Proximity Mining: Finding Proximity using Sensor Data History

Toshihiro Takada Satoshi Kurihara Toshio Hirotsu Toshiharu Sugawara
NTT Network Innovation Laboratories
NTT Corporation
{takada,kurihara,hirotsu,sugawara}@t.onlab.ntt.co.jp

Abstract

Emerging ubiquitous and pervasive computing applications often need to know where things are physically located. To meet this need, many location-sensing systems have been developed, but none of the systems for the indoor environment have been widely adopted. In this paper we propose Proximity Mining, a new approach to build location information by mining sensor data. The Proximity Mining does not use geometric views for location modeling, but automatically discovers symbolic views by mining time series data from sensors which are placed in surroundings. We deal with trend curves representing time series sensor data, and use their topological characteristics to classify locations where the sensors are placed.

***Keywords** — Proxymity Mining; Location modeling; Zero configuration; Location-aware computing; Context-aware computing; Pervasive computing; Ubiquitous computing; Spatial Data Mining; Real-space computing*

1. Introduction

As the recent interest in ubiquitous computing [20] shows, in the future we will find more and more computational devices surrounding us in our daily lives. This is made possible not only by improvements in power usage and hardware miniaturization, but also by the large address space of IPv6, local communication protocols such as IEEE 802.11 and Bluetooth, and by technologies such as RFID that allow wireless identification of objects and people. We have chosen an auto-configuration feature and human-environment interface as being two important technologies in this situation.

A method for address auto-configuration is defined in IPv6 [18], and this makes it possible for a huge number of devices to be IP-reachable. However, this auto-configuration only assigns IP addresses to these devices, and is of no use in finding the processor's physical location or the presence of other devices in the area. IETF zeroconf [6], a protocol set for configuration-free networking, merely makes name resolution and the discovery of services such as printers possible.

If all of the input devices (sensors and controllers, etc) and output devices (lighting and sound sources, actuators, etc) in the home or office were to be connected to a network, the number of devices involved would easily be on the order of 10^2 for a household and 10^3 for an office floor. Setting up this many devices manually is not realistic, and in addition, the use of wireless technology means that some of the devices will be moveable and must thus be dynamically added to and deleted from the network. It is necessary to create a system platform that allows for the auto-configuration of physical devices in a real-space environment, based on their location and input/output abilities.

But why is it necessary to have so many processors with their sensors and actuators around us? One reason is that the objects to be controlled and the data to be measured exist all throughout the real-space environment. For example, in a house there are many electronic devices and lighting, heating and water appliances to be controlled. In order to control these devices and appliances, it is necessary to collect data from the environment (data such as temperature, humidity, noise, and scents, etc). The same is true in an office or factory setting. For example, a rack of computers and networking devices would include data in the traditional sense (the contents of files and packets, etc) as well as data about the state of the devices themselves (CPU and disk usage data, network interface error rates, etc). In addition, other data would also exist such as the manufacturer's serial number, owner information, fan and disk noise, temperature, and smell (does it smell burnt?).

Indeed, in the real-world environment, various kinds of data and properties surround us. However, merely displaying all of them to a user is a recipe for failure. In order to make an environment of ubiquitous sensors, processors, and actuators a reality, it is necessary to have an interface that retrieves only the environmental data and properties needed at a particular time. We see this as an interface between users and environmental properties, and believe that it is necessary to construct a system platform that would make this possible. For the above reasons, we believe that the auto-configuration feature and the human-environment interface are two important technological factors in the creation of an environment filled with a large number of sensors, processors and actuators.

Ubiquitous and pervasive computing applications often need to know where people and things are physically located. However, most location systems require painstaking pre-configuration in order to obtain accurate location data [4]. For example, the location of beacons must be accurately determined beforehand. This kind of pre-configuration not only goes against the kind of system deployment, it also affects the dynamic adaptability of the system to the environment. Thus, our goal is to avoid pre-configuration for location systems wherever possible.

In this paper we focus on a technique to automatically build location information. The core of our contribution is Proximity Mining, a new approach to build location information by mining sensor data. The Proximity Mining does not use geometric views for location modeling, but automatically discovers symbolic views by mining time series data from sensors which are placed in surroundings. We deal with trend curves representing time series sensor data, and use their topological characteristics to classify locations where the sensors are placed.

The rest of this paper is organized as follows. In Section 2, we outline the general concept of location modeling and semantic proximity. In Section 3, we explain the relation between semantic proximity and sensor data. We then introduce the method, Proximity Mining in Section 4, and describe the result of initial experiments in Section 5. In Section 6, we discuss our initial experience regarding the prospects of our method, and conclude the paper in Section 7.

2. Location systems and proximity

2.1. Location modeling

When creating location-aware applications, how to model and express physical space is a major problem. There are many location models in the field of ubiquitous computing research [2]. These location models can be divided into two groups — "geometric model" and "symbolic model". [12] Geometric model is based on a system of coordinates (e.g. based on GPS). Symbolic model essentially involves labeling each location (and then describe the relation between labels with a tree or a graph of some kind) [3][14]. Each model has pros and cons, and several hybrid models have also been proposed that combine both kinds of model [9][10][12]. Research in autonomous robots [19] and MANET (Mobile Ad-hoc Networks) [1] has also involved similar work into location and map models.

Most outdoor location-aware systems make use of GPS — a geometric location model. On the other hand, while there are many proposals for indoor locations sensing schemes [7], because of problems with accuracy, scalability, cost, and the complex setup required, none of them have been widely adopted. Moreover, a simple geometric location modeling is not enough to create a location-aware system, as we will explain in the following section.

2.2. Semantic proximity

For building location-aware applications, it is necessary to use a more advanced definition of distance — one that is not strictly geometric. An example is the following:

> *The other side of a wall or partition may be close in terms of absolute distance, but because a person must take the long way around to reach that point, it can be considered "far".*

Here is another example, that is the real case at the authors' own office:

> *There is a workspace separated by high partitions. It is possible to talk over the partitions, or to hand something to someone on the other side. However, since it isn't possible to see the computer display situated above a coworker's desk, it is necessary to go to the coworker's cubicle to have a discussion about something on the screen.*

We refer to this advanced concept of distance, which changes depending on locale and context, as "proximity". [5][16] One of the goals of our research is to discover to what extent context-aware applications can be created using only symbolic location models based on this concept of proximity.

3. Finding proximity using sensor data history

3.1. Problem definition

In this section we will show an example of a problem that needs to be solved. Imagine an office with 19 sensor points (points A through S, See Figure 1) and 8 doors (door 1 through 8). Each sensor point is equipped with the following kinds of sensors:

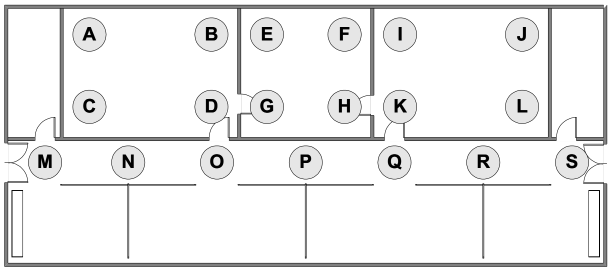


Figure 1. Sensor points and doors at an office floor

- Light level sensor
- Temperature sensor
- Humidity sensor
- Human detecting sensor (a pyroelectric motion sensor or an optical Position Sensitive Detector to detect a human body)

Moreover, each door is equipped with a reed switch. This switch acts as sensor to detect opening and closing the door. (Hereinafter sensors and reed switches are collectively called sensors.) We can collect sensed data from these sensors, but we cannot know the accurate physical locations of the sensors.

The problem we must solve here is how to find things like "rooms" "corridors", and "traffic paths" by using the data history from these sensors. For example, if the light level sensors for points A, B, C, and D always change at the same time, it can be surmised that these points are all in one room with a single light switch. If the temperature and humidity sensors for points E through H always change together (separately from the other points) it can be surmised that they exist in a room with independent air conditioning controls (a machine room, for example).

Further, if by examining the data history from the human sensors it is determined that there are large numbers of people traveling from point M to point N, and from point N to point O, it would be possible to conclude that points M, N, and O form a single "corridor" of some kind. The same thing might be said of points S and R, and R and Q. Moreover, if by combining the data history from human sensors and door sensors we often observe the sequence of events that point O detects a person, door 3 opens, point D detects a person, door 4 opens, and then point G detects a person successively, it can be surmised that points O, 3, D, 4, and G form a "traffic path".

Thus, in general, discovering proximity involves determining the relationships between sensors by examining dynamic and static sensor data correlations. The consequent proximity is used to cluster sensors. The result of clustering will be then used for forming a location model (See Figure 2).

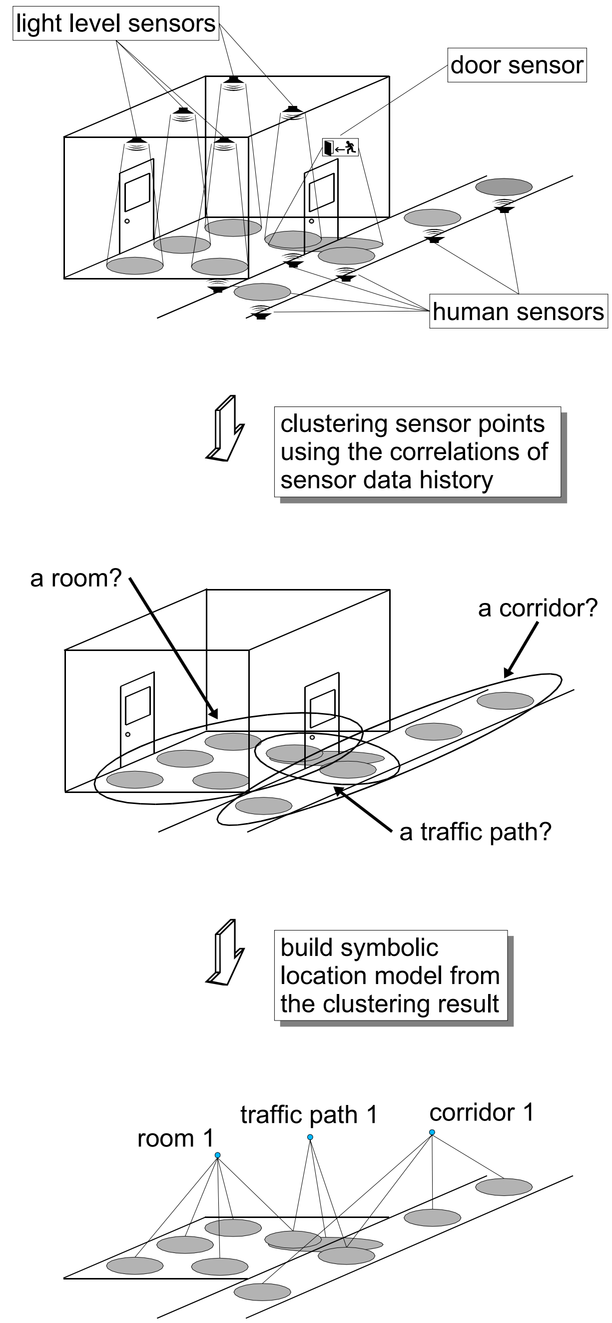


Figure 2. From sensor data to proximity clusters, from clusters to location model

3.2. Characteristics of the problem

The process for finding proximity of sensors can be seen as a kind of data mining. Thus, some compensation techniques for error data used in normal data mining can be applicable. We should also note that the goal of the mining is to cluster not the data values but rather the aspects of how data changes. For example, imagine the following kind of situation:

- The illumination in a room with a single light switch always changes in the same manner, but the absolute intensity of illumination (light level) differs from location to location in the room

We have to therefore calculate the correlation between data by not the value of sensed data, but rather the time when the value of sensed data distinguishably change.

In addition, there are two types of the changes. Namely,

- Changes of sensor data values that happen almost synchronously (Figure 3), applicable to the case of detecting a "room" using the data history from light, temperature and humidity sensors
- Changes of sensor data values that happen successively (Figure 4), applicable to the case of detecting a "traffic path" using the data history from human and door sensors

Thus, the problem we have to solve can be regarded as a sort of qualitative similarity detection on a time series curve.

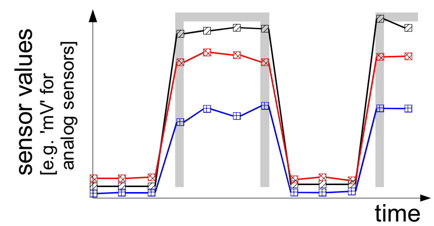


Figure 3. Values change synchronously

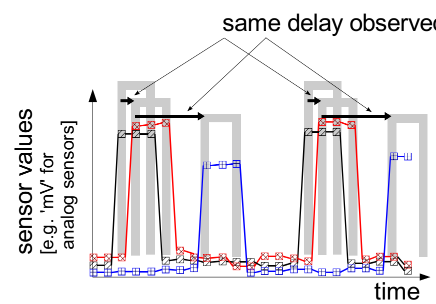


Figure 4. Values change successively

4. Proximity Mining

4.1. Outline of the method

In this section we outline the Proximity Mining being proposed in this paper. It makes use of an analysis of the topological characteristics of time series curves [15]. We first extract *outstanding peaks and bottoms* (described in Section 4.2) from the time series of sensed data values, and calculate degrees of correlation for their order of appearance in the time series as degrees of correlation for the curve. The clustering of time series curves are then performed using the degrees of correlation. This method is based on a characteristic that the locations of the peaks and bottoms on the time series curve are firm properties in relation to the continual change of the time axes and observed data values.

The outline of the whole method is as follows:

Phase 1. Preprocess data
- Discard obvious error data (e.g. outside of sensor range)

Phase 2. Calculate the sequences of outstanding peaks and bottoms for each sensor data history

Phase 3. Cluster sensors
- Do simple clustering. The edit-distance between the sequences is used as a distance measure of the similarity between objects which to be clustered (sensors, in this case)

In the next section, we will elaborate on an algorithm used in Phase 2. Then, we will show a brief explanation of Phase 3 in Section 4.3.

4.2. Algorithm for calculating the peak/bottom sequences

At the core of our method is an algorithm for calculating outstanding peak/bottom sequences for each sensor data history. The data history of each sensor is represented by a measured value on the time series. A summary of the algorithm is as follows (also see Figure 5).

Step 0. Define threshold h*

Step 1. Extract only data that represent the peaks and bottoms of the time series

Step 2. Remove any peak and bottom data whose relative height is less than threshold h*

Step 3. Extract only peaks and bottoms from the data remaining after Step 2

Step 4. Repeat Step 2 if any peaks or bottoms whose relative height is less than h* exist. If not, processing finishes

We define that any data remaining after this processing has finished are the outstanding peaks and bottoms of that data set. The sequence of outstanding peaks and bottoms, $C(f)^{[t,t+Tk]}$, is defined by the order of appearances of peak P and bottom B for a particular time period $[t,t+Tk]$ of

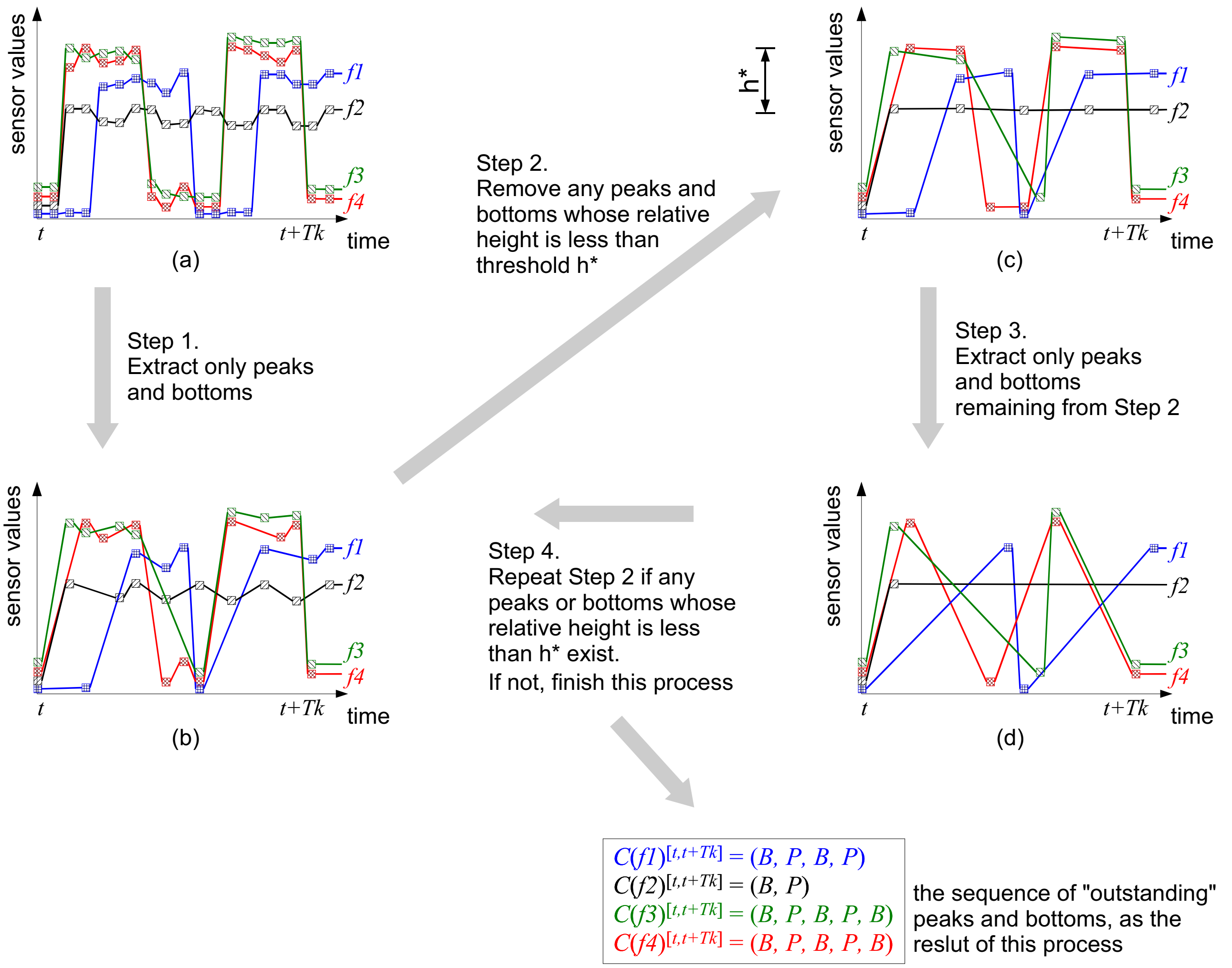


Figure 5. Proximity Mining: the overview of the algorithm for calculating the peak/bottom sequences

time series data f, which are arrived at through the algorithm.

For example, in Figure 5, there are four pieces of sensor data history, *f1, f2, f3,* and *f4*. Please note that, in the case of this example, we assume that *f3* and *f4* change synchronously, *f3* and *f1* change successively (of course *f4* and *f1* change successively too), and *f2* changes independently. As shown in the Figure 5, the resulting peak/bottom sequences become:

$$C(f1)^{[t,t+Tk]} = (B,P,B,P)$$
$$C(f2)^{[t,t+Tk]} = (B,P)$$
$$C(f3)^{[t,t+Tk]} = (B,P,B,P,B)$$
$$C(f4)^{[t,t+Tk]} = (B,P,B,P,B)$$

4.3. Clustering sensors

In this section, we will describe briefly how sensors are clustered. The general idea of clustering is to group similar objects together. In this paper, an object to be clustered is a sensor (not a sensor point). This means that we can find even the correlation between sensors at the same sensor point.

A distance measure of the similarity between two objects is essential to most clustering procedures. In our method, the edit-distance between the sequences is used as the distance measure. For instance, illustrating with examples in Section 4.2 (and Figure 5), the distances between *f1, f2, f3* and *f4* are:

$$D(f3,f4)^{[t,t+Tk]} = 0$$
$$D(f1,f3)^{[t,t+Tk]} = D(f1,f4)^{[t,t+Tk]} = 1$$
$$D(f1,f2)^{[t,t+Tk]} = 2$$
$$D(f2,f3)^{[t,t+Tk]} = D(f2,f4)^{[t,t+Tk]} = 3$$

In the context of this paper, the shape, size, and number of the cluster is generally unknown. So, we solely use the simple clustering algorithm [10] at this time. The sensor clustering is actually done while altering the time period $[t,t+Tk]$. At the present time, we use five patterns for the length of the time period (Tk), 1-hour, 6-hours, 1-day, 4-days, and 1-week. Since the sensor data values are profoundly affected by the activities of the people using the environment, it was thought best to choose lengths of time that might correspond to the lengths of normal human activities.

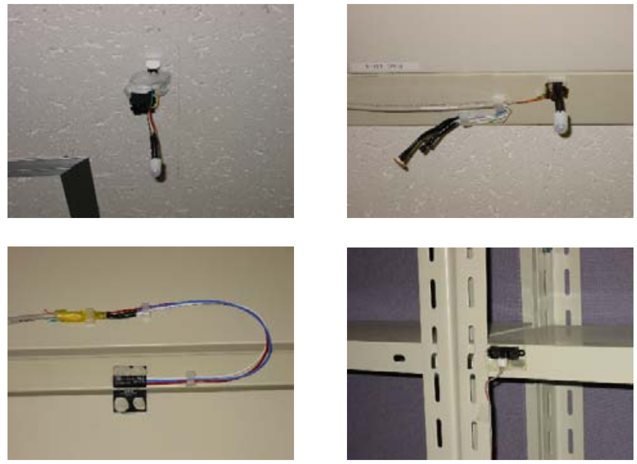


Figure 6. Sensors in our office

5. Initial experiments of Proximity Mining

5.1. Sensor hardware and setting

We have implemented an initial experimental setup for the Proximity Mining. This section explains sensor hardware and setting used in this experimentation.

At the time of this writing, we are collecting data from a total of 52 sensors in 12 points on a single office floor. We place sensors in three rooms, three doorways, and one corridor. There are 9 kinds of sensors used:

1) *Light level sensors* — Using CdS or photodiode. Yield a nominal light level (the intensity of illumination). Original sensor circuit is designed to provide a signal between 0 and +12 volts DC.

2) *Temperature sensors* — Using LM35DZ by National Semiconductor Corp. Provide a signal between 0 and +5 volts DC, linear +10.0 mV/°C scale factor, and ±0.5 °C accuracy.

3) *Humidity sensors* — Using CHS-GSS by TDK. Original sensor circuit is designed to provide a signal between 0 and +12 volts DC, linear +10.0 mV/% scale factor, and ±5 % accuracy.

4) *Odor sensors* — Using NAP-11AS, In_2O_3 semiconductor type gas sensor, by Nemoto & Co., Ltd. Yield a nominal odor level of various smells generated in a normal living environment, such as cooking odors, putrid smells, organic solvent smells, cigarette smoke, cosmetics, coffee, etc. Original sensor circuit is designed to provide a signal between 0 and +12 volts DC.

5) *Voltage sensors* — Voltmeter for some electric devices in the office. Original sensor circuit is designed to provide a signal between 0 and +12 volts DC.

6) *Current sensors* — Ammeter for some electric devices in the office. Original sensor circuit is designed to provide a signal between 0 and + 12 volts DC.

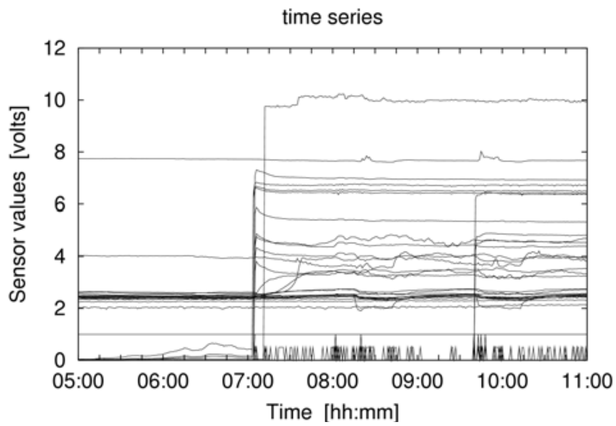
7) *Pyroelectric motion sensors for human detecting* — Using NaPiOn by Matsushita Electric Works, Ltd. Detect human bodies (detect changes in infrared light that occur due to the movement of a living body). Provide a signal between 0 and +5 volts DC. Detecting distance is about from 0 to 5 meter.

8) *Optical Position Sensitive Detectors for human detecting* — Using PSD by Sharp Corp. Measure a distance to an object (suspecting the human body). Original sensor circuit is designed to provide a signal between 0 and +5 volts DC. Detecting distance is about from 0.2 to 1.5 meter.

9) *Reed switches* — Detect open and close the door. Original sensor circuit is designed to provide a signal 0 volts DC when the door closes and +5 volts DC when open.

Analog signals from sensors are passed to micro-controller (PIC) boxes. These boxes do analog-to-digital conversion, encapsulate them into UDP packets, and then send the packets to a data-gathering host via Ethernet. To send data, either wired (10/100BASE-T) or wireless (IEEE 802.11a/b) network connection is used, according to where sensors are located. The analog-to-digital conversions have 10 to 16 bits accuracy (depend on which micro-controller chip is used). In the most cases, a single micro-controller box serves 4 to 8 sensors.

For the sensors 1) to 6), they are sampled and data-sent at a rate of once per minute. On the other hand, for the sensors 7) to 9), sensors are sampled at a rate of once



(a) 6 hours

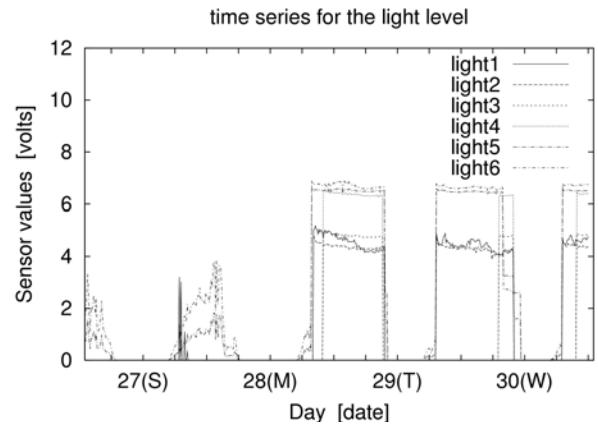
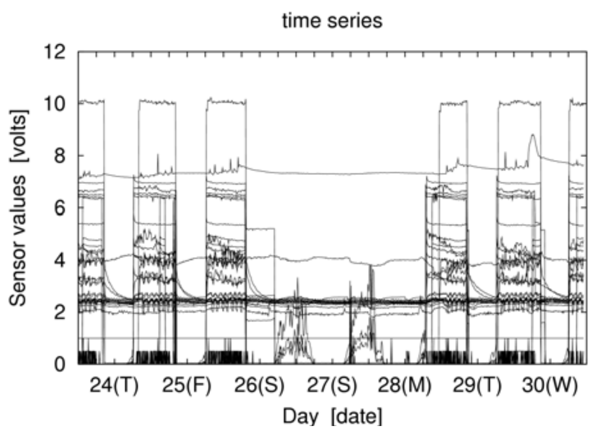


Figure. 8 Time series of the light sensors



(b) 1 week

Figure 7. Time series of all sensor data

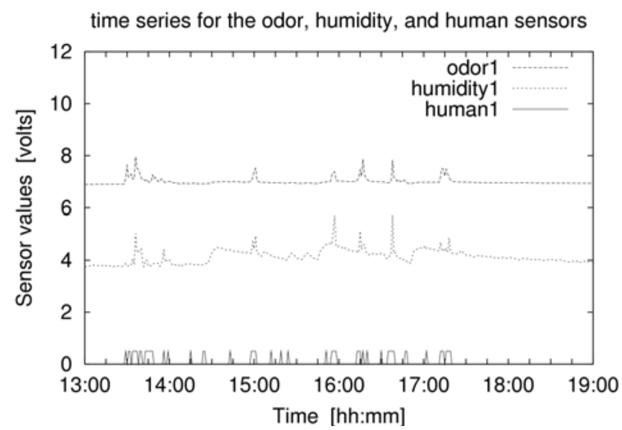


Figure. 9 Time series of the odor, humidity, and human sensor

per 10 to 100 milliseconds, but the micro-controllers just keep watching over whether the human body is detected (the door opens). And only once in a minute, the results of the occurrence of the human detection (door opening) during the minute are sent to the host. Figure 6 shows pictures of sensors placed in our office.

5.2. Experimental results

First of all, we will show examples of time series data from sensors. One of easiest and fastest way to grasp general characteristics of these data and also to predict how a system could group the raw data into clusters, is by plotting the output of all sensors directly on a time scale in parallel. Figure 7(a) shows time series of all sensor data during six hours, and Figure 7(b) shows ones for a week. Since there are too many data to plot, we cannot show the functions of each time series in these figures.

However, some periodic changes can be found in these plots, and it could be imagined that those changes are affected by the activities of the people in our office. For example, Figure 7(a) shows that, one day, in the morning, a person came to the office and turned the light of a room on at just after 7:00 am. He or she switched some office equipment on at about 7:10 am. At around 9:45 am, the light of another room was switched on. In addition, Figure 7(b) shows that we did not work at midnight, as well as on the weekend.

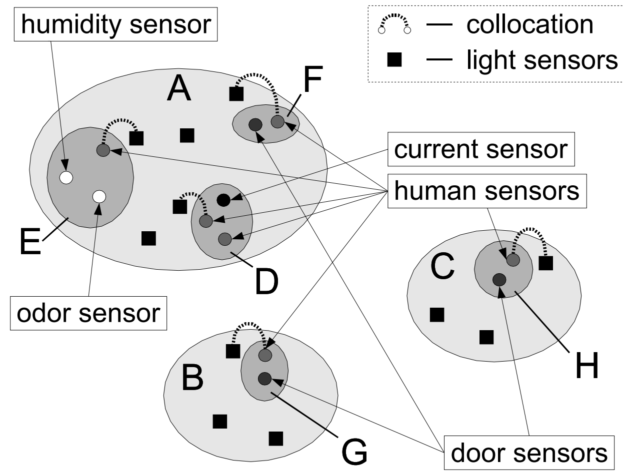
To pay attention to some time series data, Figure 8 shows that time series of six sensors values indicating the light level in three different rooms during four days. Actually, light1 and light2 are in the room-1, light3 and light4 are in the room-2, and light5 and light6 are in the room-3. Similarly, Figure 9 shows that time series of three sensors values during six hours. They are one odor sensor, one humidity sensor, and one human detecting sensor.

With this experimental environment and the sensor setup, our algorithm typically results in generating the following eight clusters:

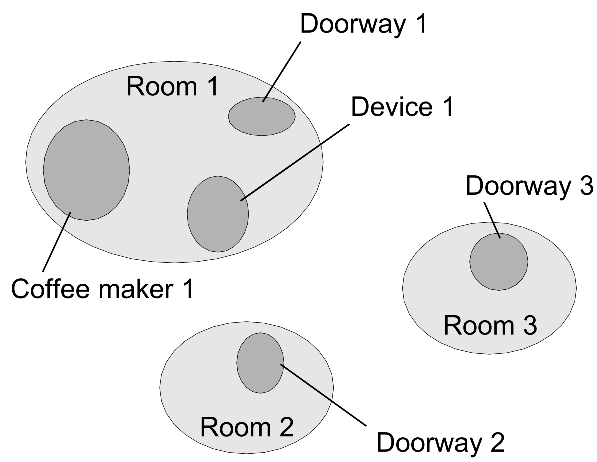
1) *Cluster-A*
   - Contains five light level sensors
2) *Cluster-B*
   - Contains three light level sensors
3) *Cluster-C*
   - Contains three light level sensors
4) *Cluster-D*
   - Contains one current sensor and two human sensors
5) *Cluster-E*
   - Contains one odor sensor, one humidity sensor, and one human sensor
6) *Cluster-F*
   - Contains one door sensor and one human sensor
7) *Cluster-G*
   - Contains one door sensor and one human sensor
8) *Cluster-H*
   - Contains one door sensor and one human sensor

Based on the above clustering result and a knowledge that some light sensors and human sensors are placed at the same sensor point, we can now assume that Cluster-A includes Cluster-D, Cluster-E, and Cluster-F, Cluster-B is includes Cluster-G, and Cluster-C includes Cluster-H, respectively. Figure 10(a) illustrates the final result of the clustering.

Now it is time to examine the result. For Cluster-A, Cluster-B, and Cluster-C, these clusters clearly represent the discovery of three rooms. Cluster-D represents the existence of a certain device in our office and the people using it. Cluster-E shows the area where people appear, and an odor and moisture breaks out sometimes. Actually, Cluster-E reflects a coffee maker table in the room. Cluster-F, Cluster-G, and Cluster-H represent doorways. We illustrate this interpretation of the clustering result in Figure 10(b).



(a) Resulting clusters and sensors



(b) Interpretation of the resulting clusters

Figure 10. The result of the sensor clustering

6. Discussion

6.1. Experience with sensor data mining

Current Proximity Mining algorithm only generates *subsymbolic* information of locations. This is principally because, at the moment, we do not use any ID-based information on things and persons (like using RFID). Thus, as shown in the previous section, it is difficult to determine things like which cluster corresponds to which room, or how the doorways are connected to those rooms. This makes it hard to be grounded [7] clusters in actual properties. However, we think such subsymbolic information is a clue to elucidate the logical structure of surroundings with less administrative effort.

Another possible problem with our method is that it takes a certain amount of time to build up the data history needed to create the location information. However, we think that this boot-up procedure could be shortened by users' assistance. For example, at the initial time, a user can turn the light a room on and off, quickly over again intentionally, to make the system recognize this room. A user also can touch human sensors one by one along the hallway, to show the "traffic path".

Now let us discuss the sensors separately. First of all, the light level sensors are particularly informative. Almost all rooms (with a single light switch) can be found by mining only the light sensor data. On the other hand, temperature and humidity sensors are not so usable. One of the reasons for this incompetency is that the experimentation is done in “well air-conditioned” office. At this moment, we could not estimate if these sensors are usable in less air-conditioned environments or not. However, as shown in the result, the humidity sensor played an unforeseen role, finding steam. It is easy to imagine that humidity sensors (and odor sensors) will be used for finding a kitchen in the home environments, and furthermore, used for mining the context of the kitchen. The human detectors (both pyroelectric motion sensors and optical Position Sensitive Detectors) and door sensors by reed switches work very well.

The experimental environment has been set at part of our office. Eight persons usually work, five to ten neighbor colleagues often drop in, and casual guests visit sometimes at this area. It is not yet considered that the effect of the number of people in the environment, especially upon finding like corridors and traffic paths.

6.2. Related work

Some kinds of data mining make use of data from spatial phenomena, and there is research being done into this “spatial data mining”. [13] Spatial data mining in general is concerned with using the spatial characteristics of data to extract patterns and clusters. We wish to point out that in our own research we are concerned with extracting spatial structure itself from time series data, and in that respect it is different from the work being done by others.

The most closely related to our work is that research in sensor-based context-awareness. For example, in [5][17], sensor data are mined to obtain the context of the people (or things like mobile phone and coffee cup). However, in contrast to their approach, we are now focusing to obtain location information, in other words, to reveal the semantic structure of the surroundings where people are. We think these two approaches will complement each other.

7. Conclusion and Future Work

In this paper we have introduced Proximity Mining, the new approach to build location information by mining sensor data. It analyzes sensor data history to examine the dynamic and static sensor data correlations, and then clusters sensors by using the correlations to find the structure of surroundings. Also we have reported the results of our initial experiments of this approach.

In a sensor-filled real-world ubiquitous computing environment, it is very important to be able to determine the placement of sensors with less administrative effort. We plan to build an actual location system by solely using the analysis of device proximity data, and implement context-aware applications based on this location system.

Our main contention in this research is that location-aware applications only actually need the answers to queries about things like “distance” and “inclusion”, and that geometric coordinates are merely ways of calculating these values. Thus, if the answers to these queries can be arrived at in some other way, there is no need to insist on using a geometric location model. It remains to be seen what kinds of applications can be created using the data obtained through our method, and which kinds are impossible.

We also started introducing ID-based sensor systems (RFID) to our environment. We would like to stress here that the purpose of introducing RFID is to be grounded clusters in actual properties. Thus, we do not presuppose the situation that every objects in the world will become RFID-equipped. One of our aims is to achieve more intelligent surroundings with less ID-equipped human and artifacts.

Acknowledgements

The authors would like to thank many people at our laboratory group for their generous help. This paper would not have been possible without the valuable discussions with Kensuke Fukuda, Susumu Shimizu, Kenichi Kourai, and Shigemi Aoyagi. Keiji Hirata and Yasunori Harada have made some excellent suggestions regarding our project. Minoru Kubota has encouraged us in our research. Suggestions of the reviewers helped improve this paper, and we are particularly grateful to them for their valuable comments.

References

[1] Bauer, M., Becker, C., and Rothermel, K., “Location models from the perspective of context-aware applications and mobile ad hoc networks”, In [2], pp. 35-40.

[2] Beigl, M., Gray, P., and Salber, D. (eds.), *Location Modeling for Ubiquitous Computing,* Workshop Proceedings, UbiComp 2001, 2001.

[3] Brumitt, B. and Shafer, S., “Topological world modeling using semantic spaces”, In [2], pp. 55-62.

[4] Bulusu, N., Estrin, D., and Heidemann, J., “Tradeoffs in location support systems: the case for quality-expressive location models for applications”, In [2], pp. 7-12.

[5] Gellersen, H.-W., Schmidt, A., and Beigl, M., “Adding some smartness to devices and everyday things”, In *Proceedings of the 3rd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2000),* pp. 3-10, 2000.

[6] Guttman, E., “Autoconfiguration for IP networking: enabling local communication”, *IEEE Internet Computing,* Vol. 5, No. 3, 2001, pp. 81-86.

[7] Harnad, S., “The Symbol Grounding Problem”, *Physica D,* Vol. 42, 1990, pp. 335-346.

[8] Hightower, J. and Borriello, G., *A Survey and Taxonomy of Location Systems for Ubiquitous Computing,* Technical Report UW-CSE 01-08-03, University of Washington, 2001.

[9] Hightower, J., Brumitt, B., and Borriello, G., “The location stack: a layered model for location in ubiquitous computing”, In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002),* pp. 22-28, 2002.

[10] Jain, A. K., Murty, M. N., and Flynn, P. J., “Data clustering: a review”, *ACM Computing Surveys,* Vol. 31, No. 3, 1999, pp. 264-323.

[11] Jiang, C. and Steenkiste, P., “A hybrid location model with a computable location identifier for ubiquitous computing”, In *UbiComp 2002 Proceedings*, LNCS 2498, pp. 246-263, 2002.

[12] Leonhardt, U.: *Supporting Location-Awareness in Open Distributed Systems,* PhD thesis, Imperial College, University of London, 1998.

[13] Ng, R. T. and Han, J., “CLARANS: a method for clustering objects for spatial data mining”, *IEEE Transactions on Knowledge and Data Engineering,* Vol. 14, No. 5, 2001, pp. 1003-1016.

[14] O'Connell, T., Jensen, P., Dey, A., and Abowd, G., “Location in the aware home”, In [2], pp. 41-44.

[15] Okabe, A. and Masuyama, A., “An exploratory method for qualitative trend curve analysis – a robust method against low quality data”, *International Journal of Geographical Information Science,* Vol. 15, No. 1, 2001, pp. 65-76.

[16] Schiele, B. and Antifakos, S., “Beyond position awareness”, In [2], pp. 107-112.

[17] Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., and Van de Velde, W., “Advanced Interaction in Context,” In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing (HUC 99),* LNCS 1707, pp. 89-101, 1999.

[18] Thomson, S. and Narten, T., *IPv6 Stateless Address Autoconfiguration,* RFC 2462, 1998.

[19] Thrun, S., *Robotic Mapping: A Survey,* Technical Report CMU-CS-02-111, Carnegie Mellon University, 2002.

[20] Weiser, M., “The computer for the 21st Century”, *Scientific American,* Vol. 265, No. 3, 1991, pp. 94-104.