2012 19th Asia-Pacific Software Engineering Conference

Implementing Virtual Agent as an Interface for Smart Home Voice Control

Shimpei Soda, Masahide Nakamura, Shinsuke Matsumoto,
Shintaro Izumi, Hiroshi Kawaguchi, Masahiko Yoshimoto

Graduate School of System Informatics, Kobe University
1-1 Rokkodai, Nada, Kobe, Hyogo, 657-8501 Japan
soda@ws.cs.kobe-u.ac.jp

***Abstract*—We have been developing a hands-free voice controller for a *home network system* (HNS) by using microphone arrays. In our current implementation, however, all human-HNS interactions are performed by voice only. Hence, the interactions tend to be mechanical, dreary and uninformative. To achieve richer interactions, we try to introduce the virtual agent technology as a feedback interface of the HNS. In this paper, we implement the virtual agent as a Web service, by using MMDAgent Toolkit extensively. The agent is then integrated with the HNS and microphone arrays in a service-oriented fashion. Finally, we conduct a user experiment with three versions of virtual agents. In the experiment, we evaluate how the virtual agent can enrich the interactions.**

***Index Terms*—smart home, voice interface, hands free, MMDAgent, microphone array**

I. Introduction

The *home network system* (HNS) is a core technology of the next-generation smart house, achieving value-added services by networking various household appliances and sensors [1]. Since a variety of appliances and services are deployed, intuitive and easy-to-learn human interface to control the HNS is required. *Voice interface* is one of promising technologies for such a universal controller. In our ongoing project [2], we are developing a *hands-free voice controller* for our HNS. Using a microphone array built in a ceiling, a user can operate appliances and services by just speaking to the house. The HNS recognizes the speech, invokes a corresponding HNS operation, and returns a feedback voice. However, all interactions between the user and the HNS are performed by the voice only. Thus, the interactions tend to be mechanical, dreary and uninformative.

In order to achieve richer interactions, we consider to employ the *virtual agent* technology [3][4], which can introduce affinity and humanity in spoken dialog systems. By integrating a virtual agent with our hands-free controller, we expect that a user can enjoy operating the HNS through more natural conversations with the agent.

In this paper, we implement a virtual agent as an interface of our HNS hands-free voice controller. For this, we extensively use *MMDAgent* [5], which is an open-source toolkit to implement virtual agents for spoken dialogue systems. We re-engineer MMDAgent so that it can invoke Web services. We then isolate the virtual agent from the voice recognizer, and deploy the agent as a Web service. Thus, the microphone array, the voice recognizer, the virtual agent and the HNS are integrated in a service-oriented fashion via REST Web-API.

Finally, we conduct a user experiment, where each subject operates actual HNS appliances by speaking to three versions of virtual agents. In the experiment, we investigate how the virtual agents affect human behaviors and impressions in smart house environment.

II. Preliminaries

A. Home Network System (HNS)

The appliances and sensors are connected via a network. Each device has control API to allow users or external agents to control the device over the network. Application services include personal home controllers, remote monitoring/controls, appliance orchestration, energy saving, context-aware services, etc.

In our research group, we have implemented an actual HNS environment, called CS27-HNS. Introducing the concept of service-oriented architecture (SOA) [6], CS27-HNS integrates heterogeneous and multi-vendor appliances by standard Web services. Since every API can be executed by SOAP or REST Web service protocols, it does not depend on a specific vendor or execution platform.

B. Hands-Free Voice Controller Using Microphone Array

The *voice interface* is a promising technology to implement a universal controller of the HNS, since it can abstract heterogeneous operations in terms of speech. Most conventional voice interfaces require to use close-talking microphones (e.g., ones with headsets or smartphones). However, carrying always such microphone devices everywhere in the house burdens significant constraint in the daily life.

To cope with the problem, we are developing a *hands-free voice controller with a microphone array* [2], built in a ceiling of CS27-HNS. A microphone array is a device, consisting of multiple microphones arranged in a grid form. Using time difference of sound arriving to each microphone, the array can estimate the direction of sound source, and can separate multiple sound sources [7] [8]. This allows users to speak from

1530-1362/12 $26.00 © 2012 IEEE
DOI 10.1109/APSEC.2012.39

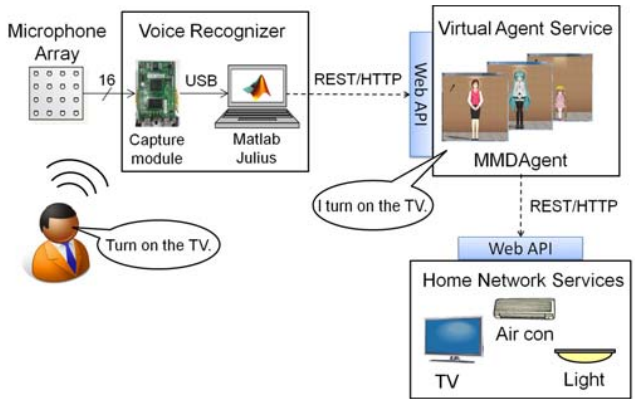

Fig. 1. Overview of interactive hands-free voice interface.

everywhere without being aware of microphones, and achieves good quality of voice sampling in noisy home environment.

C. Challenges to Richer Voice Interactions

In the current implementation of our hands-free home controller, all the interactions are performed by voice only. This yields the following problems in practice.

- It is unusual for home users to speak to something invisible, which prevents the users from using the voice controller spontaneously.
- The only way for the HNS to deliver the status and information is voice, which is often uninformative.
- Long system feedback often leads to mis-recognition of the microphone array.

To cope with the above problems, we need multimodal interaction combining the voice and something else. Towards the richer interactions, we especially focus on the use of virtual agent in this paper.

III. Implementing Virtual Agents for Home Controller

A. System Overview

Figure 1 shows an overview of the proposed system. First, 16 channels of voice sampled from a microphone array is input to the *voice recognizer*. The voice recognizer analyzes the sampled voice using MatLab, and aggregates the 16ch into 1ch of high-quality voice. This process is performed by an algorithm proposed in [2]. The voice is then redirected to a voice recognition engine, Julius [9]. Based on the recognition result, the voice recognizer generates a command for the virtual agent, and sends the command to the *virtual agent service*.

On receiving a command from the voice recognizer, the virtual agent service responds a feedback voice with a choreography, and executes a corresponding HNS operation. Thus, it works as an interface between the user and the HNS. We implement the virtual agent service by re-engineering *MMDAgent* [5], extensively. Note that the virtual agent service is loosely coupled with the system via Web-API. This allows us to reuse the virtual agents for other systems easily.

B. MMDAgent

MMDAgent is an open-source toolkit to implement virtual agents for spoken dialog systems [5]. To represent the virtual agents, MMDAgent uses choreographed 3D models created by *MikuMikuDance*, very popular software in Japanese subculture. We can produce various virtual agents by changing the 3D models.

The choreography of the agent is defined by a *script*, which specifies motion, movement, camera work, speech synthesis, speech recognition, lip sync, application invocation, according to a state transition model with events and actions. The original MMDAgent provides a ready-made framework, using Julius as its voice recognition engine, and Open JTalk as voice synthesize engine. It also assumes a direct voice input (via microphone) and local application invocation. We need to re-engineer the original MMDAgent to fit it to our system configuration.

C. Implementing Virtual Agent Service

We implement the virtual agent service by re-engineering MMDAgent. The point of the re-engineering is to extract the core feature of the virtual agent as a self-contained service. More specifically, we conduct the following modifications:

Isolation of Event Handling:

The original MMDAgent handles every event by itself, and a corresponding command is passed to the virtual agent, internally. We modify the code so that external software can send a command directly to the agent. This isolates the event handling process from the agent behaviors. So we can use our own voice recognizer (with the microphone array) as the front-end of the virtual agent.

Invocation of Web Service:

The original MMDAgent has a command that the virtual agent invokes internal command-line applications. We add an extra command that can execute external Web services (with REST protocol). The new command allows the agent to trigger our CS27-HNS (see Section II-A). Moreover, it facilitates integration with any other service-oriented systems. For example, we can integrate an external voice synthesis service, instead of using built-in voice synthesizer.

Deployment as Web Service:

Finally, we determine the API of the modified version of MMDAgent, and deploy it as a Web service. Major API includes `sendCommand()` that sends a command to a specified virtual agent, `sendEvent()` that emits an event to the agents. Thus, the virtual agent can be used as a service from various applications in a platform-independent manner.

D. Dialog Design for Controlling HNS

Figure 2 shows a flowchart representing a dialog flow of the proposed system. In the figure, a rectangle shows a process, a diamond represents a condition, an oval balloon represents a speech of a user, and a rectangle balloon represents a speech of an agent.

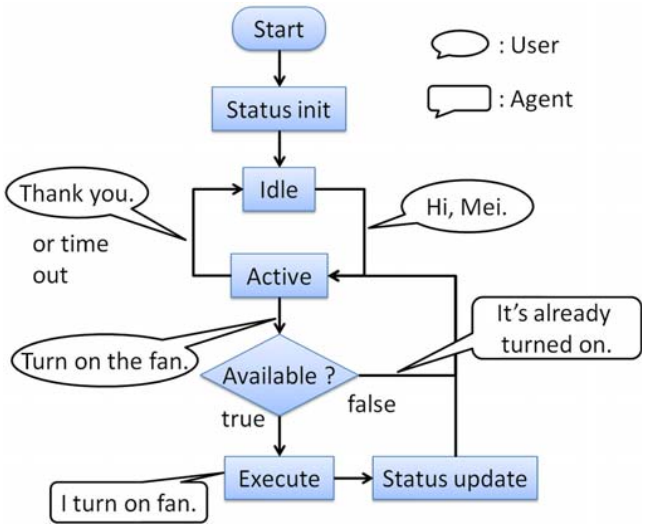

Fig. 2. Dialog flow of the proposed system

On executing the system, the agent first collects the current status of every appliance via `getStatus()` API of HNS. For a virtual agent, we define two kinds of states: *idle* and *active*. In the idle state, an agent ignores all the input voice but his/her name. When a user calls the name of the agent ("Hi, Mei." in this example), the agent moves to the active state, and becomes ready to accept the voice command. The agent in the active state goes back to the idle state when a user says "Thank you" or as a timer expires.

When the agent receives a voice command (e.g., "Turn on the fan") in the active state, the agent checks if the command is available on the current status of the appliance. If available, the agent executes a corresponding command with a feedback voice ("I turn on the fan"). Finally, the agent updates the appliance status. If the command is not available (e.g., the case where the fan is already turned on), the agent returns the action of failure.

E. Implemented Virtual Agents

Using the proposed system, we have implemented three kinds of virtual agents: *Ai, Miku, and Mei*, as shown in Figure 3(a). All the three agents work with the common logic, but have different appearance and voice. Ai is a little child speaking in slow and childish voice. Miku is a girl speaking in a brisk voice. Mei is a lady speaking a calm voice. The 3D models for Ai, Miku and Mei have been borrowed from [10] [11] [5], respectively.

Figure 3(b) enumerates the voice commands currently available. The row #2 represents the commands to activate an agent, while the row #15 is the one to de-activate the agent. The rows from #3 to #14 are commands for operating appliances within CS27-HNS. The rows #1 and #16 are commands for simple greetings. The supported language is currently Japanese only. So, all the command should be spoken in Japanese.

No	Command (Japanese)	Command (English)
1	ただいま	I'm home.
2	アイちゃん ミクちゃん メイちゃん	Ai-chan. Miku-chan. Mei-chan.
3	カーテン閉めて	Close the curtain.
4	カーテン開けて	Open the curtain.
5	テレビ消して	Turn off the TV.
6	テレビつけて	Turn on the TV.
7	扇風機つけて	Turn on the fan.
8	扇風機消して	Turn off the fan.
9	ライトつけて	Turn on the light.
10	ライト消して	Turn off the light.
11	エアコンつけて	Turn on the air con.
12	エアコン消して	Turn off the air con.
13	全部つけて	Turn on the all.
14	全部消して	Turn off the all.
15	ありがとう	Thank you.
16	行ってきます	I'm going.

(b)

Fig. 3. (a) Virtual agents implemented. (b) List of available commands.

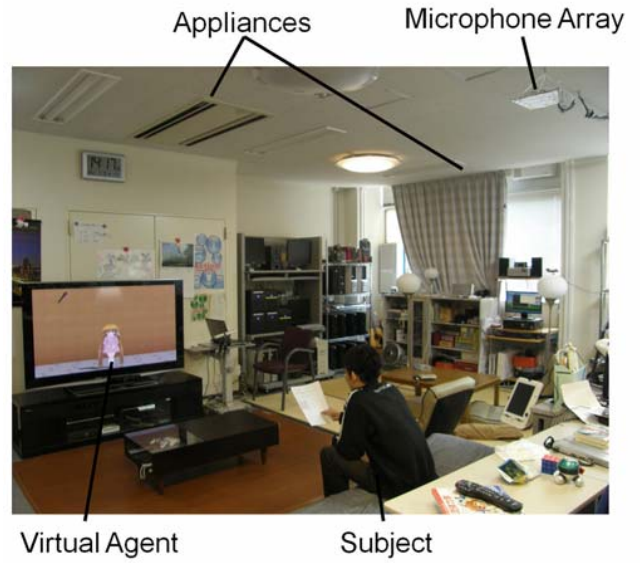

Fig. 4. Experiment environment.

IV. Experimental Evaluation

A. Experiment

To evaluate the developed system, we have conducted an experiment with six subjects. We asked each subject to perform the following two tasks.

- Task T1: A subject speaks each command in Figure 3(b) to a virtual agent. If the agent failed to recognize the command, the subject repeats the same command up to three times. This task is to evaluate the practical feasibility of the proposed system.
- Task T2: A subject freely interacts with each of the three agents, until the subject achieves five correct HNS operations. This task is to see how the appearance of the agent influences human behaviors.

Figure 4 shows a scene of the experiment in our CS27-HNS experimental room. A subject is sitting on a sofa, interacting with a virtual agent displayed in a TV. The microphone array is deployed in the ceiling. When a user speaks a command,

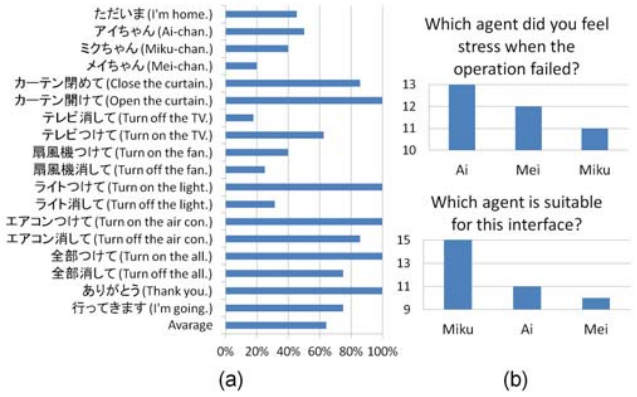

Fig. 5. (a) T1: Recognition rate of each command. (b) T2: Difference of impression in each agent.

an appliance actually performs a corresponding operation.

B. Results

Figure 5(a) shows the results of the T1. The vertical axis represents the commands. The horizontal axis shows the accuracy of the speech recognition. The total average of the accuracy was 64%, however, the accuracy varies a lot depending on the commands. Investigating the commands with low accuracy, we have found that commands which are similar pronunciation induced mis-recognition. For instance, "senpuuki (fan)" was often mis-recognized as "zenbu (all)". To obtain sufficient accuracy for practical use, it is necessary to design the dialog model that does not overlap similar words. It is also interesting to use other language (e.g., English) and compare the accuracy.

Figure 5(b) shows the result of subsequent questionnaires of Task T2. In each question, the three agents were ranked by every subject. The graphs plot the weighted sum of the rank. In the first question, no significant difference was seen among the three agents. Thus, in this experiment, appearance of agent did not affect much human feelings against failed jobs. From the second question, it was interesting to see that the favorite agent was chosen by individual preference to the character, not by the performance of the agent. All six subjects agreed with having virtual agents for the HNS voice control. They gave us the following comments.

Strength:

- Agents reduce mental resistance to speak to appliances.
- Agents are helpful to grasp timing of saying commands.
- Agents gives quick visualization of the system response.

To be improved:

- More variety of motions would make agents more attractive and more likely to human.
- Services in which the agent takes initiative would be useful (e.g., instruction of appliances usage).

In the experiment, it was shown that virtual agents could give a certain extent of affinity to the voice interactions that were previously dreary and uninformative. As for further improvement, subjects pointed out that the motions of the agent were not entirely satisfactory. Indeed, the common motions were used for all three agents in our current implementation, which might be reflected the result of Figure 5(b).

To achieve more natural interaction, it is also necessary to implement not only appliance operation but also conversation carried out in everyday life. We leave these issues our future work.

V. Conclusion

In this paper, we have implemented a virtual agent as an interface of our HNS hands-free voice controller. We have also conducted an experiment with three virtual agents. In the result, the overall average recognition rate of input commands was 64%. It was shown that virtual agents can give affinity to the inorganic interface and promote natural dialogue to the user. The future work includes enhancement of the dialog model, and development of attractive motions of agents.

VI. Acknowledgments

This research was partially supported by the Semiconductor Technology Academic Research Center (STARC), the Japan Ministry of Education, Science, Sports, and Culture [Grant-in-Aid for Scientific Research (C) (No.24500079), Scientific Research (B) (No.23300009)], and Kansai Research Foundation for technology promotion.

References

[1] Masahide Nakamura, Akihiro Tanaka, Hiroshi Igaki, Haruaki Tamada, and Kenichi Matsumoto, "Constructing home network systems and integrated services using legacy home appliances and web services," *International Journal of Web Services Research*, vol. 5, no. 1, pp. 82–98, 2008.

[2] Shimpei Soda, Shinsuke Matsumoto, Masahide Nakamura, Shintaro Izumi, Hiroshi Kawaguchi, and Masahiko Yoshimoto, "A feasibility study of home services using a microphone array network," in *Techinical Report of IEICE*, March 2012, vol. 111, pp. 73–78, (in Japanese).

[3] Magalie Ochs, Catherine Pelachaud, and David Sadek, "An empathic virtual dialog agent to improve human-machine interaction," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1*, 2008, pp. 89–96.

[4] J. Cassell, "Embodied conversational interface agents," *Communications of the ACM*, vol. 43, no. 4, pp. 70–78, 2000.

[5] MMDAgent Project Team, "Mmdagent - toolkit for building voice interaction systems," Nagoya Institute of Technology, http://www.mmdagent.jp.

[6] M. P. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Communication of the ACM*, vol. 46, no. 10, pp. 25–28, 2003.

[7] Shintaro Izumi, Hiroki Noguchi, Tomoya Takagi, Koji Kugata, Shimpei Soda, Masahiko Yoshimoto, and Hiroshi Kawaguchi, "Data aggregation protocol for multiple sound sources acquisition with microphone array network," in *20th International Conference on Computer Communications and Networks (ICCCN)*, 2011, pp. 1–6.

[8] Koji Kugata, Tomoya Takagi, Hiroki Noguchi, Masahiko Yoshimoto, and Hiroshi Kawaguchi, "Intelligent ubiquitous sensor network for sound acquisition," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 585–588.

[9] Julius Development Team, "Open-source large vocabulary CSR engine Julius," http://julius.sourceforge.jp/en_index.php?q=index-en.html.

[10] Tarunano, "Tsukuyomi ai version 1.05," http://mikudance.info/models/original/links.html.

[11] Lat, "Type lat miku version 2.3 white," http://mikumikudance.wikia.com/wiki/Miku_Hatsune_%28Lat%29.