"LISTENIN" TO DOMESTIC ENVIRONMENTS FROM REMOTE LOCATIONS

Chris Schmandt and Gerardo Vallejo

M.I.T. Media Lab
20 Ames St.
Cambridge, MA 02139
{geek, gvallejo}@media.mit.edu

ABSTRACT

This paper describes ListenIn, work in progress using audio as a monitoring medium, with emphasis on domestic environments inhabited by elder parents. The primary goal of this monitoring is to provide a continuous but peripheral awareness of the monitored site and a remote location, or to a mobile user. Sound gathering and classification in the home is done in a distributed architecture server with multiple components. At transitions of activity, as measured by change in sound, a remote server receives and plays a few seconds of an "iconic" sound, the actual sound, or a "garbled" version of the actual sound, depending on confidence in the classification and whether speech is detected.

1. INTRODUCTION

With the increased attention to terrorism there is interest as well as concern by many parties about monitoring activities of people by various means. With very different motivations, we also see desire on the part of family members and caregivers to monitor the well-being of children and elders. A variety of sensing technologies are being employed in these efforts, with much emphasis on video, e.g. "nanny cams" and the ubiquitous advertisements on the web for remote controlled miniature cameras.

This paper describes ListenIn, work in progress using audio as a monitoring medium, with emphasis (based on the authors' personal lives) on domestic environments inhabited by elder parents. We suggest that audio has more utility than video and may be less intrusive. We discuss both the information gathering side of monitoring (the actual eavesdropping process) as well as the presentation side, or how the gathered acoustic information is to be presented to the care-giver or concerned family members. The primary goal of this monitoring is to provide a continuous but peripheral awareness of the monitored site and a remote location, or to a mobile user.

Our approach is server-based, with distributed digital microphone sensors feeding a hierarchical sound classifier in a distributed manner. The server resides in the home being monitored, and its algorithms decide which acoustic information is allowed to leave the premises. Ideally the server would classify all sound activity, and whenever a transition in activity is sensed, transmit an iconic version of the sound rather than the actual real audio data. This helps provide a measure of privacy. If the iconic sounds are cached at the client, the server need only tell it which sound to play.

Because we currently cannot classify these sounds with adequate performance, and there will always be novel sounds in a domestic environment, sound which cannot be classified are simply transmitted for a few seconds as samples. When speech is detected in this signal, we first modify (or "garble") it in a way in which one can tell it is speech and identify the talkers if they are well known, but understand little of what is said except the overall tenor of the conversation.

Rather than continuously stream audio to the remote monitoring client, the audio, be it real, garbled, or iconic, is transmitted only at acoustically significant transitions. This is meant to provide an ambient or background sense of presence of the monitored locale, rather than a continuous listening in. Remote clients may be PCs or mobile telephones.

In this paper we describe the architecture of the scale monitoring system and the remote user-interface design. This is work in progress and only portions of it have been implemented at the time of this writing. We describe two completed portions of the monitoring application in detail, and the audio garbling algorithm.

2. WHY AUDIO?

For monitoring domestic environments, both video and various forms of motion sensors have been used. It is also suggested that networking various household appliances can also be used as a measure of domestic activity. We believe there are some strong advantages for audio.

First, it is easier to place microphones than cameras or motion sensors, where field of view is a major consideration. Microphones are more nearly omni-directional than even wide-angle camera lenses, and because sound is more accurately reflected off more surfaces than light, there are fewer "blind" spots when eavesdropping instead of peeping in. Microphones may be more easily hidden, or much less visible. While we are not advocating surreptitious monitoring, residents of a monitored domicile may be more comfortable without the monitoring technology in their faces.

Second, audio requires significantly less bandwidth to transmit. Our architecture calls for multiple points of capture in the house, with wireless (802.11b) transmission to a server. We also wish to support mobile clients such as phones on current wireless telephone networks. Video, even slow at slow frame rates, may tax such networks.

Finally, detection is only part of the problem. We wish to enable a constant but minimally distracting awareness of the monitored environment at a remote location. A video window

21o21n a computer screen requires visual contact, i.e., being close to the screen and looking at it. Auditory cues can be heard regardless of where the recipient is looking. Even if a computer user is monitoring via a small video window in the corner of his or her screen while doing other work, motion in one's peripheral vision evokes a reflex to look at the source of motion, and would be continually distracting. If the monitoring person is mobile (walking, driving, shopping) non-intrusive auditory cues might be acceptable, but any sort of visual display will not be manageable.

3. ARCHITECTURE

ListenIn uses a client-server architecture, with the server PC located in the home being monitored, and IP-based clients, either mobile or stationary, at remote locations.

3.1. Classification - server

ListenIn uses a distributed and layered architecture for sound capture and classification (figure 1). Wireless IP-based digital microphones are distributed around a house, approximately one per room, and communicate to the server via WiFi networking. In the current implementation these mics are actually iPaq PDAs with some simple control software. Rather than continually stream audio to the server, this software sends a message to the server when the audio level it is monitoring changes significantly, and the server then requests streaming audio in near real time (network congestion may cause delays; this is why the server controls transmission). Although the iPaq is powerful enough to do some initial classification (see discussion of the baby monitor below), which would reduce bandwidth use still more, we prefer to design to an implementation using less expensive components and computation will be less expensive on the server. The server may also use the raw audio for alerting.

Sound classification is far from perfect, and we would like to be able to deploy ListenIn without having to acquire and label extensive training data for each home in which it is deployed. We use fairly generic audio classification based on either the nearest feature line (NFL) in which information provided by multiple prototypes per class is explored [1] or the well-known Nearest Neighbor (NN). To represent audio these methods use perceptual and cepstral features and their combinations, and both are easy to implement and tune. When sound can be classified with confidence, then the server simply sends the client a pointer to a generic "iconic" sound which represents the activity being detected; in other words, if "baby crying" is detected, the server uses a generic baby cry sound, which helps preserve the privacy of the monitored residence.

The alerting model is compatible with mediocre sound classification; if we cannot classify a sound we simply transmit samples to the remote client. As sound classification improves, more representative audio icons and less of the real sound will be transmitted from the house. To detect speech we use a speech/non-speech discrimination algorithm for noisy environments; in essence this algorithm detects vowels within sampled audio. The algorithm first detects bursts of energy of duration 50 to 500 milliseconds within the frequency band of 50 – 1200 Hz. It then convolves the samples corresponding to this burst with the remaining audio samples within a window; the resulting peaks indicate similarity, which are taken as the presence of vowels. When speech is detected in the audio to be transmitted the outgoing audio is pre-processed in an attempt to minimize the intelligibility of the conversation.

There are a number of audio events in a home which are difficult to classify or differentiate. For example, it may be hard to distinguish the sound of the television from people having a conversation in the living room (one approach is to use directional microphone arrays, but this requires more careful placement and mapping of the room). The shower may sound like an egg frying, and as the radio switches between talk and music, or as the CD player advances to the next track, we prefer not to signal a new acoustic event.

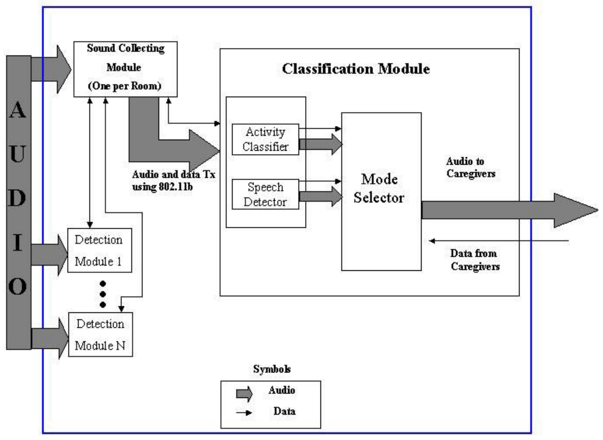

Figure 1. *The distributed architecture of ListenIn.*

Because fixed sources of sound in the home are limited, we augment the per-room microphone with low power local sound sensors. The server then fuses this data with the output of the classifier, giving high weighting to the local sensors.

The local sensors are small battery-powered boards with inexpensive microphones and radio transmitters similar to garage door remote controls. These sensors, built for another project, simply transmit on or off signals, depending on whether they hear sound or not. The sensor package is small and is meant to be placed directly on the surface to be sensed, e.g. the speaker enclosure of the home stereo, the surface of the stove or counter next to the sink, or on a shower stall or toilet tank. Encased in material such as modeling clay and set for low sensitivity, these sensors are highly reliable indicators of activity precisely at their location.

The server PC in the home gathers the inputs from the remote microphones and small sound sensors, selects which microphone to listen in on, and classifies the sound if possible. When a transition to a new sound is sensed (including the sound of "silence"), a message is sent via IP to the remote client(s), and optionally about 10 seconds of sound associated with the event, or an iconic sound (if the client has not cached these already). The server also provides access control, and may process the sound while streaming it to the remote client.

3.2. Alerting - client

Several versions of client software have been created; all connect to the server via IP. The desktop client connects to the server and receives audio events. If the event is classified by the server, the server sends a string describing the event so the client can play a locally cached iconic sound representing it. If the client cannot cache the iconic sound, it is transmitted from the server. The client simply plays either the iconic sound or whatever audio samples are sent by the server. The user can control playback volume, duration of each play event (up to the maximum of 10 seconds sent by the server), and optionally filter to exclude certain common events. Ultimately the client will also support an interface to query the server for the past occurrence of some event or to hear a brief audio history of recent activity, playing a number of sounds sequentially.

A mobile client under Linux on an iPaq using WiFi networking is essentially the same software as the desktop client. More interesting is a client based on Motorola iDen mobile telephones; the phones support fairly low rate IP connectivity through Nextel, and run Java. The Java client connects to the server over IP. Currently these phones cannot play sampled sounds, so events from the server play tones; such phones will soon be able to play sampled audio. This client is most useful when trigger sound is properly classified as the slow IP data rate imposes a delay when sampled sound is played, but iconic sounds can be cached on the phone.

4. CLASSIFICATION APPLICATIONS

Although the full ListenIn system as described above is not yet been finished, portions with limited sound classification were written for Impromptu [2]. Impromptu is a client-server based runtime environment for accessing multiple audio applications from a single mobile voice-over-IP player.

4.1. CryBaby

The most complete fragment of ListenIn is a baby monitor application, Cry Baby. For this application, the audio sampling, trigger, and classification functions have been built into a single server module running on an iPaq, rather then the fully distributed model described above. The baby cry detector continuously digitizes audio and analyzes it for a pattern characteristic of crying babies, i.e. a fairly loud sound followed by relative quiet as the infant inhales. Our detector requires at least three cry cycles, where a cry is defined as between 400 and 2000 milliseconds of sound above a threshold, interspersed with 200 to 1000 milliseconds of quiet between the cries (figure 2). The threshold reflects background noise level, and is allowed to vary slowly; the lawn mower outside the window will not trigger the baby monitor but the baby will have to cry more loudly to compete with its noise.

When a cry is detected, the baby monitor server sends an alert message to the mobile Impromptu client, which plays an iconic baby cry sound. Impromptu allows the user to respond by activating the application and then establishes a full duplex audio channel to the baby's location. A different baby monitor client, independent of the rest of the Impromptu architecture, was written in Java for an iDen mobile phone.

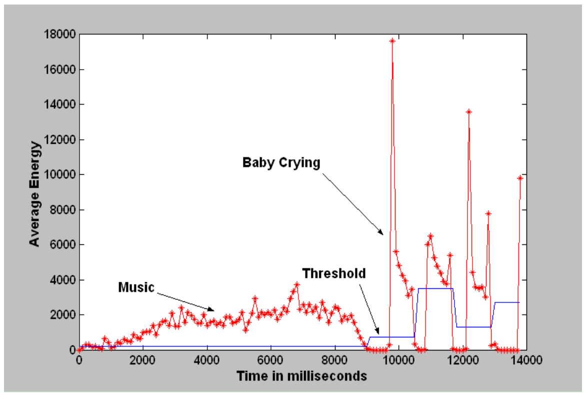

Figure 2.. *Baby cry detection. In response to the music, the base threshold is increased. Noise and quiet periods trigger the detector after the third cry.*

4.2. WatchDog

A simpler classifier in the same architecture alerts the remote user to a loud noise. Sudden increase in sound energy (the background threshold is again allowed to vary slowly) triggers the iconic sound of a dog barking. If activated, the server plays the stored samples of that sound (which may have been transient) to the client and then enables full duplex audio.

5. GARBLING AUDIO

If we cannot classify monitored sound, but determine that it contains speech, it is processed before transmission. The goal is to allow the listener to identify the talker(s), the tenor of the conversation, without understanding its actual content.

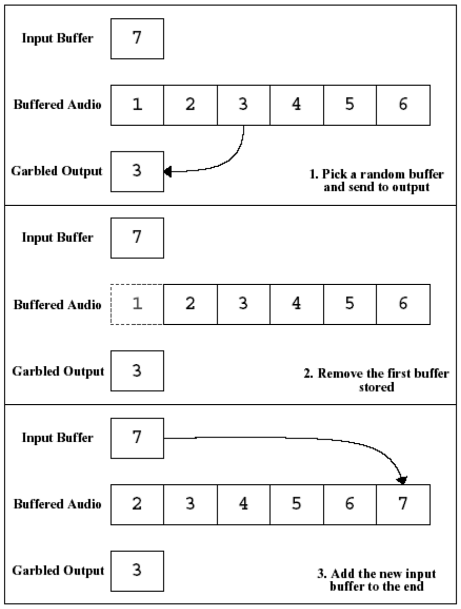

Figure 3.*Block shuffling garbling algorithm.*

The audio is “garbled” by shuffling 100 millisecond blocks of randomly, with cross-fading at block boundaries. The block size allows a few syllables to pass at a time, and hence conveys some affect and intonation. Every 100 milliseconds, one of the previous six blocks is randomly selected, as shown in figure 3. Then the least recent old block is deleted, and the just sampled block added to the candidates for future transmission. The cross fading requirement actually inserts a one block delay, but this is not noticed. Examples of garbled audio may be found at http://www.media.mit.edu/speech/projects/garblephone.html.

6. RELATED WORK

Most of the related work is in awareness management in computer mediated communication systems. Hubbub [7] is an audio-based messaging and group awareness system which automatically notifies others when a group member becomes active, by playing a sound specific to that person. Smith and Hudson [3] use the audio energy of a person's current conversation to modulate a stored acoustic profile of that person to create a minimally disturbing background audio presence. By contrast, Thunderwire [4] was an always on full-duplex audio channel among a work group; ListenIn is designed towards a much more peripheral awareness of the remote location.

There is little work on ambient auditory awareness of domestic environments, but Casablanca [5] explored several alternatives. Some of the earliest work on audio monitoring in a shared environment was Gaver’s ARKola [6]; more recently Kilander [8] used a more subtle approach.

7. CONCLUSIONS

We have presented ListenIn, a means of sharing or monitoring a remote environment through auditory detection, classification, and transmission between a server and several forms of remote client. This paper has focused on the technology used for this approach; real field deployment will be required before we can adequately evaluate the various aspects of ListenIn: interplay between the various classification components, classification performance, and effectiveness of the remote ambient auditory display. Certainly the interface will need tuning, but our prototype applications have been encouraging

It is appropriate to close with a few words on privacy. These days it is easy, and rightfully so, to be concerned about who has access to what information about us. Although it could be abused, the technology discussed herein would clearly be subject to regulation under wiretap laws. But we imagine deploying it only with the consent of the monitored party, or for parents tracking their own children. Why would anyone agree to be monitored? Conversely we might ask, who would really want to listen to all the noise of domestic life? By providing safer environments children might have more confidence that their elder parents can live independently until later in life. Couples may wish to share their domestic lives across work/home environments. We already accept monitoring of our children for their own safety; we may also wish to monitor homes or pets while at work or on vacation.

Digital, IP-based technology allows for easy monitoring at a distance or while mobile. We have discussed how classification on site can help guard privacy by presenting iconic rather than sampled sounds, and can process sounds containing speech to minimize intelligibility. As classification improves less sampled sound would leave the home. Of course this is reassuring only if the person being monitored trusts the technology to actually withhold the sampled audio data.

8. REFERENCES

[1] Li, S.Z. "Content-based classification and retrieval of audio using the nearest feature line method," *IEEE Trans. ASSP:* 619-625, 2000.

[2] C. Schmandt, J. Kim, K. Lee, G. Vallejo, M. Ackerman. "Mediated Voice Communication via Mobile IP," *UIST International Conference:* 141-150, 2001.

[3] I. Smith and S. Hudson. "Low disturbance audio for awareness and privacy in media space applications," *Proceedings of the ACM Conference on Multimedia*: 91–97, 1995.

[4] M. Ackerman, D. Hindus, S. Mainwaring, and B. Starr. "Hanging on the ’Wire: A field study of an audio-only media space," *ACM Transactions on Computer-Human Interaction*, 4(1): 39–66,1997.

[5] D. Hindus , S.D. Mainwaring , N. Leduc , A. Hagström , O. Bayley. "Casablanca: designing social communication devices for the home," *Proc. SIGCHI conference on Human factors in computing systems*: .325-332, 2001

[6] W. Gaver , R. B. Smith , T. O'Shea. "Effective sounds in complex systems: the ARKOLA simulation," *Human factors in computing systems conference proceedings on Reaching through technology*: 85-90, 2001.

[7] E. Isaacs, A. Walendowski, and D. Ranganthan. "Hubbub: A sound-enhanced mobile instant messenger that supports awareness and opportunistic interactions, " *Proceedings of Human Factors in Computing Systems (CHI.02). ACM*, 2002.

[8] F. Kilander and P. Lonnqvist "A Whisper In The Woods - An Ambient Soundscape For Peripheral Awareness of remote processes," *Proceedings ICAD* 2002.