A Stochastic Approach for Creating Context-aware Services based on Context Histories in Smart Home

Hua Si, Yoshihiro Kawahara, Hiroyuki Morikawa, Tomonori Aoyama
Aoyama Morikawa Lab, Hongo Campus Bldg #3.
Department of Information and Communication Engineering, Faculty of Engineering,
The University of Tokyo 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656 JAPAN
+81 3 5841 6710
{sihua, kawahara, mori, aoyama}@mlab.t.u-tokyo.ac.jp

ABSTRACT
In this paper, we present a context-aware service platform called Synapse and its simple smart home test bed. By exploiting the recorded histories of contexts and services, Synapse can learn different users' habits. Then Synapse can predict the most relevant services that users will use in the current situation based on their habits, and provide services in Active Mode and Passive Mode. Considering the challenges faced, we apply a stochastic approach – Bayesian Networks [17] to build the model of Synapse, and implement a flexible, end-user manageable system, which can absorb various uncertainties from multi-dimensional sensor data and provide personalized services.

Keywords
Context-aware service, HMM (Hidden Markov model), context histories, smart home

INTRODUCTION
Context-awareness is now regarded as a key ingredient for pervasive computing, and several toolkits such as Context Toolkit [21] and Location Stack [4] have been proposed to incorporate users' contexts into network applications. An attractive category of context-aware applications is the service automation in the indoor environment such as home and office [15], which aims at providing users with dynamic services that adapt to changing environment on the basis of users' habits. Obviously, the best ground for learning users' habits exists in the recorded histories of the users' interactions in context (context histories for short). The usefulness of location history has been explored to report users' mobility patterns in an office [13]. However, in the real world applications, users' contexts contain diversity, from users' activities to environmental status; and users' habits vary widely, from the usage of services to the mobility patterns. Therefore, a dynamic mechanism is necessary, which can provide various services by taking the users' contexts into account. We are developing a context-aware service platform – Synapse, which can learn different users' habits by exploiting the recorded histories of contexts and services, then predict and provide the most relevant services that users will use in the current situation based on their habits. We are implementing a smart home test bed of Synapse, and three simple scenarios are used to examine the practicability of our methods.

The following "daily scenarios" are assumed to be learned from users' context histories, and are used for evaluation:

- **The "Light" Scenario**: if it is too dark in the room, Synapse will automatically turn on the light.
- **The "TV" Scenario**: Synapse will recommend TV programs appropriate for people in the living room. (If only kids are watching TV, cartoon videos will be recommended. When parents and kids are watching TV together, Discovery or some other channels will be recommended.)
- **The "Music" Scenario**: Synapse will automatically turn down the volume of the music player when someone is using the phone, and turn up the volume after using it.

We faced several challenges when we designed our system. First, considering the flexibility of system and the ease of management for end-users, we should apply a dynamic mechanism rather than binding the contexts and services in a specification language such as ECA [12]. Second, since users' habits may slowly change as time advances, our algorithms should have the ability of updating to reflect it. Third, corresponding to the diverse contexts (such as "the user is sitting", "the brightness in a room") and various services (such as "turn on light", "select TV channel 3"), our model should have the capability to deal with multi-dimensional inputs and outputs. Fourth, personalized services are desired by different users. Finally, the system should work with imperfect and noisy sensor data.

With these challenges in mind, we apply a stochastic approach for Synapse, which is based on one of Bayesian Networks [17] – HMM (Hidden Markov Model) [18]. The model of Synapse consists of continuous cycles. Each cycle is composed of two phases: Learning Phase and Executing Phase. In the Learning Phase, Synapse learns the relationship between contexts (we call them "sensor events" in Synapse) and services by exploiting the recorded histories of them. Then in the Executing Phase, based on the learned relationship and the current sensor events, Synapse predicts the most possible services to be used and provides them to users. Since users would like to enjoy autonomous services in a moderate degree without losing control of them [2], Synapse provides services in two modes: Active Mode will start a service automatically based on sensor events, while Passive Mode recommends the top 5 relevant services in a list and let users select. The results of the Learning Phase are used as prior knowledge for the next cycle. To easily achieve personalization, user ID is treated as a sensor event.

The related works of time-series prediction and smart home projects will be introduced in section 2. The architecture of Synapse will be explained in section 3. The preliminary evaluation of Synapse will be discussed in section 4. The conclusion and future work will be given in section 5.

RELATED WORKS

For time-series prediction of continuous data, linear models (such as ARIMA, ARMAX [5]) or non-linear models (such as neural networks or decision trees [14]) are usually used. For discrete data, n-gram models [8] or variable-length Markov models [20] are common choices. Compared to these methods, Dynamic Bayesian Networks (DBN) [17] have some advantages appropriate for the challenges we face: First, it is easier for DBN to deal with multi-dimensional inputs and outputs. Second, prior knowledge is easy to be incorporated, so the prediction of the future is based on all the past history. Third, DBN is more flexible than simple supervised classifiers. Finally, DBN has been successfully used in many areas [6, 11, 19] for time-series prediction.

Therefore, we choose HMM [18], one of Bayesian Networks, to build the core model of Synapse. This core model is a general context-aware platform, which can be used not only in smart home environment, but also in a broad range of context-aware applications that need to correlate the contexts and services, since it provides standard interfaces for contexts and services.

Several smart home projects are in progress. The Georgia Tech Aware Home [1] and MIT House_n [7] use an array of sensors to determine users' locations and activities within an actual house. The Neural Network House [16] balances the goals of anticipating user needs and energy conservation through a neural network. The MavHome [3] uses an intelligent and versatile home agent to perceive the state of the home through sensors and act on the environment through effectors. The industrial examples are also available, such as the Microsoft Easy Living project, the Cisco Internet Home, and the Verizon Connected Family project. Although, similar with these projects, our smart home test bed of Synapse extracts contexts from raw sensor data and adopts services from smart devices, our original core model guarantees the uniqueness of Synapse.

ARCHITECTURE OF SYNAPSE

The smart home test bed of Synapse consists of four parts: 1) the sensor event collection part that captures real world information, 2) the service control part that provides services, 3) the Synapse Core, and 4) the user interface. Architecture of Synapse is shown in Figure 1.

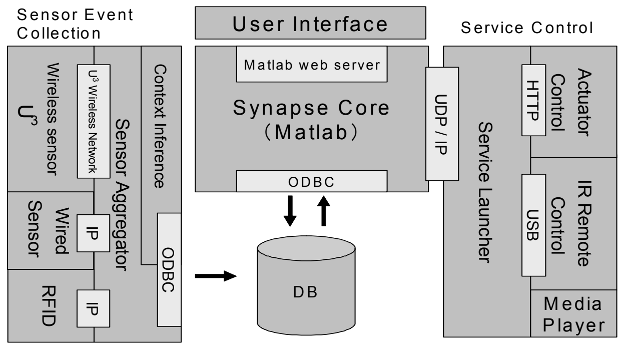

Fig. 1: Architecture of Synapse.

Sensor Event Collection Part

The sensor event collection part captures real world information from various sensors, converts raw data into useful contexts (we call them "sensor events" in Synapse), and records these sensor events in database. Sensor Aggregator fuses the raw sensor data and reduces the noise. For instance, the average temperature in a room is fused from different temperature sensors. Context Inference extracts complex events such as "the user is sleeping" from simple events. On this test bed, 4 kinds of sensors are used to produce 11 events: RFID is used to identify users, U^3 wireless sensor nodes [9] are used to capture the temperature, brightness and human motion, a contact detector detects whether the phone is in use, and an e-calendar detects a day of the week.

All the sensor events are recorded as a time series $\{E_1, E_2 \ldots\}$ in database. Each sensor event is recorded as E (EN, EV, ET), which respectively represents the event ID, the event value and the time at which this event is recorded. We predefine a set of events $\{e_1, e_2 \ldots e_N\}$ (such as e_1 means "temperature", e_2 means "brightness"), and EN $\in \{e_1, e_2 \ldots e_N\}$. Many context inference schemes can be used to recognize events and their values from raw sensor data [10]. However, since event values are generated from different types of sensors (e.g. the temperature is 25^oC, and the humidity is 60%), and it is difficult for a general core to process all types of values, we use fuzzy sets [22] approaches to unify all the event values between 0 and 1 as in [10], which means EV $\in [0, 1]$. Basically, an event will be recorded when the value changes. However, in many scenarios, it is not necessary to record events as frequently as they change, so we can add some requirements to event recording. Events will not be recorded, until they satisfy these requirements. (e.g. one requirement is "e_1 is over 0.7", so e_1 will not be recorded until it is over 0.7.)

Service Control Part

The service control part controls various devices to supply services. Service Launcher operates as a proxy between Synapse Core and the devices. It can receive a service ID from Synapse Core through UDP/IP networks, and controls the device corresponding to this service ID. It can also send the ID of a selected service to Synapse Core for service recording. As a result, it is easy for Synapse to add new services, since Synapse Core can manage them with only IDs, and ignore the various operations of different devices. On this test bed, 4 devices are used to provide 23 services: a light and a fan provide on/off services, a TV provides on/off, 12 channels and 2 videos, and a music player provides on/off and music mute/loud services.

All the services are recorded as a time series $\{S_1, S_2 \ldots\}$ in database. Each service is recorded as S (SN, ST), which respectively means the service ID, and the time at which this service is recorded. We predefine a set of services $\{s_1, s_2 \ldots s_M\}$ (such as s_1 means "turn on light", s_2 means "mute music"), and $SN \in \{s_1, s_2 \ldots s_M\}$.

Synapse Core

We apply HMM to model the relationship between the sensor events and the services. Figure 2 shows one cycle of Synapse model. There are two basic components in HMM: the hidden state X_t and the observation of state Y_t. In Synapse, each hidden state X_t corresponds to a service S_t (*not lowercase s*), to indicate the situation in which this service is used, and the observation Y_t is a vector of event values $(y_1, y_2 \ldots y_N)$, which are the current values of $\{e_1, e_2 \ldots e_N\}$. There are three parameters in HMM: the prior probability $\pi(i)=P(X_1=i)$ which represents the initial state, the transition matrix $A(i,j)=P(X_t=j|X_{t-1}=i)$ which represents the probability of transfer from $X_{t-1}=i$ to $X_t=j$, and the observation model $P(Y_t|X_t)$ which represents the relation of X_t and Y_t [16]. The learned results in one cycle are used as the initial estimations of the next cycle.

In the Learning Phase ($1 \le t \le T$), Synapse uses the history records of sensor events and services that happened during t=1, 2…T to compute $A(i,j)=P(X_t=j|X_{t-1}=i)$ and $P(Y_t|X_t)$. We assume that sensor events, which happened in a certain interval before a service, indicate the situation in which this service is used. For instance, in Figure 2, E_2 and E_3 indicate the situation in which S_2 is used. Since X_t cannot be observed directly, we firstly use the forwards-backwards algorithm [17] to infer $X_{1:T}$ from the observation $Y_{1:T}$. In the forwards pass, we recursively compute the filtered estimate $\alpha_t=P(X_t=i|Y_{1:t})$, and in the backwards pass, we recursively compute $\beta_t=P(Y_{t+1:T}|X_t=i)$; then combine them to produce the smoothed estimate $\gamma_t(i)=P(X_t=i|Y_{1:T})$ and the smoothed two-slice estimate $\xi_{t-1,t}|T(i,j)=P(X_{t-1}=i,X_t=j|Y_{1:T})$. After that, we use EM (expectation maximization) algorithm [17] to learn $A(i,j)=P(X_t=j|X_{t-1}=i)$ and $P(Y_t|X_t)$ from $\gamma_t(i)$ and $\xi_{t-1,t}|T(i,j)$.

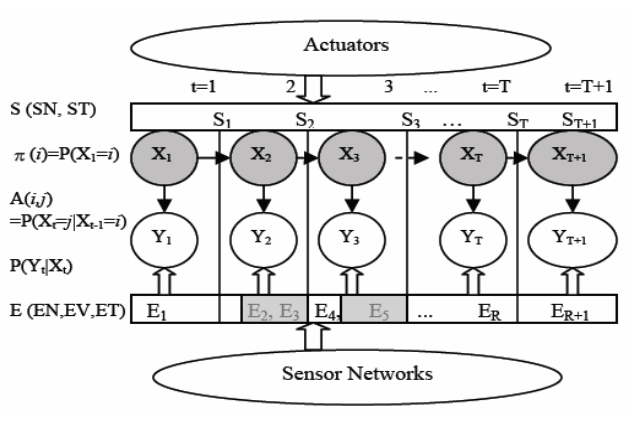

Fig. 2: One Cycle of Synapse Model. The grey rectangles indicate a certain interval before S_t.

In the Executing Phase (t>T), Synapse uses the learned transition matrix $A(i,j)=P(X_t=j|X_{t-1}=i)$, observation model $P(Y_t|X_t)$ and the current observation Y_t to compute the occurrence probability of each service. A two-step filtering algorithm is applied: in update step, the probabilities of current state can be gained as we compute $P(X_t|Y_t)$; in predict step, the probabilities of next state can be predicted as we compute $P(X_{t+1}|Y_t)$. As a result, the occurrence probability of each service can be computed as the occurrence probability of each state corresponding to these services. After that, we can sort the services in a descending order of probability. If a probability is higher than a user-defined threshold, the corresponding service will automatically start in Active Mode. The top 5 services will be recommended as a list to the user interface in Passive Mode. Passive Mode is mainly used in Synapse. All these algorithms are implemented on Matlab.

User Interface

Synapse provides a user interface in XML form on Matlab Web Server. Users can browse this web through PC, PDA, or cellular phone, and start a service by selecting the service ID. The recommended service list on this web can automatically update after a fixed interval, or be manually updated by users.

PRELIMINARY EVALUATION OF SYNAPSE

In order to examine the practicability of our methods, we implemented three simple scenarios on the smart home test bed, and preliminarily evaluated Synapse on three aspects: 1) feasibility of Synapse, which means whether Synapse can successfully provide services based on the learned habits and the current sensor events, 2) time complexity of algorithms, which examines whether it is practically quick enough to gain the results, 3) correctness of the recommendation, which examines whether the results of prediction are practically accurate enough.

Feasibility of Synapse

Using the sensors and devices mentioned in section 3, we collected 200 training samples: 40 of which are "Light" scenario using "Light_On" service, 80 of which are "TV" scenario using "Video" and "TV_1ch" services (40 respectively), and 80 of which are "Music" scenario using "M_Mute" and "M_Loud" services (40 respectively). Each sample is a combination of one service and a group of sensor events. For instance, in "Light" scenario, when a user was in the room and it was too dark, he selected "Light_On" service, so the user's ID, the brightness and the "Light_On" service were recorded as one training sample. We used such training samples to learn users' habits in three scenarios.

After learning, we changed the status of users and environment, and Synapse successfully provided dynamic services adapting to the changed situation. For instance, in "Music" scenario: when we was using the phone, Synapse provided "M_Mute" to turn down the volume of the music player; when we finished using the phone, Synapse provided "M_Loud" to turn up the volume. These were collected as test samples, which were used to examine the correctness of recommendation.

Time Complexity of Algorithms

We simulated the time complexity of algorithms on Matlab. The number of hidden state – M and the number of training sample – T are important to estimate the complexity.

In the Learning Phase, if there are M hidden states, it will take $O(M^2)$ operations at every time slice, since we must do several matrix-vector multiplies per time slice. And as we must repeat this procedure during t=1, 2…T, it will totally take $O(M^2T)$ time. In the Executing Phase, algorithms do approximately the same work as learning algorithms do at one time slice. Therefore, the time complexity is $O(M^2)$. Figure 3 depicts the time complexity of learning algorithms.

Figure 3 shows that for 50 states and 2600 training samples, it takes approximately 80 seconds to learn the parameters, which reveals that our methods are practically quick enough for real life.

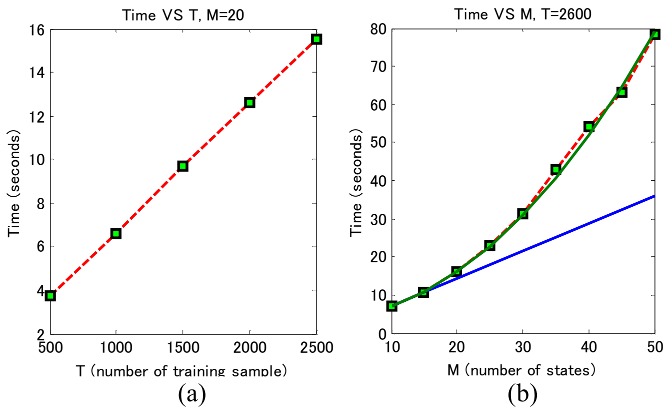

Fig. 3: (a) shows the relation of Time and T. (b) shows the relation of Time and M. The green curve is quadratic, and the blue one is linear.

Correctness of Recommendation

We verified the correctness of recommendation with 150 test samples: 30 of which are "Light" scenario test samples, 60 of which are "TV" scenario test samples (30 for Video, 30 for TV_1ch), and 60 of which are "Music" scenario test samples (30 for M_Mute, 30 for M_Loud). We only tested correctness of the first recommendation because of the definitude of result. (The correctness of top 5 recommendations will be tested by real inhabitants in the future.) The correctness of recommendation is shown on Table 1, which reveals that our methods are practically accurate enough for real life.

Table1: Correctness of Recommendation

Services	Light_On	Video	TV_1ch	M_Mute	M_Loud
Correct	96.7%	93.3%	90.0%	93.3%	90.0%

CONCLUSION AND FUTURE WORK

In this paper, we presented a context-aware service platform – Synapse and its smart home test bed. By exploiting the recorded histories of contexts and services, Synapse can learn the users' habits. After that, Synapse can predict the most relevant services that users will use in current situation based on their habits, and provide services in Active Mode and Passive Mode. We described our algorithms and the implementation of smart home test bed in detail. The preliminary evaluation with real world data revealed that Synapse was practicable and should be built at home.

Now we are extending the sensor and service parts and implementing an entire Synapse system in a house. The experiment with real inhabitants will be conducted in the future.

ACKNOWLEDGMENTS

This work is supported by Ministry of Public Management, Home Affairs, Posts and Telecommunications.

REFERENCES

1. Aware Home. http://www.cc.gatech.edu/fce/ahri/.
2. Barkhuus, L. Is Context-Aware Computing Taking Control Away from the User? *Proceedings of Ubicomp 2003*, LNCS 2864.
3. Das, S. The Role of Prediction Algorithms in the MavHome Smart Home Architecture. *IEEE Wireless Communications*, vol. 9, no. 6, pp. 77-84, Dec. 2002.
4. Graumann, D., Lara, W. Real-world implementation of the location stack: The universal location framework. *Proceedings of WMCSA 2003*, 122–128.
5. Hamilton, J. *Time Series Analysis*. Wiley, 1994.
6. Horvitz, E. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. 1998.

7. House_n. http://architecture.mit.edu/house_n/.
8. Jelinek, F. *Statistical methods for speech recognition*. MIT Press, 1997.
9. KAWAHARA, Y. Design and Implementation of a Sensor Network Node for Ubiquitous Computing Environment. *Proceedings of IEEE Semiannual Vehicular Technology Conference*, 2003.
10. Korpipä, P. Bayesian approach to sensor-based context awareness. *Personal and Ubiquitous Computing*, Vol. 7 Issue 2, July 2003.
11. Korvemaker, B. Predicting UNIX Command Lines: Adjusting to User Patterns. *National Conference on Artificial Intelligence 2000*, AAAI press, 230-235.
12. López, D., Katsiri, E. An ECA Rule-Matching Service for Simpler Development of Reactive Applications. *IEEE Distributed Systems* 2001, Vol. 2.
13. Mantoro, T. Location history in a low-cost context awareness environment *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003,* Vol. 21.
14. Meek, C. Autoregressive tree models for time-series analysis. *Proceedings of the Second International SIAM Conference on Data Mining 2002*, 229–244.
15. Meyer, Sven. A survey of research on context-aware homes. *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003*, Vol. 21.
16. Mozer, M. An intelligent environment must be adaptive. *IEEE Intelligent Systems*, vol. 14, no. 2, pp. 11-13, Mar. /Apr. 1999.
17. Murphy, K. Dynamic Bayesian Networks: Representation, Inference and Learning. PhD Dissertation, UC Berkeley, 2002.
18. Rabiner, L. R. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE* 1989, 77(2):257–286.
19. Rao, S., Cook, D. J. Predicting Inhabitant Actions Using Action and Task Models with Application to Smart Homes, *International Journal of Artificial Intelligence Tools*, 13(1), 81-100, 2004.
20. Ron, D., Singer, Y. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning* 1996, 25.
21. Salber, D., Dey, A.K. The context toolkit: Aiding the development of context-enabled applications. Technical report (2000), Georgia Institute of Technology.
22. Zadeh, L.,”Fuzzy sets”, Information and Control 8:338-353, 1965.

BIOGRAPHIES

Hua Si received the B.E. in Electronic Techniques and Information Systems from Tsinghua University, Beijing, China, in 2002. He is currently a master student of the Graduate School of Information Science and Technology at the University of Tokyo with an emphasis on ubiquitous middleware applications and machine learning. He is a student member of IEEE and IEICE.

Yoshihiro Kawahara received the B.E., M.E., and Dr. Eng. degrees in Information Science and Technology from the University of Tokyo, Tokyo, Japan, in 2000, 2002, and 2005, respectively. He is currently a Research Associate of the Graduate School of Information Science and Technology at the University of Tokyo. His research interests are in the areas of context-aware computing, computer networks, and wearable computing. He is a member of IEEE, IEICE, and IPSJ.

Hiroyuki Morikawa received the B.E., M.E., and Dr. Eng. degrees in electrical engineering from the University of Tokyo, Tokyo, Japan, in 1987, 1989, and 1992, respectively. He is currently an Associate Professor of the Department of Frontier Informatics at the University of Tokyo. From 1997 to 1998, he stayed in Columbia University as a visiting research associate. His research interests are in the areas of computer networks, ubiquitous networks, mobile computing, and wireless networks. He serves as Editor of Transactions of Institute of Electronics, Information and Communication Engineers (IEICE) and on the technical program committees of IEEE/ACM conferences and workshops. He is a member of IEEE, ACM, ISOC, IPSJ, and ITE.

Tomonori Aoyama received the B.E., M.E. and Dr. Eng. from the University of Tokyo, Tokyo, Japan, in 1967, 1969 and 1991, respectively. Since he joined NTT Public Corporation in 1969, he has been engaged in research and development on communication networks and systems in the Electrical Communication Laboratories. From 1973 to 1974, he stayed in MIT as a visiting scientist to study digital signal processing technology. In 1994, he was appointed to Director of NTT Opto-Electronics Laboratories, and in 1995 he became Director of NTT Optical Network Systems Laboratories. In 1997, he left NTT, and joined the University of Tokyo. He is currently Professor in the Department of Information and Communication Engineering, Graduate School of Information Science and Technology, the University of Tokyo. His research activities cover the next generation networking technologies from layer 1 (e.g. photonic networks) to higher layers including middleware for network collaboration, P2P routing, mobile networking, and ubiquitous networking. Dr. Aoyama is involved in several governmental projects such as Japan Gigabit Network (JGN) and the Ubiquitous Networking Forum, and in some non-profit organizations and consortiums such as the

Photonic Internet Forum (PIF) and the Digital Cinema Consortium (DCC) in which he is serving as Chairman. Dr. Aoyama is IEEE Fellow and was a Members-at-Large of the IEEE ComSoc Board of Governors. He served as President of IEICE Communication Society. He also served Chair of IEEE ComSoc Japan Chapter. He is a member of IPSJ and IEEE.