Exploiting Eye Gaze Information for Operating Services in Home Network System

Kohei Mitsui[1], Hiroshi Igaki[2], Masahide Nakamura[1], Ken-ichi Matsumoto[1], and Kentaro Takemura[3]

[1] Graduate School of Information Science, Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0192, Japan
{kohei-m, masa-n, matumoto}@is.naist.jp

[2] Department of Information and Telecommunication Engineering, Nanzan University
27 Seirei, Seto, Aichi 489-0863, Japan
igaki@nanzan-u.ac.jp

[3] Department of Electrical Engineering, Nara National College of Technology
22 Yata, Yamatokoriyama, Nara 639-1080, Japan
kenta-ta@elec.nara-k.ac.jp

Abstract. This paper presents a system which extensively exploits user's eye gaze information for operating services and appliances in the emerging home network system (HNS). We design and implement the system called AXELLA, which captures user's gaze, then invokes a service operation, and finally announces the response via voice. AXELLA interprets the gaze information together with supplementary information as a *gaze context*, and triggers a service module associated by a service rule. Thus, a simple gazing activity can be used for various service operations. Service developers (or even home users) can easily develop context-aware HNS services with the eye-gaze-based UI. We demonstrate a practical service called "See and Know" implemented using AXELLA, where a user can acquire the current status information of every appliance just by looking at the appliance. It was shown that the proposed system can reduce the artificial dependency significantly with respect to ease-of-learning and system scalability.

1 Introduction

With the emerging ubiquitous technologies, various objects have been equipped with network functionalities. A *home network system* (HNS) is a major application of such ubiquitous technologies. In a HNS, general household appliances, such as a TV, a DVD/HDD recorder, an air-conditioner, lights, curtains, a ventilator, an electric kettle and sensors, are connected to a LAN at home, in order to provide sophisticated services for home users. It is expected in the near future that a great variety of services and appliances for a HNS will be available. Several products are already on the market (e.g., [3][13][19]). Currently, a user interface (UI) for the HNS services is supposed to be provided with a hand-held device (e.g., a handy-phone, a PDA, a proprietary remote controller, etc.) [5][10][14], or with a built-in control panel operating GUI applications [3][8][13].

However, as the number of appliances and services grows, a user will face with the problem of *artificial dependency*. Here we mean the artificial dependency by a set of factors that the user has to comply with, in order to use the HNS services[1]. For example, to use a service the user has to be familiar with usage of the hand-held device (or the control panel). The user also has to learn how to operate services on the UI, which generally varies from appliance to appliance as well as service to service. Moreover, the UI must keep up with the new appliances and services deployed. The problem is that the artificial dependency would be accumulated exponentially in the number of appliances and services, without careful consideration. The artificial dependency always exists but can be reduced. Therefore, we consider it quite important for the future HNS to minimize the artificial dependency.

In this paper, we introduce user's *eye gaze* as a UI for HNS services, which aims to reduce the artificial dependency especially in the aspects of *ease of learning (adaptability)* and *system scalability*. Specifically, we develop a system called AXELLA (Adaptive and eXtensible Environment for Legacy and Leading Appliances), which exploits user's eye gaze for operating HNS services. We assume such scenarios that a user can use HNS services easily just by "looking" at some appliances.

We first point out the following four requirements, which are unique but essential for such an eye-gaze-based UI for a HNS: (a) appliance-wise eye tracking, (b) context-aware actions, (c) switching to service mode and (d) system response with non-visual medium. Based on these requirements, we then design and implement four sub-systems for AXELLA. Specifically, as for (a), we introduce an *eye gaze analyzer* with a face tracking system, which has been developed in our previous work. For (b), we propose a context-based *service processor*. To achieve (c), we exploit a small wireless device, called *trigger button*. For (d), we select a voice-based information presentation using a text-to-speech technology.

What most original and significant in AXELLA is that it interprets the gaze information as a context. Specifically, we use supplementary information including time, location and userID together with the gaze. Upon receiving the gaze context, AXELLA activates a service module associated by a service rule. Thus, a simple gazing activity can be used for triggering various service operations. Since the service rule is written in a simple IF-THEN format, the developer (or even home user) can create and customize context-aware HNS services easily.

As a practical example, we implement an appliance presence service called "See and Know" using AXELLA. When a user looks at an appliance, the system speaks its current status, taking the current context into consideration. Through qualitative discussion, we show that the proposed system can reduce the artificial dependency significantly with respect to ease-of-learning and system scalability.

[1] The original definition of artificial dependency was given in the context of service-oriented architecture (SOA) [17].

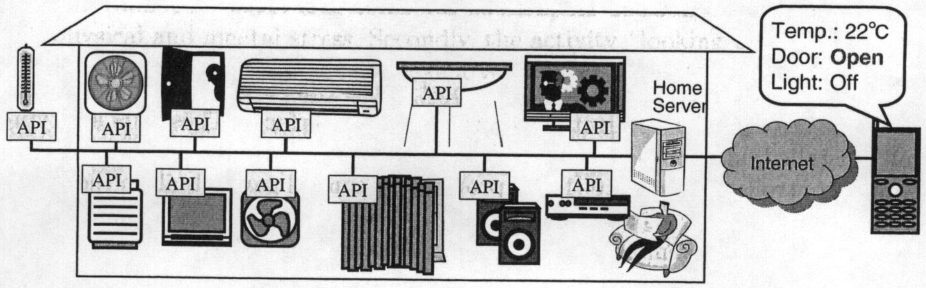


Fig. 1. Example of a home network system

2 Preliminaries

2.1 Home Network System

A home network system (HNS) consists of one or more *networked appliances* connected to a local area network at home. In general, every appliance has a set of *device control APIs*, by which users or external software agents can control the appliance via network. Figure 1 shows an example of a HNS, where various appliances are networked. In this example, a *home server* manages all the appliances in a centralized manner. It also plays a role as a residential gateway to the Internet. The communication among the appliances is performed with an underlying *HNS protocol.* Several HNS protocols are currently being standardized, such as DLNA [1], UPnP [15], ECHONET [2].

It is natural to regard each appliance in a HNS as a self-contained *distributed object* [18], since an appliance generally has an internal *state* and a set of *methods* affecting the state. For instance, let us consider a networked TV. A TV has a state consisting of *attributes* such as `power`, `channel` and `volume`. The current values of the attributes characterize the (current) state, e.g., `[power=ON, channel=2, volume=15]`. The device control APIs correspond to the methods. Typical methods (APIs) for the TV include `ON()`, `OFF()`, `selectChannel()`, `setVolume()`, `getState()`, etc. For convenience, we denote `A.m()` to represent an API invocation `m()` of an appliance `A` (e.g., `TV.setVolume(20)`, `Curtain.open()`, `DVD.getState()`).

2.2 HNS Services

A service for a HNS is basically implemented as a *software application* that executes the APIs with a pre-defined logic. For instance, invoking `getState()` API for every appliance implements an *appliance monitoring service*, The user can watch the current status of the appliances at any time and any place (see Figure 1). Another example is a *sleep service* that uses `OFF()` API for a set of registered appliances. The user can shut down all the appliances at once before leaving home or going to bed. The service applications are usually installed on the home server [6], which manages all the appliances in the HNS.

2.3 Conventional User Interfaces for HNS Services

The service application of a HNS are provided for a home user through a certain user interface (UI). As far as we know, most of the UIs currently available adopt the conventional GUI applications or Web-based interfaces. These UIs are typically installed on hand-held devices (e.g., a handy-phone, a PDA, a proprietary remote controller, etc.) [5][10][14] or proprietary control panels built in the house [3][8][13].

However, the conventional UIs impose a certain extent of the artificial dependency upon users, which is an additional cost to use the HNS services. First of all, a user is required to use the hand-held device or the control panel. Then, the user has to *learn* how to operate services on the UI. In general, such a UI varies from service to service. Therefore, when a new service becomes available, the user has to learn the operation for the new service. From the system point of view, the conventional UIs are not so adaptive (scalable) for *evolution* of a HNS. Basically the GUI is designed for known services and appliances, and not for future ones currently unknown.

As the number of services and appliances grows, the problem of the artificial dependency will become serious. Therefore, it is quite important for the future HNS to minimize the artificial dependency. This motivated us to investigate new UIs that can complement the conventional ones.

3 Exploiting Eye Gaze Information for HNS

3.1 Advantage and Drawback

To minimize the artificial dependency problem, our key idea is to introduce user's *eye gaze* as a UI for operating HNS services. More specifically, our goal is to achieve such an environment that a user can activate (or deactivate) a service, and can control or monitor appliances by just looking at the networked appliances.

The primary reason why we chose the eye gaze is that it is quite a simple and native activity for human beings. It is also known in the cognitive science that looking at an object reflects a human interest or attraction to the object [9]. Due to the above reasons, there has been much research work using the eye gaze for supporting UIs, although application to a HNS is few.

The major advantage in applying the eye gaze to the UI for HNS services is that it can reduce the artificial dependency drastically in the sense of *ease-of-learning*. This is due to the simple and intuitive nature of the eye gaze. Also, even if the number of appliances (or services) grows, the user can keep the same manner to operate with HNS services. That is, new appliances and services never affect the UI. Thus, the artificial dependency can be reduced with respect to *system scalability*. Thus, the problems caused by the conventional GUIs would be improved.

However, introduction of the eye gaze brings some new problems. Firstly, the HNS should be able to capture user's gaze with certain extent of accuracy. For this, we cannot force the user to use extra complex devices lest the user should feel physical and mental stress. Secondly, the activity "looking at an appliance" is so simple that it is difficult to make a variety of operations for the appliance solely with the gaze. Thirdly, it is necessary to identify whether the gaze at an appliance is for service operation or just for seeing the appliance. Finally, since user's eyes are occupied for the service operation, the user cannot *see* the system response simultaneously.

3.2 System Requirements

To cope with the drawbacks, we propose the following four requirements to be satisfied by the system with the eye-gaze-based UIs for operating HNS services.

Requirement R1 (appliance-wise eye tracking): The system must be able to track user's gaze on *every* appliance in a HNS. For this, no complex device should be worn by the user.

Requirement R2 (context-aware actions): The system must have a capability of mapping the gaze at the same appliance to different actions (operations). Preferably, the mapping should be performed based on the *context* derived from the gaze information.

Requirement R3 (switching to service mode): The system must have a means that a user can easily switch the system to a *service mode*, where the system accepts user's gaze as an operation to a HNS service or an appliance.

Requirement R4 (system response with non-visual medium): The system must notify a user of the response (or result) of a service operation via *non-visual medium.*

4 AXELLA — Proposed System

Based on the requirements in Section 3.2, we develop a system called AXELLA (Adaptive and eXtensible Environment for Legacy and Leading Appliances).

4.1 System Architecture

To satisfy the requirements, AXELLA is composed of the following four components (sub systems).

Eye gaze analyzer: To achieve Requirement R1, the eye gaze analyzer identifies which appliance the user is currently looking at, based on data polled from an external eye camera. Then, combining the appliance information with supplementary information (time, location, userID), the analyzer generates a *gaze context.*

Service processor: To satisfy Requirement R2, this system collects the eye gaze context polled from the eye gaze analyzer. Then, the service processor interprets the context, consults the user-defined service rules, and finally triggers an operation for the HNS appliances. The service processor also sends the response of the operation to the speech engine (See below).

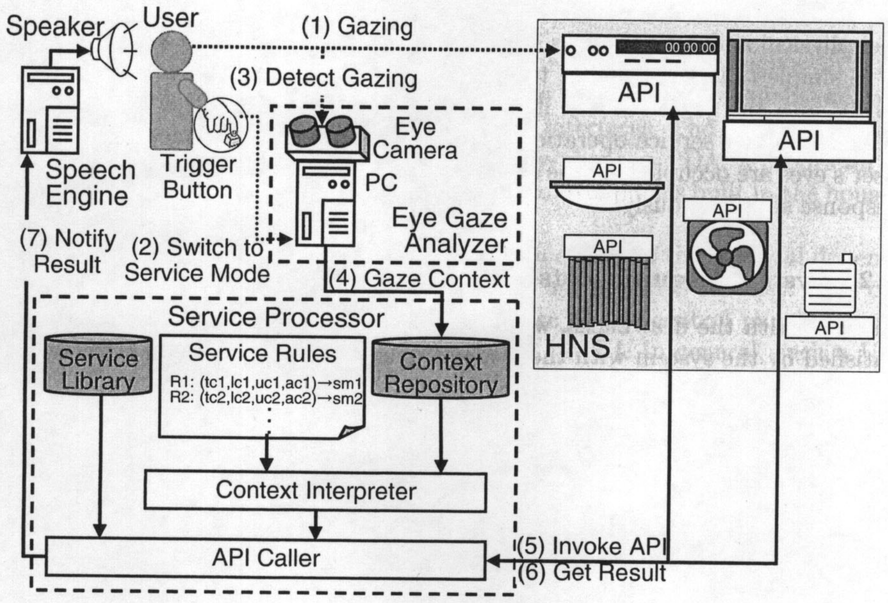


Fig. 2. Overall system architecture of AXELLA

Trigger button: This is a small wireless device for achieving Requirement R3, which is supposed to be carried by each user. When the user clicks the button, AXELLA switches to the service mode where it accepts user's eye gaze as an operation.

Speech engine: The speech engine reports the system response to the user via voice, which copes with Requirement R4. Using the text-to-speech technology, the engine dynamically synthesizes the voice from the text sent from the service processor.

Figure 2 shows overall system architecture of AXELLA. In the figure, a dotted arrow represents an interaction among a user and the system, whereas a solid arrow denotes the data flow among components. First, (1) a user gazes an appliance and (2) presses the trigger button to enter the service mode. Then, (3) the eye gaze analyzer detects the gaze and (4) sends the gaze information as a gaze context to the service processor. Upon receiving the gaze context, the service processor interprets it and chooses an appropriate service module based on a service rule. (5) The service processor invokes APIs as specified in the module, and (6) gets the results. Finally, (7) the result is sent to the speech engine to announce to the user.

For each of the four components, we give more detailed explanation in the following subsections.

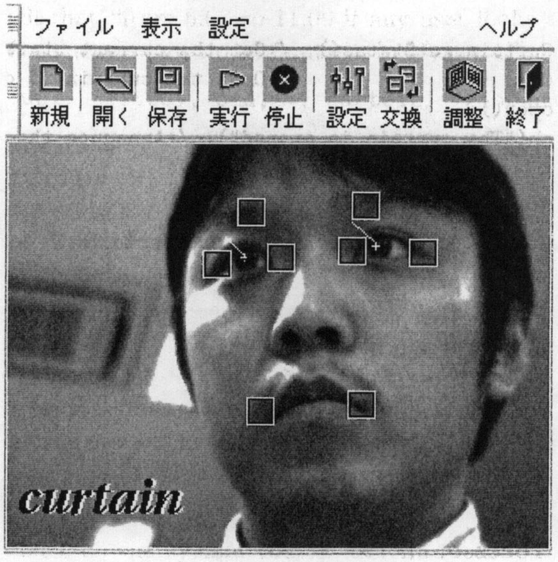

ファイル 表示 設定 ヘルプ
新規 開く 保存 実行 停止 設定 交換 調整 終了
curtain

Fig. 3. Screen shot of eye gaze analyzer

4.2 Eye Gaze Analyzer

The eye gaze analyzer captures user's eye gaze information. It identifies which appliance the user is currently looking at. To implement the analyzer, we employ the existing face-and-gaze measurement system [11]. The measurement system consists of a three-dimensional stereo camera and a PC. It captures user's face image, and performs an image processing to identify position of every facial organs. Then, the system calculates the direction and angle of eyes. Finally, the system identifies the object (i.e, appliance) by detecting the intersection of the eye direction and the object based on floor plan data prepared in advance. The system outputs the gaze information in a form of continuous bulk data. To cover a wider range of the gaze measurement, we assume to deploy multiple eye gaze analyzers in the HNS.

In this paper, we extend the measurement system so that it can capture a *context* from the gaze information. Here we define a *gaze context* as follows:

Definition (Gaze Context). Suppose that at the time t a user u in a location l keeps looking at an appliance a during a pre-defined period. Then, we define a quad-tuple $c = (t, l, u, a)$ as a *gaze context.*

A gaze context involves information on when, where and who in addition to the appliance, which is likely to reflect the user's intention and wish. Also, these attributes help the system interpret a simple eye gaze activity in different ways. For instance, a gaze context `(7:30, bed, userA, curtain)`, where user A is looking at a curtain on the bed, could characterize a wish that user A wants to

```
openCurtain {
  status s = Curtain.getStatus(); /*Get the current status of curtain*/
  if (s == CLOSE) {               /*Open the curtain if closed*/
     Curtain.open();
     TTS.speech("The curtain is opened"); /*Announce the completion*/
  }
}
```

Fig. 4. Service module `openCurtain`

open a curtain. For another context `(22:30, bed, userA, curtain)`, user A may want to close the curtain before sleeping.

The purpose of the pre-defined period in the definition is to distinguish a gaze from a *glance*. Our empirical study shows that around 1.5 seconds is its reasonable value. Figure 3 shows a screen shot of the eye gaze analyzer, where a user is currently looking at a curtain.

4.3 Service Processor

The service processor activates an appropriate HNS service upon receiving a gaze context, which plays a key role of AXELLA. It consists of five components as shown in Figure 2.

The *service library* is a collection of service modules which are reusable components to construct HNS services. More specifically, a service module is a program module that wraps invocations of appliance APIs with a certain control flow. Each service module corresponds to a self-contained action that a user performs against a HNS. Figure 4 shows an example of the service module, written in a C++-like pseudo code. This module invokes two curtain APIs to achieve user's action "open the curtain". The result is notified to the user with API of the speech engine (see Section 4.5).

A *service rule* specifies an association between a gaze context and a service module, in terms of a simple IF-THEN format.

Definition (Service Rule). A service rule r is defined as

$$r : \quad (tc, lc, uc, ac) \quad \rightarrow \quad sm$$

where tc, lc, uc, ac are *guard conditions* on time, location, user and appliance, respectively. These conditions are combined with AND semantics, which forms a *guard* over gaze contexts. A guard (tc, lc, uc, ac) is evaluated to be true or false for a given gaze context (t, l, u, a). sm is a corresponding *action*, which is given as a reference (name) to a service module. The service module sm is to be activated when the guard is satisfied.

Each of the guard conditions is supposed to be given in a logical formula. Considering that even home users can define their own service rules, our system currently supports a small set of constructs, including identifiers and operators $*$ (don't care) and `[from .. to]` (range). For instance, the following `oc1` defines a service rule such that "from 6:00 to 11:00 if any user looks at curtain on the bed, then open the curtain (with `openCurtain` in Figure 4)".

```
oc1:    ([6:00 .. 11:00], bed, *, curtain) -> openCurtain
```

Corresponding to various operations using eye gaze, the service processor can have multiple service rules.

The *context repository* is a database that stores gaze contexts polled from the eye gaze analyzer. Upon a new gaze context c arrives, the *context interpreter* looks up the service rules to find a rule whose guard is satisfied by c [2]. If a service rule is found, the context interpreter passes the name of the corresponding service module to the *API caller*. Based on the name, the API caller loads a corresponding service module from the library, and executes it. The results of the API invocations can be redirected to the speech engine to notify the user via voice.

4.4 Trigger Button

The trigger button tells the system that a user initiates to execute the HNS services by the eye gaze. The button should be small and be capable of wireless communication. We assume that every user has a trigger button with a unique signature. AXELLA recognizes the signature as a user ID, and uses it in the gaze context.

Currently, we are using a small wireless mouse as the triggering button. When the user click the button, AXELLA enters to a service mode and waits for the eye gaze operation by the user. After a preset period (currently we set it 10 seconds), AXELLA automatically reverts to the normal mode in which user's eye gaze is not interpreted as an operation.

4.5 Speech Engine

As for the non-visual medium mentioned in Requirement R4, we choose voice announcement. When the service processor passes the API results in a text format, the speech engine synthesizes the voice from the text, and plays back to the user. For the task, the speech engine exhibits a speech API to the network. With the speech API, the speech engine can be used within HNS services in a similar fashion to the ordinary networked appliances.

According to a voice-based interaction guideline [4], the text-to-speech is suitable for system output, especially when (1) a user uses eyes for interactions, (2) the user frequently moves from one place to another, or (3) the user cannot easily access the PC monitor. Thus, our application reasonably fits the guideline. Compared with visual information, information delivered by voice tends to stay longer in human memory, but its information capacity is limited. Therefore, it should be careful not to send long sentences at a time to the speech engine.

[2] Our implementation returns only the *first match* found, even if there exist multiple rules matching c. This is to avoid the *feature interaction problem* [7], which is a functional conflict among multiple service modules.

5 Implementation

5.1 AXELLA

We have implemented AXELLA using the following components.

Eye Gaze Analyzer: Dell – Optiplex (Vine Linux3.2+gcc), Pointgrey Research – Frea x 2 (for the eye camera)
Service Processor: minipc.jp CF700 (Windows XP, JDK1.5.0_06)
Trigger Button: Logicool – V200 Bluetooth Mouse
Speech Engine: minipc.jp CF700 (Windows XP, JDK1.5.0_06, Apache Tomcat5.5.12, Apache Axis1.3), PENTAX VoiceTEXT (for text-to-speech engine).

Then, we deployed AXELLA in an existing HNS developed in our previous work [12]. The HNS is composed of legacy infrared-based appliances each of which is networked with Web services. The networked appliances include a PDP (Plasma Display Panel), a DVD/HDD recorder, two lights, an air cleaner, an air circulator, a curtain, a door, a thermometer, an illuminometer, a sound-level meter, and an airflow meter.

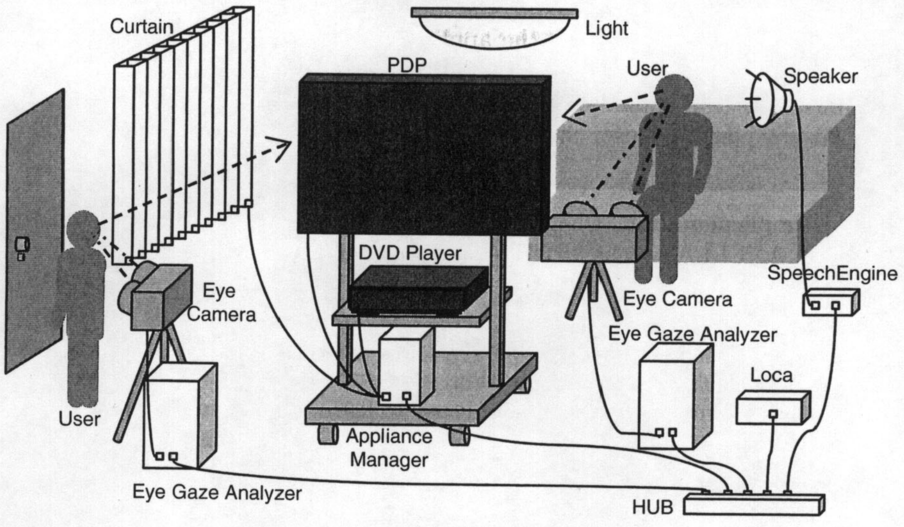
Figure 5 shows a picture and a floor plan of our experimental laboratory. As seen in the floor plan, we deployed two sets of the eye gaze analyzer, in order to capture the gaze contexts in different locations. The person in the bottom of the picture is operating HNS services using his eye gaze.

5.2 See and Know: Appliance Presence Service

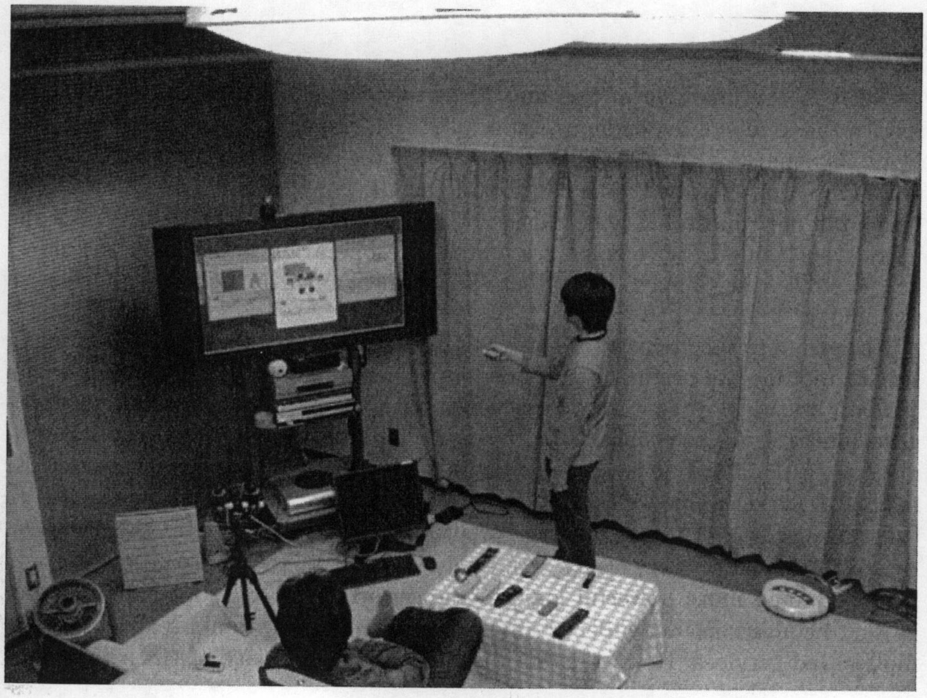
As a practical application of AXELLA, we have implemented an interesting service, named *See and Know.* This service allows a user to acquire the current state information (i.e., *presence*) of every appliance in a HNS, just by looking at the appliance.

As seen in Section 2.1, a state of each appliance generally consists of multiple attributes. However, it is not smart to present all the attributes every time, since the status information a user wants may vary depending on the context. For instance, suppose a situation that a user wants to check if a PDP is surely switched off before leaving a room. Thus, nearby the door the user is interested in only `power` attribute of the PDP. On the other hand, when looking at the same PDP at a sofa, the user would be interested in `input` attribute to check which contents the PDP is currently showing. In this case, the user does not need the `power` attribute, since the power status is obvious at the sofa in front of the PDP.

Our See and Know service achieves the *location-based* information filtering by extensively using the gaze context of AXELLA. Figure 6 shows a list of service rules implementing the See and Know service. In the figure, `notify_X_Y` represents a service module that obtains the current state of appliance `Y` through network, and then tells the status (current value) of attribute `X` via voice. In this service, we assume the following use case scenarios.



(a) Floor plan



(b) Picture

Fig. 5. Our experimental laboratory for HNS and AXELLA

UC1: On the sofa in a living room, the user checks a working status specific to each appliance by looking at the appliance.

UC2: Near the exit door, the user checks power status for each appliance, to confirm that the appliance is surely turned off before leaving the room.

UC3: At any place, the user checks temperature upon looking at the thermometer.

In this implementation, when a user gazes at the PDP at the door and the PDP is off, AXELLA speaks "PDP is now power off". This is as specified in `sk6` to achieve the use case UC2. Another example is that AXELLA speaks "DVD recorder still have 40GB of free space" when the user looks at the DVD recorder on the sofa. This is by `sk2` involved in the use case UC1.

Through a simple usability evaluation, we have confirmed that See and Know service was quite intuitive, efficient, and easy to learn. In a cognitive sense, gazing an object and checking the object are tightly coupled with each other. Therefore, few artificial dependencies were imposed to the user. With the conventional UIs, the user would have taken more time and expertise to use the same service.

6 Discussion

6.1 Qualitative Evaluation

As seen in the example of See and Know service, AXELLA can significantly reduce the artificial dependency, especially with respect to the ease-of-learning. Note also that AXELLA is able to cope with the addition of new services and appliances quite easily, in the context of the UI. For this, the service developer just adds new lines of service rules, without re-engineering the UI. As for the user, even if new service rules are added, the operation is basically the same, i.e., looking at appliances. This fact shows that AXELLA achieves the system scalability.

According to the architecture of AXELLA (see Figure 2), a gaze context and a service module are originally independent. Therefore, the service developer can develop and deploy new service modules without assuming any link to user's gaze contexts. A gaze context and a service module are connected explicitly by a service rule. Thus, it is actually the service rules that implement a concrete AXELLA service. Due to its simple syntax, we believe that even home users can customize existing rules or define new rules. This fact implies that the users can create their own services easily.

Introducing more powerful logic in the service rules (e.g., a temporal order in guard conditions, dependency among service modules) would realize more sophisticated services. However, this increases the complexity of the service rule description, which is a trade-off against the easy service creation. We will investigate this issue in our future work.

A limitation in our current implementation is that gaze contexts can be captured only in the position where the eye camera is placed. This is due to the limitation of the eye camera of the eye gaze analyzer. To cover a wider range,

```
# See and Know Service Rules
# rule: (When, Where, Who, Appliance) -> service_module
# at sofa in living room...
sk1:  (*, sofa, *, PDP)         -> notify_input_PDP
sk2:  (*, sofa, *, DVD)         -> notify_remainingfreespace_DVD
sk3:  (*, sofa, *, LIGHT)       -> notify_brightness_LIGHT
sk4:  (*, sofa, *, AIRCLEANER) -> notify_drivemode_AIRCLEANER
sk5:  (*, sofa, *, CIRCULATOR) -> notify_flowlevel_CIRCULATOR
# at exit door...
sk6:  (*, door, *, PDP)         -> notify_power_PDP
sk7:  (*, door, *, DVD)         -> notify_power_DVD
sk8:  (*, door, *, LIGHT)       -> notify_power_LIGHT
sk9:  (*, door, *, AIRCLEANER) -> notify_power_AIRCLEANER
sk10: (*, door, *, CIRCULATOR) -> notify_power_CIRCULATOR
# for temperature
sk11: (*, *, *, THERMOMETER)   -> notify_temperature_THERMOMETER
```

Fig. 6. Service rules for See and Know service

we have to deploy many cameras in the room. However, we believe that this problem will likely be alleviated as the technology matures.

6.2 Comparison to Conventional UIs

We do *not* consider that AXELLA with the eye-gaze-based UI completely replaces the conventional UIs for a HNS. The point is that it is important to make an appropriate choice of AXELLA and/or the conventional GUIs, taking the *nature* of target services into account.

According to its eye-gaze-based UI, we consider that AXELLA is well suitable for implementing HNS services that have the following characteristics:

A1: Daily routines or operations frequently used (e.g., open/close curtains, switch on/off lights).

A2: Simple operations over multiple appliances (e.g., turn on and play TV and DVD together, group shutdown)

A3: Information presentation intuitively linked to gazing (e.g., See and Know).

On the other hand, due to the simple and intuitive nature of gazing, AXELLA is not good for dealing with the following kinds of services.

B1: Services that requires many parameter inputs and stateful operations (e.g., information search, HDD recorder setting).

B2: Operations not intuitively connected to concrete appliances (e.g., showing stock prices).

For these services, we consider it difficult to achieve solely with AXELLA. It should be complemented by the conventional or new other UIs. Further investigation on suitable applications of AXELLA is left for our future work.

6.3 Related Work

There have been a few existing research that uses the eye gaze to control appliances, although they are not for the purpose of HNS services. Takemura et al. proposed an application called *universal remote controller* [11]. In this application, the eye gaze is basically used to *select* an appliance that a user wants to control. Actual operations are issued by a user on a PC using a mouse or a keyboard. Since the user cannot see the PC monitor while gazing, the operations are limited to relatively simple ones. Also, it does not consider the gaze contexts or the link from a gaze to any service operation, as AXELLA does.

Vertegaal et al. developed a series of appliances called *media eyepliances* [16]. Each eyeplliance provides features strongly coupled with the gazing activity of human beings. For instance, an eyeplliance, called Attentive Television, plays the contents only while a user is looking at the TV. Otherwise TV pauses the contents. The approach is quite device-centric, and a relationship between a gaze and an action is fixed. This is quite different from our service-centric approach.

7 Conclusion

In this paper, we proposed a system AXELLA, which exploits user's eye gaze information extensively for services in the future home network systems (HNS). We first investigated system requirements for eye-gaze-based UIs for the HNS services. Then, based on the requirements, we designed and implemented four components: the eye gaze analyzer, the service processor, the trigger button, the speech engine. Introducing a notion of the gaze context, AXELLA allows service developers (or even home users) to create context-aware services with the eye gaze information. As a practical application, we have implemented a practical service – See and Know. Finally, we discussed and evaluated the proposed system to see its advantage and limitation.

Our future work is to analyze and enhance the expressive power of the service rule description. We also plan to clarify the service domain that makes the best use of AXELLA. Investigating human activities other than eye gaze for HNS services is also an interesting issue.

Acknowledgement

This research was partially supported by: the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B) (No.18700062), Grant-in-Aid for 21st century COE Research (NAIST-IS — Ubiquitous Networked Media Computing), and Pache Research Subsidy I-A-2,2006 of Nanzan University.

References

1. Digital Living Network Alliance
`http://www.dlna.org/home/.`
2. ECHONET Consortium
`http://www.echonet.gr.jp/english/.`
3. Matsushita Electric Industrial Co., Ltd. kurashi-net:`http://national.jp/appliance/product/kurashi-net/.` (in Japanese)
4. Mayhew, D.J.:"Principles and Guidelines in Software User Interface Design," Englewood Cliffs NJ: Prentice-Hall
5. NANO Media Inc., "App-rimo-con",
http://www.nanomedia.jp/english/service/s02.html
6. OSGi Alliance
`http://www.osgi.org/.`
7. Reiff-Marganiec, S., Ryan, M.D., editors., Proceedings of 8th International Conference on Feature Interactions in Telecommunications and Software Systems. IOS Press. 2005
8. Rich, C., Sidner, C., Lesh, N., Garland, A., Booth, S., Chimani, M.: "Diamond-Help: A New Interaction Design for Networked Home Appliances," Personal and Ubiquitous Computing, Springer-Verlagm ISSN: 1617-4909, Vol. 10, Issue 2, pp. 187-190, January 2006.
9. Selker, T.:"Visual attentive interfaces," BT Technology Journal, Volume 22, No 4, pp.146-150, Oct. 2004.
10. SUGIYAMA ELECTRON Co., Ltd. Remocon Sauser:
`http://www.sugi-ele.co.jp/remoconsaucer.htm.`
11. Takemura, K., Minamide, H., Matsumoto, Y., Ogasawara, T.: "What You Look at Is What You Control:A Universal Remote Control Based on Gaze Measurement Technology," 1st IEEE Technical Exhibition Based Conference on Robotics and Automation (TExCRA2004), 2004
12. Tanaka, A., Nakamura, M., Igaki, H., Matsumoto, K.: "Adapting Conventional Home Appliances to Home Network Systems Using Web Services," Technical Report of IEICE, IN, 2005. in Japanese
13. Toshiba Consumer Marketing Corp., "FEMINITY" :
`http://www3.toshiba.co.jp/feminity/feminity_eng/.`
14. Toshiba Consumer Marketing Corp., "Life Remote Controller" :
`http://www3.toshiba.co.jp/feminity/feminity_eng/keitai/keitai_01.html.`
15. UPnP Forum
`http://www.upnp.org/.`
16. Vertegaal, R., Cheng, D., Sohn, C., Mamuji, A.: "Media EyePliances: Using Eye Tracking For Remote Control Focus Selection of Appliances," In Extended Abstract of ACM CHI 2005 Conference on Human Factors in Computing Systems. Portland, OR: ACM Press, 2005.
17. W3C "Web Services Glossary" :
`http://www.w3.org/TR/ws-gloss/.`
18. W3C "Web Services Architecture" :
`http://www.w3.org/TR/ws-arch/.`
19. ZOJIRUSHI CORPORATION "iPot" :
`http://www.mimamori.net/.`