Instant Learning Sound Sensor: Flexible Real-world Event Recognition System for Ubiquitous Computing

Yuya Negishi and Nobuo Kawaguchi

Graduate School of Engineering, Nagoya University
1, Furo-Cho, Chikusa-ku, Nagoya, 464-8601, Japan
negishi @ el.itc.nagoya-u.ac.jp, kawaguti @ nagoya-u.jp

Abstract. We propose a smart sound sensor for building context-aware systems that instantly learn and detect events from various kinds of everyday sounds and environmental noise by using small and low-cost device. The proposed system automatically analyzes and selects an appropriate sound recognition process, using sample sounds and a parameter templates database in the event learning phase. A user is only required to input target event sounds from a microphone or sound files. Using the proposed sensor, the developer of ubiquitous service can easily utilize real world sounds as event triggers to control appliances or human's activity monitors for presence services without a signal processing programming.

1 Introduction

Context-aware systems are beginning to play an important role in supporting human activities. Various input devices such as accelerometers, pressure and temperature sensors, and GPS are used to input context information into systems. For example, small network devices with these sensors have been developed (MOTE[1] Smart-Its[3]) and used to build a smart room. A health care system that senses a user's actions such as washing hands and taking a shower has been proposed by Jianfeng Chen[6]. Paul Lukowicz[10] proposed an activity sensing system for workshops that uses a microphone and accelerometers worn on the body. There has also been some research on systems that use signal processing with time series data from sensors like microphones and accelerometers that shows this kind of processing is useful in obtaining rich context information[7][8]. However, designing the recognition algorithm for a complex signal pattern is not easy because analyzing feature quantity requires a lot of time, and in the ubiquitous environment, it is assumed that there are many sensing targets.

We propose an instant learning sensor that can learn signal patterns instantly using a small inexpensive device. The proposed sensor can record and learn the time series signal pattern that a user wants to detect as an event. The instant learning sensor has the following three features.

(1) Instant Learning: On the sensor installation site, the sensor can instantly learn a signal pattern by user's demonstration of a target event. The

system automatically analyzes the recorded signal and chooses the most appropriate algorithm to extract feature quantity. This enables the user to build context-aware systems without signal processing programming.

(2) Smart Sensor: The sensor can process the signal by itself. It can also cooperate easily with other devices or context-aware systems.

(3) Simple Device: The sensor can be implemented on low-cost devices such as microcontrollers. So, we aim as much as possible at light processing. To make the device simple, we think a sensor only has to be able to recognize a few events and specialize in specific events that reduce the size of the memory and processing power.

We believe our instant learning sensor can simplify building of an advanced ubiquitous service that function as event notifiers. For example, in systems like eBlocks [12], that can easily construct smart environments simply by connecting block devices, instant learning sensors can be used as event triggers for control of appliances. They can be also used for more detailed sensing of peoples' activities when they are embedded into mobile phones or smart rooms and buildings.

We focus on instant learning and describe the prototype of the instant learning sensor, which can recognize environmental sounds and sounds made by humans and machines. Sound has very rich context information. We designed and implemented an instant learning sound sensor, which we embedded in a PC, and an inexpensive DSP microcontroller using a cheap piezoelectric device as a microphone.

2 Contributions

Obtaining information from the real world is one of the most important issues in ubiquitous computing. Current research on connecting the real and digitized worlds focuses mostly on specialized devices or algorithms for each type of real world event. However, there are an enormous number of types of real world events that must be recognized by these specialized devices or algorithms. It is almost impossible to cover all of these events by studying them one by one.

We propose an instant learning sound sensor built on an inexpensive device. Sound is one of the most rich context media. We believe that most real-world activities such as walking, opening and closing doors, using a vacuum cleaner, watching TV, pouring tea, and bathroom activities can be recognized merely by using the smart sound sensor. Of course, some of these events are easily recognized by sensing devices such as mechanical switches or motion detectors. However, we think it is costly and bothersome to develop a device or method for each event. Our smart sound sensor is a single inexpensive device, is very flexible, and has a wide range of applications, such as do-it-yourself smart spaces and rapid prototyping of context-aware systems.

The main contributions of this paper are summarized as follows.

(1) Proposing a simple sensor, which can only recognize a few sounds. This enables sound recognition in a low-cost device with less processing power and memory. If user wants to recognize several events, just use several sensors.

(2) Consideration of on-site configuration of sensing algorithms and parameters. This enables a single device to be flexibly utilized.

3 Signal Processing

3.1 Light-weight Sound Recognition Process

In this section, as an example of a light process for everyday sound recognition process, we describe signal processing using DP matching.

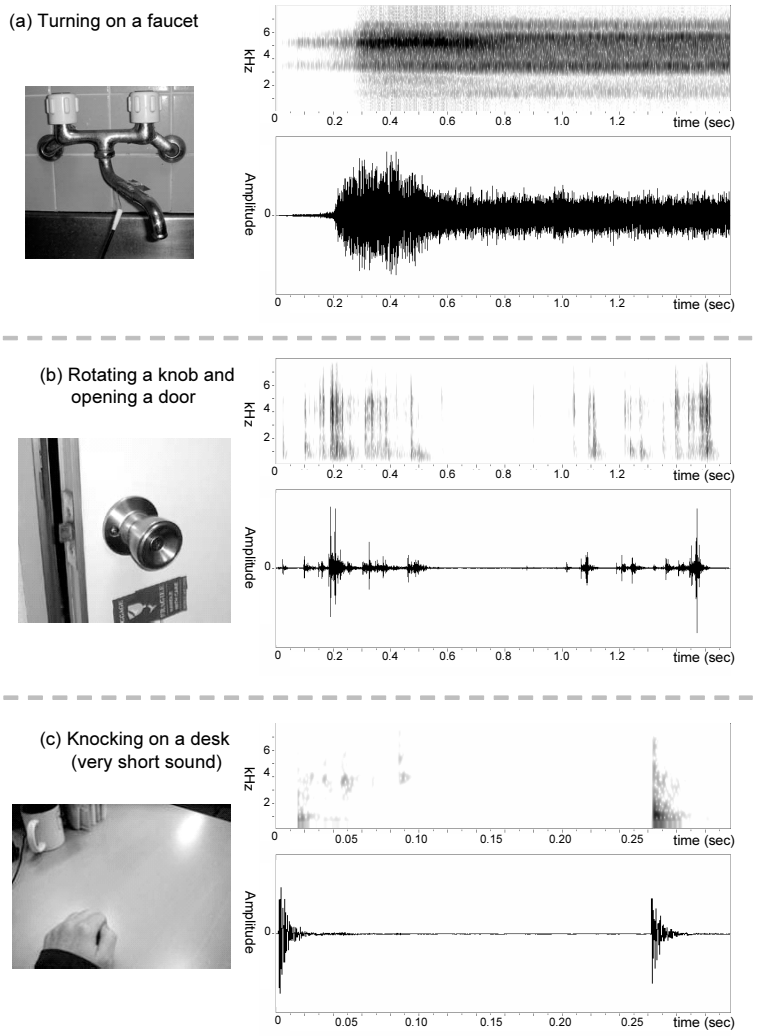

Fig. 1. Spectrograms and waveforms of environmental sound

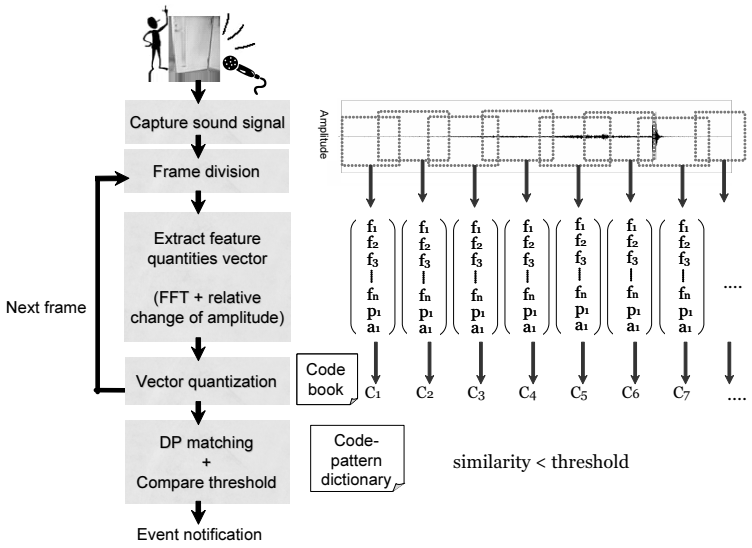

Fig. 2. Basic sound recognition process

A lot of speech recognition system use speech spectra such as mel-cepstrum or liner prediction coefficients (LPC) to extract feature quantities. We use a case study to discuss the signal processing for everyday sound recognition.

Figure 1 shows examples of waveforms and spectrograms of target sounds. First, we considered appropriate feature quantities for recognition of a sound. As can be seen in these figures, there are individual characteristics in each sound. Sound (a) in Figure 1 has a characteristic constant frequency over time. Sound (b) has a characteristic change of amplitude over time. Therefore, we think feature quantity should include not only frequency but amplitude characteristics. It is also be necessary to adjust weights between feature quantities for each target sound.

We designed a basic sound recognition process that calculates the similarity between an input sound and a target event sound using DP matching (dynamic programming matching[5], dynamic time warping[2]) about a change in frequency and amplitude characteristics. The process has some flexibility parameters. It can adjust weights between feature quantities and has options such as whether to enable a low pass filter. Figure 2 shows the overview of a recognition process, and Table 1 shows a list of parameters.

(1) First, it divides inputted waveform into frames. The length of each frame is F_{length} points, and the shift length is F_{shift} points.

(2) Second, in each frame, it extracts feature quantity vectors, calculates power spectra using fast Fourier transform (FFT), divides it equally to N_v sections, calculates the average of each section, and normalizes by maximum power. The range of each element is $0.0 \sim 1.0$. In addition to these N_v elements, it adds an amount of the change of power by the following expression as the $(N_v + 1)$th

Table 1. Parameter List

parameter	description
F_{length}	frame length
F_{shift}	frame shift length
N_v	dimension of frequency characteristics in feature quantity vector
W_f	weight of frequency characteristics in distance function
W_p	weight of power change characteristics in distance function
W_a	weight of amplitude characteristics in distance function
L	variable L in expression (1)
A_{max}	upper amplitude for normalization
$EnableLPF$	flag to enable a low pass filter
$CodebookSize\_S$	code book size of a stationary sound
$CodebookSize\_T$	code book size of a target event sound
$Threshold$	threshold for acceptance criterion by similarity of DP matching

element, and the maximum amplitude value in each frame as the (N_v+2)th element. L is a constant number. The range of the (N_v+1)th element is $0.0 \sim 1.0$, and the range of the (N_v+1)th element is $0.0 \sim 1.0$ using upper value A_{max}. Each feature quantity vector has a total of N_v+2 elements.

$$Vec(N_v+1) = \frac{Power_{max}(n) - Power_{max}(n-1)}{L} \tag{1}$$

(3) Third, it changes the vectors to code array using vector quantization[11] with a code book of target sound and a following distance function. In the expression, W_f is a weight for frequency characteristic, W_p is a weight for the amount of change in power, and W_a is an amplitude characteristic.

$$dis(v_1, v_2) = W_f \times dist_f(v_1, v_2) + W_p \times dist_p(v_1, v_2) + W_a \times dis_a(v_1, v_2) \tag{2}$$

$$dis_f(v_1, v_2) = \sum_{k=1}^{Nv} (v_1(k) - v_2(k))^2 \tag{3}$$

$$dis_p(v_1, v_2) = (v_1(N_{v+1}) - v_2(N_{v+1}))^2 \tag{4}$$

$$dis_a(v_1, v_2) = (v_1(N_{v+2}) - v_2(N_{v+2}))^2 \tag{5}$$

(4) Finally, using DP matching, it calculates the similarity between the inputted code pattern and one of the target event sounds. Then, it compares the similarity with the threshold for detection judgment.

These are processed in real time every F_{shift} points. The code book and code pattern of the target event sound are made in advance.

3.2 Parameter Configuration

Table 2 shows examples of manual configuration for the sounds in Figure 1. As is the case in Table 2, we think the configurations of the parameters should contain

Table 2. Example of parameter configurations

parameter name	sound (a)	sound (b)	sound (c)
F_{length}	256	128	512
F_{shift}	60	30	120
N_v	16	12	32
W_f	1.0	0.4	0.5
W_p	0	0	0.5
W_a	0	0.6	0
L	-	-	120.0
A_{max}	-	30000	-
$EnableLPF$	false	false	false
$CodebookSize\_S$	2	4	4
$CodebookSize\_T$	4	16	16

fewer vector dimensions and less codebook size for light processing and require less memory.

Our aim is to automatically generate a recognition program for unknown everyday events. This requires a search for specific and appropriate parameter configuration for each target event sounds. Therefore, we designed a method of flexibly configuring the feature quantities to select a proper set for the target sound. In the next section, we describe this flexible configuration method.

4 Design of Instant Learning Sound Sensor

4.1 System Architecture

We designed an instant learning sound sensor to have two modes (Figure 3).

(1) Event Learning Mode: In this mode, the system analyzes a target event sound that is inputted by a user, and automatically selects a specific parameter configuration. Then, it makes a codebook and code pattern dictionary for the target sound.

(2) Event Detection Mode: In this mode, the system monitors a target event sound and notifies other systems via a network if the event is detected.

The proposed system consists of the following two components.

(1) Sensor Configurator: This software supports the event-learning mode and has a function to install the selected recognition process and configuration in sensor devices.

(2) Sound Sensor Device: This is a device that runs the sound recognition process in event detection mode.

To simplify the sensor device, we implemented the sensor configurator in a powerful computer such as a PC.

As a microphone, we chose a piezoelectric device. This device is very cheap small and thin, so it can also be attached to almost any kind of object, for example a wall, a desk, a cup, a faucet, a book, a telephone, or a trash can.

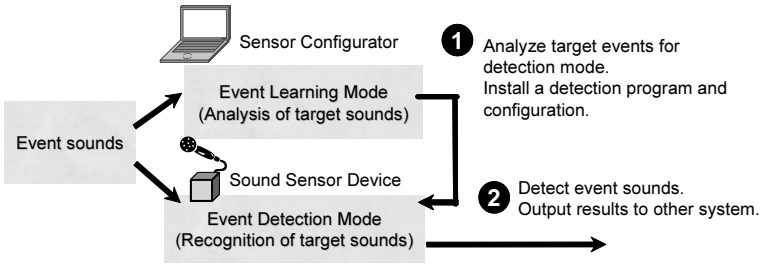

Fig. 3. System Overview

Generally, it is said that sound sensing using a microphone is easily disrupted by environmental noise and human speech. However, because a piezoelectric device is a contact sensor, we don't use noise-reduction process in current system.

4.2 Sample Sounds and Parameter Templates Database

In section3.2, we mentioned the need for specific parameter configurations for each event sound. The most naive configuration method is to try all combinations of parameters, but this takes a very long time. Therefore, we collected various environmental and everyday sounds as samples and built a sample sounds database that uses sample sounds and corresponding parameter configurations as templates. The template is an appropriate set of parameters for the light recognition process and is made in advance by such as a full search. In the event learning mode, the sensor configurator utilizes the templates of similar sample sounds with a target event for parameter configurations.

4.3 Instant Learning

The sensor configurator processes the event-learning mode using the sample sounds and parameter templates database. Figure 4 shows the process.
(1) The user inputs a target event sound, then the sensor configurator calculates similarities between sample sound in terms of frequency (sim_f) and amplitude (sim_a) characteristics using the following expressions.

$$sim(s_{sample}, s_{target}) = sim_f(s_{sample}, s_{target}) \times sim_a(s_{sample}, s_{target}) \quad (6)$$

sim_f is the similarity between two N dimension frequency characteristic vectors of amount of change in power on each band in the whole. It uses the vector space model[4].

$$sim_f(s_{sample}, s_{target}) = \frac{\sum_{n=1}^{N} u(n)w(n)}{\sqrt{\sum_{n=1}^{N} u(n)^2 \sum_{n=1}^{N} w(n)^2}} \quad (7)$$

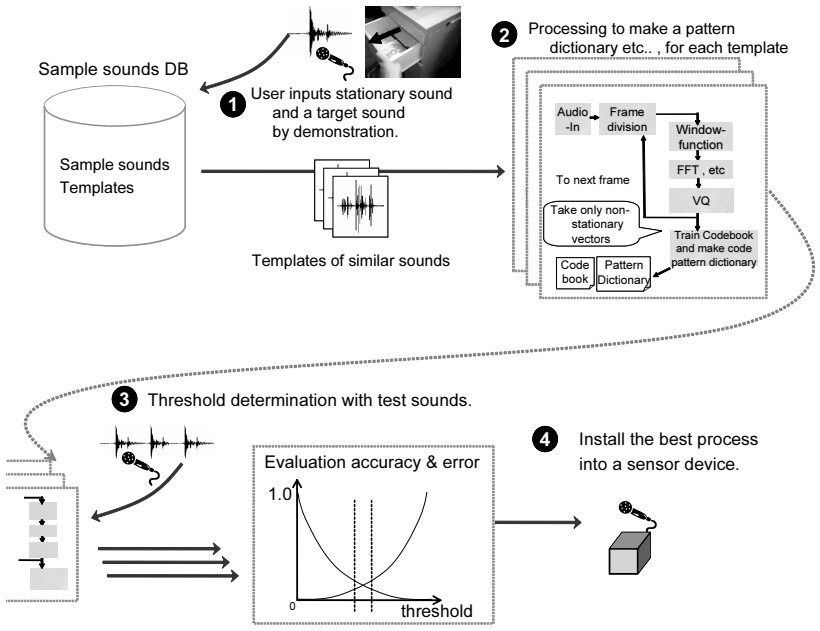

Fig. 4. Process in Event-Learning Mode

sim_a is a similarity between two time series changes of amplitude, using DP matching cost($dtwCost$).

$$sim_a(s_{sample}, s_{target}) = \frac{1}{dtwCost(s_{sample}, s_{target}) + 1} \tag{8}$$

Both similarities are normalized, so the range of $sim(s_{sample}, s_{target})$ is $0.0 \sim 1.0$. The bigger value means more similar.

(2) Second, the sensor configurator sorts all sample sounds according to their similarities, and selects some templates corresponding to superior sample sounds. Then, based on parameter configurations of selected templates, it makes code books using the LBG algorithm[13] and code pattern dictionaries of target sounds. In the prototype, the sensor configurator tries four templates.

(3) Third, the sensor configurator determines the threshold of cost in DP matching in each parameter configuration. Using target event sounds and non-target sounds, it gets the DP matching set of costs for both target and non-target sounds. The threshold is determined by the following two expressions. $P(S|n)$ is the rate of true detection when non-target sounds are inputted, and $P(N|s)$ is the rate of false detection when target sounds are inputted. The relation between the two expressions and the threshold is shown in Figure 4-(3). As can be seen in Figure 4-(3), the best threshold is estimated at around the point where $P(S|n) = P(N|s)$. For non-target sounds, it uses other sample sounds in the database.

$$P(S|n) = \frac{(Number\ of\ accepted\ non\ target\ sounds)}{(Total\ of\ non\ target\ sounds)} \tag{9}$$

$$P(N|s) = \frac{(Number\ of\ rejected\ target\ sounds)}{(Total\ of\ target\ sounds)} \tag{10}$$

(4) Finally, the sensor configurator evaluates rates of accepted non-target sounds in each parameter configuration, comparing the result of the previous step. After the evaluation, it chooses the lowest error configuration and installs them in sound sensor devices.

To execute the above process, the user is only required to prepare some recorded target event sounds. The sensor configurator automatically selects an appropriate recognition process.

5 Prototype Implementation on a PC

5.1 Sensor Configurator and Sound Sensor on a PC

Based on the above design, we implemented the first prototypes of the sensor configurator and sound sensor on a Windows PC using C language.

Sensor Configurator Figure 5 is a screenshot of the sensor configurator. The user simply (1) prepare recorded target event sounds as WAVE files, (2) drag & drop them to the sensor configurator, and (3) push the learning button with automatic template selections and evaluations. In this prototyping, the sampling frequency is 12-16 kHz, and there are 16 sampling bits.

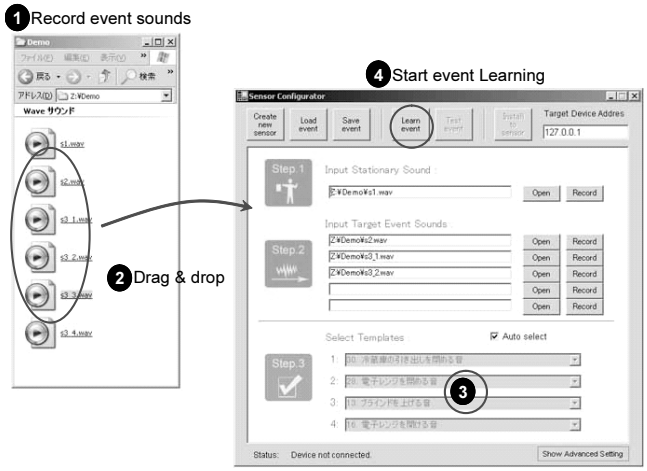

Fig. 5. Screenshot of Sensor Configurator

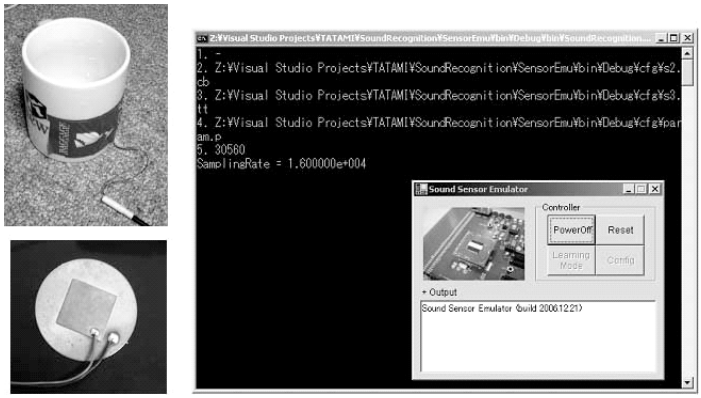

Fig. 6. Screenshot of Sound Sensor on PC

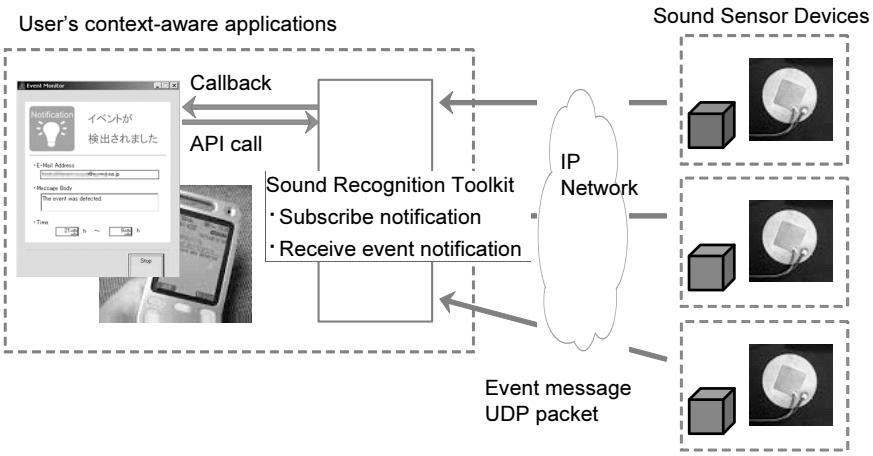

Fig. 7. Example of useage of Sound Recognition Toolkit

For the sample sounds and parameter templates database, we collected 30 events, each with about 50 sounds, for a total of about 1500 sounds. In this implementation, we adjusted parameter templates of sample sounds as light as possible by our hands. As a way of recording sounds, we attached a piezoelectric device to objects like Figure 1 and 6, then we repeated actions and captured event sounds. Table 3 lists examples of recorded sample sounds.

Sound Sensor Figure 6 is a screenshot of the sound sensor and a photo of an attached piezoelectric device on a coffee cup. On target event detection, it can send notification UDP packets to other systems through an IP network.

5.2 Sound Recognition Toolkit

We also developed a communication API library with the sound sensors on PCs via the Windows DLL IP network. A user's program can receive callback when event sounds are detected. This library can be used as shown in Figure 7.

6 Evaluation of Instant Learning

We evaluated the recognition rate of the proposed instant learning methods, using the prototype system on a PC.

6.1 Recognition rate of optimized parameters in templates

Before evaluating the instant learning with several event sounds, we evaluated the sound recognition processes with optimized templates as discussed below.

(1) Recognition rate: We inputted about 50 correct event sounds and counted a number of accepted correct sounds.

$$Recognition\ rate = \frac{accepted}{correct} \tag{11}$$

(2) Error rate: We inputted 120 incorrect sounds (24 types of event five times each), and counted a number of accepted incorrect sounds.

$$Error\ rate = \frac{accepted}{incorrect} \tag{12}$$

The results are listed in Table 3. For some sounds, like the sound of writing on a clipboard, the error rate was high. We think these sounds need other recognition algorithms or feature quantities. However, using an optimized parameter, the system can recognize the corresponding correct sounds, with 80 to 100% accuracy.

6.2 Recognition rate of unknown event sounds

Next, we evaluated the accuracy of the recognition process automatically generated by instant learning system.

(1) Removing four types of sample sounds and templates from the sample sounds database, we inputted those types of event sounds to the sensor configurator and got each recognition process.

(2) Using automatically generated recognition processes, we inputted about 50 correct sounds and 120 incorrect sounds and measured recognition and error rates, as in the previous evaluation.

The results are listed in Table 4. As with the optimized templates, the recognition rates are over 83%, meaning the system can recognize correct events sounds well.

These results show that our proposed method is useful for choosing automatically specific and appropriate parameters for environmental and everyday sounds.

Table 3. Results of evaluation with optimized templates

Event	Recognition rate	Error Rate
1. Turning on a faucet	100.0%	0.0%
2. Closing a sliding door	100.0%	1.7%
3. Closing door of refrigerator	100.0%	2.5%
4. Opening a window shade	100.0%	15.8%
5. Rotating a knob and opening a door	88.9%	0.0%
6. Pouring out tea	86.8 %	5.0%
7. Putting a phone down	86.4%	5.8%
8. Writing on a clipboard	85.6%	65.8%
9. Opening a drawer	83.3%	0.0%
10. Inserting a plug in an outlet	83.3%	0.0%

Table 4. Results for instant learning

Event	Recognition rate	Error Rate
1. Typing on a keyboard	100.0%	35.0%
2. Opening a window shade	96.9%	0.0%
3. Writing on a clipboard	85.6%	35.8%
4. Opening a drawer	83.3%	1.7%

7 Sound Sensor on Small and Low Cost Device

In the next step, we implemented the second prototype sound sensor device on a Microchip dsPIC[9] to evaluate its feasibility on a small inexpensive device.

In this prototype on the DSP microcontroller, we use a Microchip dsPIC30F60 14A with 8 KB Data RAM and a Silicon Laboratories Si3000 codec chip with 12 KHz sampling frequency and 16 sampling bits. We also used a SENA Parani ESD200 Bluetooth communication module for event notification. Figure 8 is a photo of the prototype sensor board. To program target event sound recognition processes to microcontrollers, a user can use codebooks, code pattern dictionaries, and parameter configurations header files for dsPIC C compliers converted by the sensor configurator.

For real-time recognition, the system needs to complete per-frame works within $FrameShiftLength/SamplingFrequency$ (= 5 msec). For example, a specific configuration of the sound recognition process of (a) turning on a faucet, shown in Figure 1, is (a) in Table 2. In this example configuration, we confirmed that per-frame work can be done in an average of $3.49msec$, with 58.94MHz (14.74MIPS) system clock and 60 code pattern length. The required RAM size can also come to less than 4KB of memory use. The total code size including codebook, and pattern dictionary, is about 10 KB. We also confirmed real-time recognition for the sound of opening a drawer using this device.

This implementation demonstrates the feasibility of the small inexpensive instant learning sound sensor.

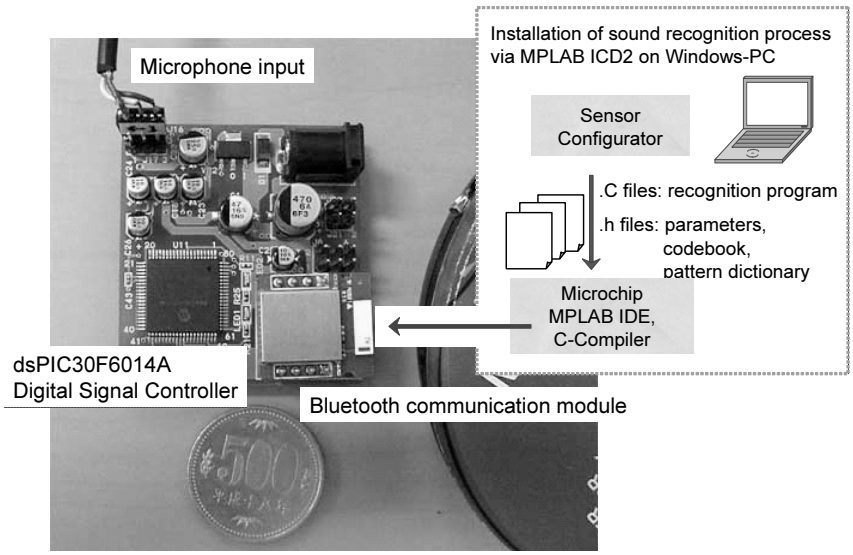

Fig. 8. Prototype of Sound Sensor Device with dsPIC

8 Conclusion

In this paper, we describe the design and implementation of an instant learning sound sensor that makes it easy to build a context-aware system using sound recognition. The proposed learning sensor can instantly learn and detect target event sounds such as everyday sounds and environmental noise. The sensor configurator automatically analyzes target event sounds and chooses the most appropriate feature quantities and parameter configuration for high recognition rate and light processing using sample sounds and parameter templates DB. Therefore, using the instant learning sensor, a user can build context-aware systems that utilize real world sound as rich context information, without signal processing programming. Users can also easily integrate the sound recognition function with their systems as event notifiers. We have confirmed that the recognition process in this prototype system can run on a small and low cost device.

In future work, we are currently working on design and implementation of small networked instant learning sound sensor devices, and considering a sensor fusion method to treat complicated sounds. We will support more kinds of feature quantities and algorithms for various sounds recognition, and collect more sample sounds and templates. Furthermore, we will extend an instant learning method to the other sensors such as accelerometer.

References

1. Crossbow Technology, MOTE, http://www.xbow.com/

2. C. S. Myers and L. R. Rabiner: A comparative study of several dynamic time-warping algorithms for connected word recognition, IEEE Trans. on ASSP, Vol. 26, No. 3. (1981) 351-363.
3. Emmanuel Munguia Tapia, Stephen S. Intille, Louis Lopez, and Kent Larson: The design of a portable kit of wireless sensors for naturalistic data collection, Pervasive 2006. (2006) 117-134.
4. G. Salton, A. Wong and C. S. Yang: A vector space model for automatic indexing, Communications of the ACM, Vol.18, No.11. (1975) 613-620.
5. H. Sakoe and S. Chiba: A Dynamic Programming Algorithm Optimization for Spoken Word Recognition, IEEE Trans. on ASSP, Vol. 26, No. 27. (1978) 43-49.
6. Jianfeng Chen, Alvin Harvey Kam, Jianmin Zhang, Ning Liu, Louis Shue: Bathroom Activity Monitoring Based on Sound, Pervasive 2005. (2005) 47-61.
7. Ling Bao and Stephen S. Intille: Activity Recognition from User-Annotated Acceleration Data, Pervasive 2004. (2004) 1-17.
8. Ling Ma, Dan Smith and Ben Milner: Environmental Noise Classification for Context-Aware Applications, Database and Expert Systems Applications. (2003) 360-370.
9. Microchip Technology Inc, dsPIC Digital Signal Controllers, http://www.microchip.com
10. Paul Lukowicz, Jamie A. Ward, Holger Junker, Mathias Stager, Gerhard Troster, Amin Atrash and Thad Starner: Recognizing Workshop Activity Using Body Worn Microphones and Accelerometers, Pervasive 2004. (2004) 18-32.
11. Robert M.Gray: Vector Quantization, IEEE ASSP Magazine, Vol. 1. (1984) 4-29.
12. Susan Cotterell, Ryan Mannion, Frank Vahid and Harry Hsieh: eBlocks - An Enabling Technology for Basic Sensor Based Systems, IPSN Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems, SPOTS. (2005).
13. Yoseph Linde, Andres Buzo and Robert M.Gray: An Algorithm for Vector Quantizer Design, IEEE Trans. on Comm, Vol. 28, No. 1. (1980) 84-95.