of this paper is structured as follows: Section 2 describes the challenges faced in our work. In Section 3, we describe the basic concept of u-Photo. Section 4 presents the design of u-Photo media and u-Photo Tools. Details about the prototype appear in Section 5. Section 6 describes the prototype experiments, which included usability tests. Section 7 presents a discussion about u-Photo. We review related work in Section 8. Finally, we present our conclusions in Section 9.

2 Pervasive Services

2.1 Classification of Pervasive Services

We classify our target pervasive services into following three groups: environmental context sensing, appliance control, and personalization.

Environmental context sensing services acquire sensor values from networked sensors. In pervasive computing environment, sensors such as ultrasound sensor, infrared ray sensor, and RF readers have network connectivity. Furthermore, networked small sensor units [3][7] have been developed and they build sensor networks [6]. These sensors will be embedded everywhere in the environment and provide sensing services. We can acquire context information of physical world such as temperature, and brightness from these services. For instance, a user can monitor his/her room temperature from a different location. Since sensors have limited sensing areas, it is important for users to know the locations of sensors and sensing areas.

Appliance control services control networked appliances. A number of networked appliances have already released (e.g., air conditioners, microwave ovens, refrigerators, and lights) and it seems that all appliances will be networked in pervasive computing environment. We can control these appliances from mobile devices, PCs, and other input devices. One example of appliance control is controlling home appliances from the user's office.

Personalization services allow control of arbitrary appliances as if they are ours. Needs for personalization services arise when nomadic users want to reproduce a service to a different device [9][11][17]. Transferring media such as videos and audios is an example of a personalization service. In other words, we can say these personalization services are roaming user's tasks. Personalization service also supports applying personal settings to unfamiliar appliances. For example, the temperature at which a user feels comfortable, can be applied to each air

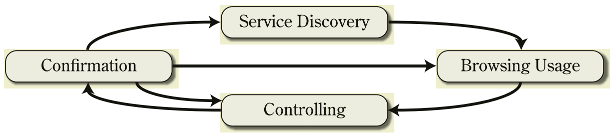


Fig. 1. User operation cycle of pervasive services

conditioner of rooms he/she visits. Personalization services works in two steps: first, users mark off settings and execution states of appliances, and reproduce them in another location.

2.2 User Operation Cycle

When using pervasive services, there is a operation cycle of *Service Discovery*, *Browsing Usage*, *Controlling*, and *Confirmation* shown in Fig.1.

– Service Discovery
  At first, a user has to discover the services that he/she wants. This operation is classified to two ways. The first classification is by function. For example, the user would look up a service as "What's the temperature here?" The second classification is by actual device (e.g., "Are there any air conditioners near here?"). In both cases, since the devices have limited actuation or sensing areas, the users have to know the links between the services' physical and virtual information.
– Browsing Usage
  After the service discovery, the user needs to know how to get data from the sensors and control the appliances. In other words, this operation involves finding out how to use the pervasive services. In this case, there are the following two types of information: information for connecting to the service and information about command-to-action binding. The former includes the IP address of the device, the port number, or other information to specify service in the network. Information such as clicking the "ON" button means turning on TV is included in the latter.
– Controlling
  After browsing the usage information, the user actually controls the appliances or sensors.
– Confirmation
  Finally, the user confirms whether the device performed the actions he/she intended. To do so, the user looks at the device or gets to the service state by using service commands. If the results are not as expected, the user goes back to the Service Discovery, Browsing Usage, or Controlling operations.

2.3 Challenges

Our research aims to provide a method that improves the user's operation cycle. To do that, the following three issues must be considered:

- **How to intuitively remind a user of pervasive services?** ··· How does a user discover the services? In particular, when he/she works with an invisible device such as a sensor embedded in a wall – how to do that? As we mentioned above, information about the services on the network is inseparable from physical entity because each pervasive service has limited actuation or sensor area in the physical world.
- **How to interact with pervasive services easily and instantly?** ··· After a user finds a pervasive service usage, how does he/she browse the service's usage? When a user wants to get temperature here and now, how does he/she do so? One way is that he/she looks up the information required for using the services such as the URL, and then executes the client application for using the service on his/her PC, and finally, he/she executes the operation. However, this would be difficult for novice users.
- **How to manage pervasive services' information?** ··· When a user controls a device from a remote location, he/she needs information about the service such as the device name and the location of the device. Moreover, personalization requires storing information about the services' states. However, it is difficult for users to manage a lot of information. When there are many caches of text descriptions of pervasive services, a user would be hard pressed to remember which description applies to which service. The stored information should be easily distinguishable from other information.

3 Concept of u-Photo: Interactive Digital Still Image

Our concept is based on an interactive, augmented still image called *u-Photo*. A u-Photo includes following three types of information about pervasive services:

1. **Physical Entities of Pervasive Services.** A digital still image can show the corresponding physical entities of pervasive services. For example, if an air conditioner in user's home appears in a still image, the image shows that the figure is a physical entity corresponding to the air conditioner service.
2. **Network Information about Pervasive Services.** Client applications for pervasive services require information about services' network. For example, when a user wants to print documents on a networked printer, he/she needs to know network information such as the printer's name, the IP address, and the type of the printer.
3. **State Information of Pervasive Services.** Personalization services need the state information of the source device, in order to transfer them to the destination device. An example of state information is content data and a timecode for playing in the video services.

3.1 Augmented Image-Based Interaction

U-Photo deals with the issue of service discovery in two phases, generating and viewing. The simple action of taking a photograph triggers the discovery of services. Users can easily determine the target to focus on and the time needed

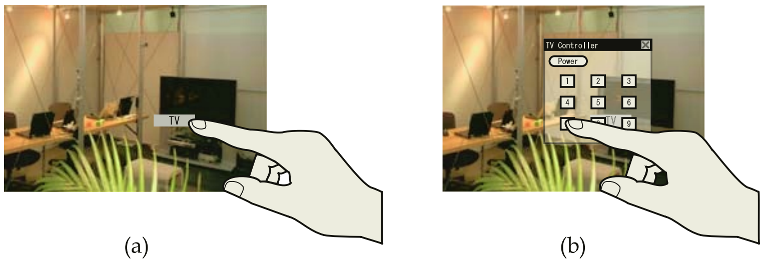


Fig. 2. The image-based interaction with pervasive services: (a) User clicks TV icon on a still image, and (b) GUI of TV control application superimposes on the still image

to capture it. Then, the discovered information is saved in a still image such as u-Photo. When a user opens u-Photo, notations such as ‘Video Service’ and ‘Sensor Service’ appear on the still image, and the user knows that pervasive services are embedded.

U-Photo also presents an image-based intuitive interaction with pervasive services. Just by clicking on the target objects in the photo image, the pervasive service’s client application is superimposed (see Fig.2). There is no need to know the IP address, the URL, or any information about the network.

To address the issue of managing information efficiently, since u-Photo media include still images, users can find the network information by browsing images even if there are a lot of them. Keeping state information is intuitive since the still image shows information about visual state of the services. In addition, digital still images are suitable for carrying and distributing. Once a user takes a u-Photo, he/she can send it to friends by attaching it to e-mail.

3.2 Pervasive Services and Their Eyemarks

Each service must have a physical entity that appears on a still image. We call the physical entity a *service eyemark*. We define three patterns for configuring service eyemarks.

The first pattern occurs when devices are visible and are the service eyemarks themselves (see Fig.3(a)). For example, display devices can be a service eyemark because they are revealed. When one device provides multiple pervasive services, it can be the service eyemark for all services.

In a case where the target devices are embedded, services working at these devices have external service eyemarks (see Fig.3(b)). In particular, sensor devices in a pervasive computing environment tend to be tiny and embedded. Since the appliances/sensors have an actuation/sensing area, users can guess the area being denoted by a service eyemark. For instance, a user can issue “temperature near the plant pot” if the plant pot is configured as a service eyemark.

The last type of service eyemarks is a combination of these two cases. A revealed device doubles with the service eyemark of an other device (see Fig.3(c)). A user would issue “temperature near the TV display” if the TV display is configured as a service eyemark of a sensor that stands near the TV display.

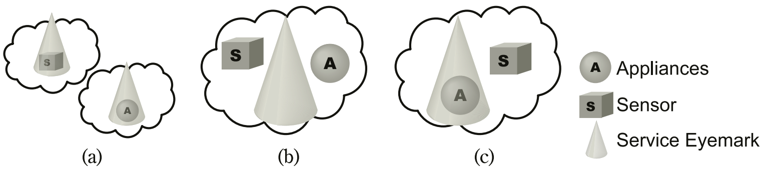


Fig. 3. Models of service eyemarks: (a) Services are working at object (a case where the object is the appliance/sensor). (b) An object has no services but stands near the appliances/sensors. (c) An object has services, and the object stands near other appliances/sensors

3.3 Scenario

In order to clarify our research goals, we present several scenarios using u-Photo.

- Scenario 1: Controlling Devices
  Bob takes pictures of his room, which are stored as u-Photo in his PDA. He goes out to work, forgetting to turn off the room light. After finishing work, he realizes he might have left the room light on. To check whether the light is on or not, he uses the u-Photo Viewer in his PDA and taps the "light icon" displayed on top of the light image in the u-Photo. His u-Photo responds and shows that the room light's state is on. He then taps the OFF button, which is displayed in the u-Photo, to turn off the room light.
- Scenario 2: Discovering Services in an Unknown Environment
  Bob is in a project meeting in a different department and wants to print a document - an easy job as he can just take a u-Photo of the printers he sees there to select one suitable, and to start the print job. The u-Photo automatically configures the printer's entry on his laptop PC.
- Scenario 3: Recording States Information of Services
  One day, Bob and Ann were watching a video at Bob's home, but Bob needed to go out to answer a phone call. Bob stored the state of the video service, such as the content information and the time code, in a u-Photo. He paused and turns off the TV from the control information of the devices. After returning to her home, Ann received an e-mail from Bob with the u-Photo attached. She opens the u-Photo and watches the rest of the video at her desktop computer using u-Photo.

4 System Design for u-Photo

We will describe the system design for our concept. We design u-Photo Tools as systems to create and view u-Photos. U-Photo Tools consist of the *Eyemark Lookup Server*, *u-Photo Creator*, and *u-Photo Viewer* as shown in Fig.4. Eyemark Lookup Server and u-Photo Creator are systems for generating u-Photo. When a new pervasive service is installed in the environment, the environment developer such as an administrator of a building registers the information about the service to the Eyemark Lookup Server. After the information is registered, users can take a u-Photo of the service. When a user takes a photo using a device with u-Photo

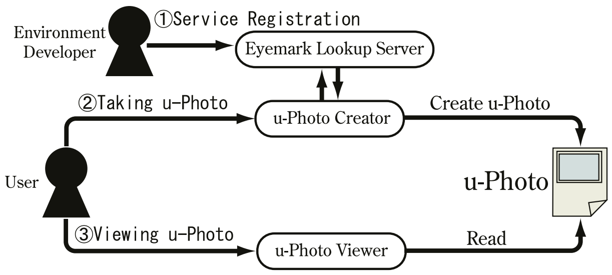


Fig. 4. Overview of u-Photo and u-Photo Tools

Creator installed, the u-Photo Creator looks up the network information in the Eyemark Lookup Server and generates a u-Photo. u-Photo Viewer is u-Photo's viewing application.

4.1 u-Photo Media Design

This subsection describes the visualization model of u-Photo and format of u-Photo Media.

In the u-Photo visualization model, there are three layers in visualizing services (see Fig.5). The *Photo Layer* shows the image of an ordinary photo. The u-Photo overlays two visual layers called the *Tag Layer* and the *Application Panel Layer* on the traditional Photo Layer. In the Tag Layer, service eyemarks are tagged with clickable icons. The Tag Layer will appear on the Photo Layer when a user first opens a u-Photo. Clicking an icon triggers the display of the Application Panel Layer. A GUI of the target service's client application, such as a TV control panel or the GUI for acquiring sensor data, is visualized in the Application Panel Layer. A user decides on a target service by searching the Photo Layer, then invokes application by clicking icons in the Tag Layer, and then uses the service from the client application GUI of the Application Panel Layer.

We will now introduce the u-Photo media format. This format is based on a JPEG, with an XML description of pervasive services in the comment area. Fig.6 shows a DTD of the XML format. `<u_photo>` has three attributes:

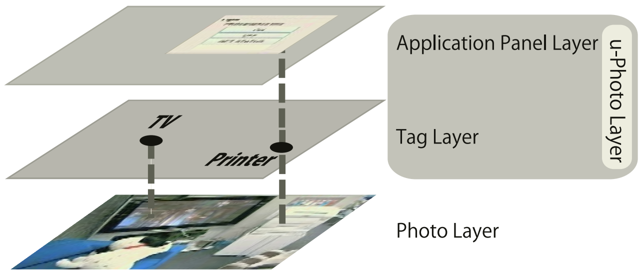


Fig. 5. Visualized layers in u-Photo

```
<?xml version="1.0"?>

<!ELEMENT u_photo (location_info, timestamp, focusing_area)>
<!ATTLIST u_photo xsize CDATA #REQUIRED>
<!ATTLIST u_photo ysize CDATA #REQUIRED>
<!ELEMENT location_info (#PCDATA)>
<!ELEMENT timestamp (#PCDATA)>
<!ELEMENT focusing_area (service_eyemark)?>

<!ELEMENT service_eyemark (coordinate)>
<!ELEMENT service_eyemark (appliance)+>
<!ELEMENT service_eyemark (sensor)+>
<!ATTLIST service_eyemark id CDATA #REQUIRED>
<!ATTLIST service_eyemark name CDATA #REQUIRED>
<!ELEMENT coordinate (x, y)>
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>

<!ELEMENT appliance (application_info*)>
<!ATTLIST appliance id CDATA #REQUIRED>
<!ATTLIST appliance name CDATA #REQUIRED>
<!ATTLIST appliance eyemark_type CDATA #REQUIRED>

<!ELEMENT sensor (application_info*)>
<!ATTLIST sensor id CDATA #REQUIRED>
<!ATTLIST sensor name CDATA #REQUIRED>
<!ATTLIST sensor eyemark_type CDATA #REQUIRED>
```

Fig. 6. DTD of u-Photo XML

`<location_info>`, `<timestamp>` and `<focusing_area>`. `<location_info>` shows the location name, global positioning system (GPS) information, or other location information. The `<service_eyemark>`, which shows the description of pervasive services, is found in the `<focusing_area>`. The Tag Layer represents the `<service_eyemark>`. For the icons put on the still image, the `<service_eyemark>` has a service eyemark ID , service eyemark name, a coordinate for the service eyemark on the still image (`<coordinate>`), and one or more appliances/sensors related to the service eyemark (`<appliance>` and `<sensor>`). `<appliance>` or `<sensor>` is used to construct the Application Panel Layer. Both the `<appliance>` and `<sensor>` have a device ID, device name, binding to service eyemark that shows whether the device is at the service eyemark or the device is near service eyemark, and application information.

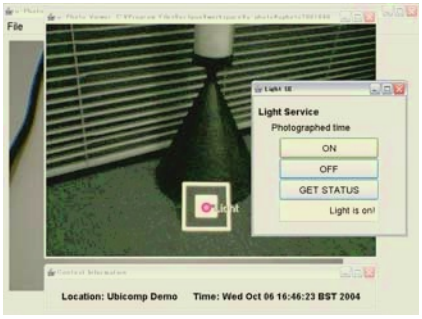
<*application_info*> is used for describing the network and state information of pervasive services. Each pervasive service's client application defines the XML format for <*application_info*>. An XML tag example of the application information is shown in Fig.7. This is a simple application to control the light. In the XML tag, there is the IP address of the light server, its port number, a command for controlling the light, and the state of the light at the time the u-Photo was created. From this description, a simple ON/OFF button GUI of the light (Fig.8) is created without requiring the users to know the IP address or any other information.

4.2 Eyemark Lookup Server

To detect service eyemark coordinate in a still image, we adopted image processing because other methods, such as attaching an RF-tag or IR-transmitter,

```
<UI name="Light">
 <state>OFF</state>
 <button name="ON">
  <ip>192.168.10.6</ip>
  <port>12345</port>
  <command>LIGHT_ON</command>
 </button>
 <button name="OFF">
  <ip>192.168.10.6</ip>
  <port>12345</port>
  <command>LIGHT_OFF</command>
 </button>
 <button name="Get Status">
  <ip>192.168.10.6</ip>
  <port>12345</port>
  <get_command>GET_STATUS</get_command>
 </button>
</UI>
```

Fig. 7. XML description of simple light ON/OFF application



Light is on!

Location: Ubicomp Demo Time: Wed Oct 06 16:46:23 BST 2004

Fig. 8. GUI of simple light ON/OFF application

are difficult to use for detecting coordinates in a still image. Processing photo images has two approaches. One approach attaches visual tags and detects the ID from the tag's color, marks, or figures. Another approach picks up the shapes of the target objects without attaching any visual tags. In the latter approach, it is difficult to distinguish objects with the same shape. Thus, we adopted the method that attaches visual tags to each service eyemark.

Next, we discuss how to bind visual tags to the service eyemarks. We assume that there is an Eyemark Lookup Server in each space. The Eyemark Lookup Server has a database of bindings between IDs of visual tags, and service eyemarks in addition to pervasive service's network information assigned to service eyemarks. If a user takes a u-Photo and visual tags are detected in u-Photo image, the tag IDs that are the results of image processing are sent to the Eyemark Lookup Server, and the network information of pervasive services with the information of the service eyemarks is returned. In additional, the Eyemark Lookup Server can issue new visual tags if a new service eyemark is installed in the target space. These mechanisms enable users to update service information from the visual tags.

4.3 u-Photo Creator

The u-Photo Creator runs on the u-Photo Camera, a digital camera with network connectivity. The modules of the u-Photo Creator are shown in Fig.9. There are six steps, as follows, in creating a u-Photo:

1. *Visual Tag Installer* periodically downloads visual tag information for image processing from the Eyemark Lookup Server.
2. *Camera Controller* provides the interface for controlling the camera. When a user presses the shutter button, the captured image will be delivered to the Image Processor.

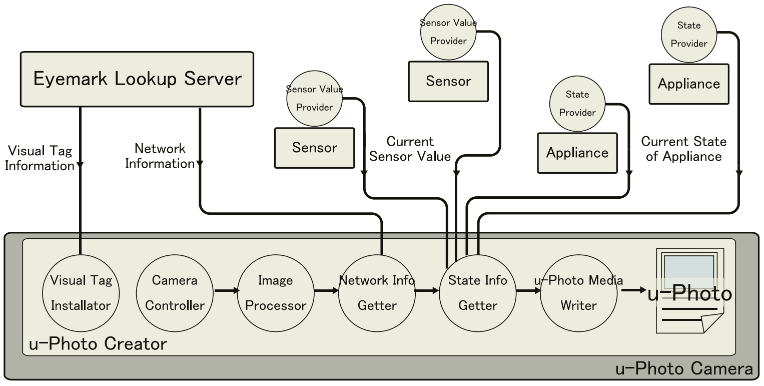


Fig. 9. u-Photo Creator

3. *Image Processor* detects a service eyemark on the photo images. Visual tag IDs and the coordinates in the still image are acquired. This information is delivered to the Network Information Getter.
4. *Network Information Getter* sends visual tag IDs to the Eyemark Lookup Server and obtains pervasive services' network information.
5. *State Information Getter* obtains state information such as the state of an appliance and its current sensor value.
6. *u-Photo Media Writer* transforms the collected information to the u-Photo's XML format and combines the XML data and the captured still image into JPEG format data.

4.4 u-Photo Viewer

The u-Photo Viewer displays both the Photo Layer and the u-Photo Layer. The client application runs on users' computers to perform services. The u-Photo Viewer has three modules as shown in Fig.10:

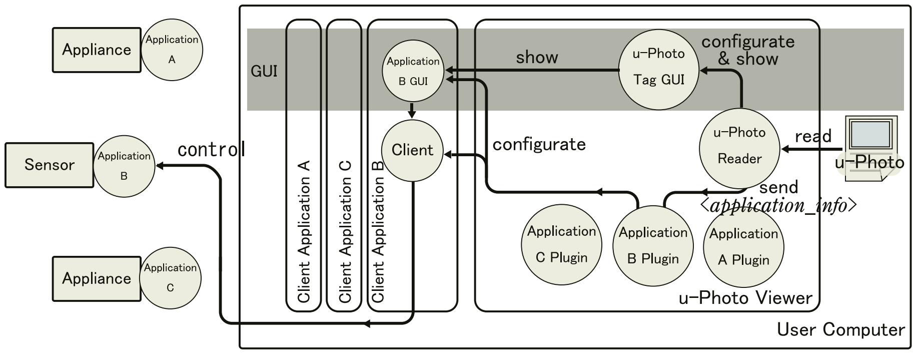


Fig. 10. u-Photo Viewer

1. The *u-Photo Reader* reads the XML data of the u-Photo file when a user opens a u-Photo file.
2. The *u-Photo Tag GUI* displays the Tag Layer using the service eyemark information from the u-Photo Reader module.
3. The u-Photo Reader also passes the XML tag to the *Application Plug-In module.* The Application Plug-In module configures the client application's network information. This module enables users to use client applications without doing any manual configuration. Each Application Plug-In module is downloaded from the Internet on demand according to the <*application_info*>'s XML tag.

5 Prototype Implementation

We used the Sony Vaio typeU with an attached USB web camera (see Fig.11(a)) for the u-Photo Camera. Table 1 and Table 2 present the devices' specification. We used visual markers from the AR Toolkit [10] as tags (see Fig.11(b)). The Image Processor was written with the AR Toolkit Version 2.65 for Windows. The AR Toolkit typically processes real-time streaming video, but we configured it to process images when a user presses a shutter button, as in Fig.11(c). Fig.12 shows the GUI sequence from taking a u-Photo to controlling the devices

Table 1. Specification of VAIO typeU

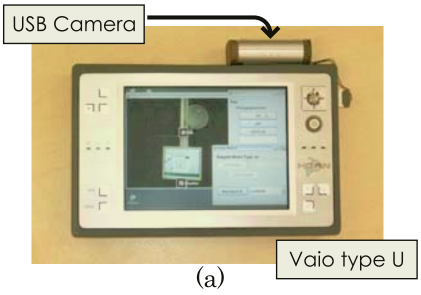
Model	Vaio type U (VGN-U70P)
Dimensions (WxDxH)	167mm x 26.4mm x 108mm
Weight	550g
CPU	Intel Pentium M 1GHz
Memory	512MB
OS	Windows XP Professional
JAVA	J2SE 1.4.2
Display	800x600(5.0inch)

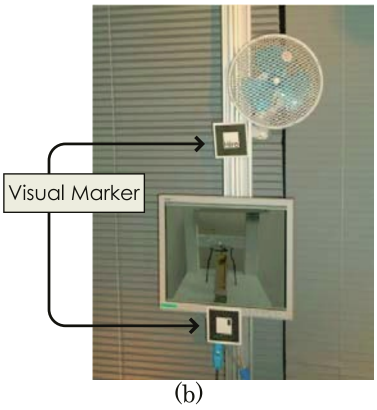
Table 2. Specification of USB Camera

Model	PCGA-UVC11
Dimensions(WxDxH)	60mm x 33mm x 34mm
Weight	42g
Optical Sensor Type	CMOS - 370,000 pixels
Video Capture Speed	30 frames per second
Resolution	640 x 480

We implemented the following four services and applications: appliance control, sensor information viewer, printing service, and suspendable video service.

– Appliance Control
  A TCP/IP-based power-control box is attached to control a light and an electric fan. Their control application is a simple network application in which the client application sends a control command in text format to IP address and port of the control server. Service eyemarks are configured by the light and the fan themselves.
– Sensor Information Viewer
  For networked sensor units, U.C. Berkeley Mica2s are used. There is a sensor server at a PC that connects to the gateway node of the Mica2. The sensor services' service eyemarks are a light, an electric fan, a printer, and a display near the sensors.





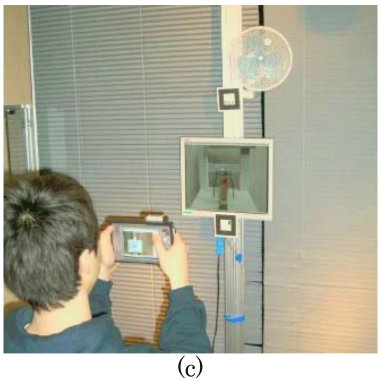


Fig. 11. Prototype implementation: (a) u-Photo Camera. (b) Visual markers of AR Toolkit. (c) Taking a u-Photo

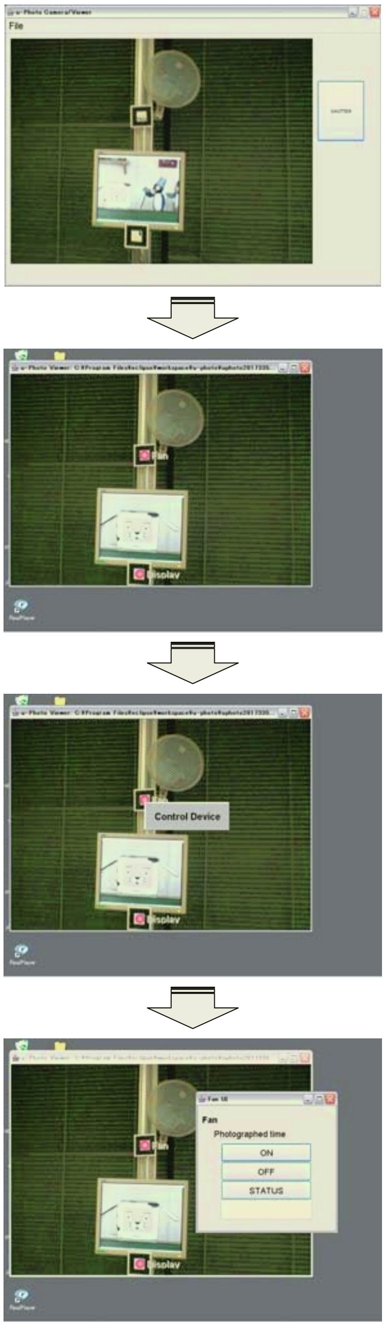


Fig. 12. GUI sequence from taking u-Photo (shown in Top figure) to controlling device using u-Photo Viewer

– Printing Service
  The printer application supports not only printing from the printer GUI invoked by clicking the printer image on u-Photo but also a drag-and-drop action, meaning dragging a document and dropping it on the printer image.
– Suspendable Video Service
  A QuickTime streaming server provides video services. To realize suspendable video service, we adopt Wapplet framework [9] that deals with media data. When we take a u-Photo of a PC's display that plays a video, the Wapplet framework running on the PC records the the URL and the time code of the video as a Wapplet profile. The Wapplet profile is written as a `<wapplet>` description (one of the <*application_info*>) of the u-Photo XML. Other video devices easily resume the same video from the same scene using the u-Photo.

The execution time from pushing the shutter button to displaying a u-Photo is 1.9 seconds. The time for creating a u-Photo is 0.9 seconds. The rest of the time is used to draw a GUI of the u-Photo Viewer.

The difference between the data format of u-Photo and that of normal JPEG file is text data of u-Photo XML. In our experimental applications, data of u-Photo XML description changes JPEG file size from 43 KB to 45 KB at the maximum.

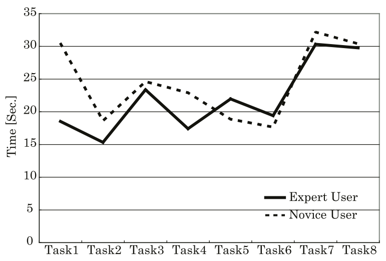
6 Usability Analysis

We performed two tests to measure the following three usability metrics of u-Photo and u-Photo Tools.

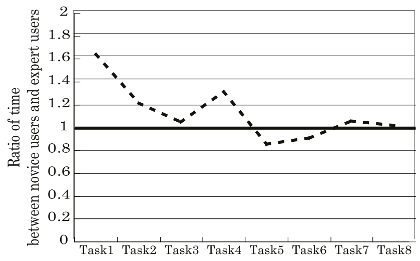
– Learnability of u-Photo Tools
– Subjective satisfaction of u-Photo Tools
– Efficiency of managing pervasive services' information as u-Photo media.

In the first test, subjects completed multiple tasks using u-Photo Tools and then filled out questionnaires. In the test, the subjects, including 12 novice users and 5 expert users of u-Photo Tools, had eight tasks to complete. In each task, the subjects either controlled appliances or received sensor data: e.g., turning on a light using u-Photo Tools and getting temperature near a plant pot. We measured how easy u-Photo Tools was to learn to use by comparing the time spent by novice users of u-Photo to finish tasks to the time spent by expert u-Photo users to finish a task. The questionnaire's answers show us the system's subjective satisfaction.

The result of comparing the time presents that using u-Photo Tools is easy to learn. The graphs shown in Fig.13 denote the expert and novices' learnability about the controlling devices. Fig.13(a) shows the time it took to finish each task, and Fig.13(b) denotes the ratio of the time taken by the experts and novices that is shown in Fig.13(a). There is a difference of over ten seconds between the novice users' time and expert users' time for Task1, which is the first task. But there almost be no difference between the expert and novices users after the first task.



(a)



(b)

Fig. 13. The time spent for controlling devices. (a) Average time that novice users and expert users took for controlling devices in each task. The solid line shows the average time of expert users and the dotted line represents average users' time. (b) Learning curves of novice users

To measure subjective satisfaction, we asked the novice users to fill out our questionnaire after completing the test. The questionnaire uses the Likert scale [12]. The statements and results are shown in Table 3. Users indicated their degree of agreement with each statement on a 1-5 scale for each statement (1 = strongly disagree, 2 = partly disagree, 3 = neither agree nor disagree, 4 = partly agree, and 5 = strongly agree). From the table, we can see that users are highly satisfied about Q1, Q3, and Q6 in both cases of acquiring information and controlling devices. In contrast, users disagree on Q2. The average ratings of the questionnaire are 3.89 for capturing and 3.94 for controlling. To calculate the average rating, the rating of the negative questions (Q2 and Q4) is reversed. The value 3.6 is known as a better estimate of "neutral" or "average" subjective satisfaction [13]. Therefore, the subjective satisfaction with our system is better than average. However, the subjects who agreed with Q2 said that there are two factors that frustrated them: GUI responses occasionally became slow and recognition of visual markers sometimes failed. These are performance issues. Thus, improvement of the prototype's usage could make subjective satisfaction better.

The objective of the second test was to evaluate the efficiency of managing pervasive services information with u-Photo media. We compared the time that

Table 3. Questionairre's results

Question	*1	*2
Q1: It was very easy to learn how to use this system.	4.83	4.83
Q2: Using this system was a frustrating experience.	2.58	2.5
Q3: I feel that this system allows me to achieve very high productivity.	3.92	3.83
Q4: I worry that many of the things I did with this system may have been wrong.	2.17	2.17
Q5: This system can do all the things I think I would need.	3.08	3.17
Q6: This system is very pleasant to work with.	4.25	4.5

*1 Average ratings for capturing information using u-Photo Tools
*2 Average ratings for controlling devices using u-Photo Tools

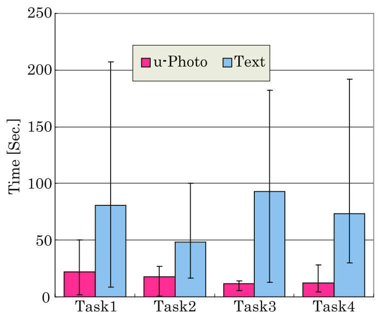


Fig. 14. The time for finding a file

expert users took to find the object in piles of u-Photos with the time it took to find the object in a pile of Wapplet profiles. Wapplet profiles are text format files and include information for personalization. In the test, we had five expert u-Photo users as trial subjects. Each subject downloaded 21 u-Photos and 21 Wapplet profiles to his/her PC and searched for four objects from the files, an air conditioner, a microwave oven, a printer, and a video display. We recorded the time it took to find the correct file by using u-Photo and Wapplet profiles.

The time spent in finding a file is shown in Fig.14. This graph shows the maximum, average, and minimum time spent. In each task, the time spent using u-Photo is shorter than the time spent using Wapplet profile in text format. In Task3 and Task4, in which other files also included printers and video displays information, there was quite a difference of time spent between u-Photo and text information. From these results, we conclude that the efficiency of managing information in u-Photo format is better than the conventional text format.

7 Discussion

Demonstrations at UbiComp2004 [19], and UCS2004 [18] also showed us users' impressions of u-Photo Tools. Over 200 participants used or viewed the system in close up. We found that our approach seemed to attract the participants, but the difference between the shape of u-Photo Camera (shown in Fig.11(a)) and a traditional camera would cause misunderstanding about u-Photo's the use. Therefore, next prototype of u-Photo Camera should be camera-shaped with a traditional shutter button.

Because the visual tags can treat the same devices as different service eyemarks, using visual tags seems to scalable against the number of services. In addition, while there is no global need for unique visual tags, the Eyemark Lookup Server supports unique visual tags. We actually used LED transmitters, which are identified by their color and blinking pattern, in our first prototypes but found them too hard to configure. In contrast, it is easy to generate visual markers in the current prototype. The environment developer simply decides on a unique black-and-white marker pattern, prints it, and attaches it to a service eyemark.

A scalability issue of users arises in the environment that has a lot of users per service. A user authentication module for u-Photo Creator and u-Photo Viewer will be necessary. In addition, u-Photo Viewer also needs exclusive access control for the controlling devices. If two users control the same light at the same time, the light should operate based on only one of the two commands determined by user's authority, location, and so on.

8 Related Work

There have been similar researches that share a part of our motivation. Passage [16] and mediaBlocks [1] use physical objects to transfer information between devices. In Passage, physical objects such as a watch and a key chain are called the "Passenger." When users want to transport digital objects, they only move the Passenger from the source device to the destination device. The mediaBlocks, which are electronically tagged wooden blocks, serve as physical icons that transport online media. Using physical objects is useful for the immediate use of personalization service but unsuitable for long-term use of multiple objects. Suppose a user has a great deal of Passengers/mediaBlocks. How can the user know which one has which information? On the other hand, since u-Photo includes the still image, it would be easier to grasp the binding between files and information.

There is a wide variety of systems that visualize environment information. NaviCam [15] displays situation-sensitive information by superimposing messages on its video see-through displays using PDAs, head mounted displays, CCD cameras, and color-coded IDs. InfoScope [5] is an information augmentation system that uses a camera and a PDA's display without any tags on objects. When a user points him of her PDA to buildings or places, the system displays the name of the place or the stores in the building on the PDA. DigiScope [2] annotates an image-using, visual, see-through tablet. In this system, a user can interact with embedded information related to a target object by pointing to the object. Although these researches are similar to u-Photo in terms of annotating an image, the researchers focused on real-time use where a user can interact with a target object in front of the user now. On the other hand, we concentrated on recording pervasive services information and reusing it.

Truong, Abowd, and Brotherton [20] have developed applications in which tasks are recorded as streams of information that flow through time. Classroom 2000, one of their applications, captures a fixed view of the classroom, what the instructor says, and other web-accessible media the instructor may want to present. In this approach, which stream to record or when to record the stream depends on each application. In addition, since the tasks they target on are never executed again, every state of the task needs to be recorded as a stream. On the other hand, the tasks we focused on are reproducible, and we note the states of tasks that are captured only when the user releases the shutter to produce digital photos.

Several products have already been provided that focus on recording contextual information to digital photos. Digital cameras provide states (e.g., focal length, zoom, and flash) and cellular phones provide GPS information. However, the present products and the photo format do not provide methods for noting the pervasive services information and using photos as user interfaces of a target object in the photo.

9 Conclusion

To address the difficulty in discovering and using pervasive services, this paper presented u-Photo, an interactive, digital still image. Taking a photograph also captures information for service discovery. U-Photo, which is generated by taking a photo, becomes a GUI for specifying the target services. We have developed u-Photo Tools for generating and viewing u-Photo. Experiments with the prototypes gave us the following three usability results: (1) After novice users of u-Photo Tools use u-Photo Tools only a few times, they can complete tasks in the same amount of time as expert users. (2) The subjective satisfaction with u-Photo Tools is better than average. (3) Users can find information in u-Photo a file easily more than text-based files. However, we learned that improving the u-Photo Camera hardware will make our system more useful. We also think that a user authentication and device access control will make u-Photo Tools a more scalable and robust system.

Acknowledgements. This work has been conducted partly as part of the YAOYOROZU Project by the Japanese Ministry of Education, Culture, Sports, Science and Technology, and the Ubila Project by the Japanese Ministry of Internal Affairs and Communications.

References

1. H. Ishii B. Ullmer and D. Glas: mediaBlocks: Physical Containers, Transports, and Controls for Online Media. In *Computer Graphics Proceedings (SIGGRAPH'98)*, 1998.
2. A. Ferscha and M. Keller: Digiscope: An Invisible Worlds Window. In *Adjunct Proceedings of the 5th International Conference on Ubiquitous Computing (UbiComp 2003)*, 2003.
3. H.W. Gellersen, A. Schmidt, and M. Beigl: Multi-Sensor Context-Awareness in Mobile Devices and Smart Artefacts. In *Mobile Networks and Applications*, 2002.
4. E. Guttman, C. Perkins, J. Veizades, and M. Day: Service Location Protocol, version 2. In *Internet Request For Comments RFC 2608*, 1999.
5. I. Haritaoglu: Infoscope: Link from Real World to Digital Information Space. In *Proceedings of the 3rd International Conference on Ubiquitous Computing.* Springer-Verlag, 2001.
6. W. Heinzelman, J. Kulik, and H. Balakrishnan: Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proceedings of the International Conference on Mobile Computing and Networking*, 1999.

7. J. Hill and D. Culler: A Wireless Embedded Sensor Architecture for System-Level Optimization. Technical report, U.C. Berkeley, 2001.
8. Sun Microsystems Inc. Jini technology overview. *Sun White papers*, 1999.
9. T. Iwamoto, N. Nishio, and H. Tokuda: Wapplet: A Media Access Framework for Wearable Applications. In *Proceedings of the International Conference on Information Networking*, 2002.
10. H. Kato, M. Billinghurst, I. Poupyrev, K. Imamoto, and K. Tachibana: Virtual Object Manipulation on a Table-top AR Environment. In *Proceedings of the International Symposium on Augmented Reality (ISAR 2000)*, 2000.
11. T. Kawamura, T. Hasegawa, A. Ohsuga, and S. Honiden: Bee-gent: Bonding and Encapsulation Enhancement Agent Framework for Development of Distributed Systems. In *the 6th Asia Pacific Software Engineering Conference*, 1999.
12. M.J. LaLomia and J.B. Sidowski: Measurements of Computer Satisfaction, Literacy, and Aptitudes: A Review.: *International Journal on Human Computer Interaction* volume 2, 1990.
13. J. Nielsen and J. Levy: Measuring Usability-Preference vs. Performance. In *Communications of the ACM* 37, 1994.
14. Universal Plug and Play Forum. http://www.upnp.org.
15. J. Rekimoto and K. Nagao: The World Through the Computer: Computer Augmented Interaction with Real World. In *Proceedings of Symposium on User Interface Software and Technology.* ACM, 1995.
16. N.A. Streitz S. Konomi, C. Muller-Tomfelde: Passage: Physical Transportation of Digital Information in Cooperative Buildings. In *Proceedings of the Second International Workshop on Cooperative Buildings (CoBuild'99)*, 1999.
17. I. Satoh: A Mobile Agent-based Framework for Location-based Services. In *Proceedings of IEEE International Conference on Communications (ICC)*, 2004.
18. G. Suzuki, D. Maruyama, T. Koda, S. Aoki, T. Iwamoto, K. Takashio, and H. Tokuda: Playing with Ubiquitous Embedded Information using u-Photo. *Demo Session of International Symposium on Ubiquitous Computing Systems*, 2004.
19. G. Suzuki, D. Maruyama, T. Koda, T. Iwamoto, S. Aoki, K. Takashio, and H. Tokuda: u-Photo Tools: Photo-based Application Framework for Controlling Networked Appliances and Sensors. In *Electronic Adjunct Proceedings of The 6th International Conference on Ubiquitous Computing (UbiComp2004)*, 2004.
20. K. N. Truong, G. D. Abowd, and J. A. Brotherton: Who, What, When, Where, How: Design Issues of Capture & Access Applications. In *Proceedings of the 3rd International Conference on Ubiquitous Computing.* Springer-Verlag, 2001.
21. W. Yeong, T. Howes, and S. Kill: Lightweight Directory Access Protocol. In *Internet Request For Comments RFC 1777*, 1995.