SoundSense: Scalable Sound Sensing for People-Centric Applications on Mobile Phones

Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury and Andrew T. Campbell
Department of Computer Science Dartmouth College
{hong, pway, niclane, campbell}@cs.dartmouth.edu,
tanzeem.choudhury@dartmouth.edu

ABSTRACT

Top end mobile phones include a number of specialized (e.g., accelerometer, compass, GPS) and general purpose sensors (e.g., microphone, camera) that enable new people-centric sensing applications. Perhaps the most ubiquitous and unexploited sensor on mobile phones is the microphone – a powerful sensor that is capable of making sophisticated inferences about human activity, location, and social events from sound. In this paper, we exploit this untapped sensor not in the context of human communications but as an enabler of new sensing applications. We propose SoundSense, a scalable framework for modeling sound events on mobile phones. SoundSense is implemented on the Apple iPhone and represents the first general purpose sound sensing system specifically designed to work on resource limited phones. The architecture and algorithms are designed for scalability and SoundSense uses a combination of supervised and unsupervised learning techniques to classify both general sound types (e.g., music, voice) and discover novel sound events specific to individual users. The system runs solely on the mobile phone with no back-end interactions. Through implementation and evaluation of two proof of concept people-centric sensing applications, we demostrate that SoundSense is capable of recognizing meaningful sound events that occur in users' everyday lives.

Categories and Subject Descriptors

C.3 [**Special-Purpose and Application-Based Systems**]: Real-time and embedded systems

General Terms

Algorithms, Design, Experimentation, Human Factors, Measurement, Performance

Keywords

Audio Processing, Mobile Phones, Sound Classification, People-centric Sensing, Urban Sensing

MobiSys'09, June 22–25, 2009, Kraków, Poland.
Copyright 2009 ACM 978-1-60558-566-6/09/06 ...$5.00.

1. INTRODUCTION

We are on the brink of a new era in the development of the ubiquitous mobile phone. Many top end mobile phones [28] [4] [17] now come with GPS, WiFi and cellular localization and embedded sensors (e.g., digital compass, proximity sensors, and accelerometers). These mobile devices are leading to the emergence of new applications in health care, gaming, social networks, and recreational sports - enabling people-centric sensing applications [11] [1] [10]. Perhaps one of the most overlooked "sensors", available not only on top end phones but also on every mobile phone, is the microphone. Along with the camera, the microphone is one of the two most ubiquitous mobile sensor. This paper focuses on turning the microphone on mobile phones into a personalized sound event sensor capable of supporting a myriad of people-centric sensing applications.

Sound captured by a mobile phone's microphone is a rich source of information that can be used to make accurate inferences about the person carrying the phone, their environments and social events. This single modality is capable of supporting a diverse set of inferences, such as: conversation detection, activity recognition, location classification [30] [14], discovery of social network structure [13], and even classification of dietary intake of a person [2]. Moreover, the microphone is robust to a range of phone contexts (e.g., body position of the device). Many modalities are rendered useless by particular phone context (e.g., the camera when the phone is in a pocket or the accelerometer when the phone is placed on a table or in a bag). The information available from an audio stream degrades more gracefully (e.g., sound heard through a muffled microphone in the pocket) and is still useful albeit with a diminished capacity. Despite the potential of sound as a robust source of information for sensing applications, the microphone and inference based on sound has received little attention in the literature in comparison to other sensors such as the GPS and accelerometers.

In this paper, we propose SoundSense, a scalable sound sensing framework for mobile phones. SoundSense represents the first general purpose sound event classification system designed specifically to address a number of system design challenges presented by mobile phones: (i) scaling well to large number of people, each of whom may have different everyday sound environments, (ii) operating robustly under various phone context conditions, and finally (iii) allowing the phone to function unhindered, where the algorithms must be simple enough to run on resource constrained mobile phones yet be effective and make an attempt to safeguard the privacy of users.

SoundSense leverages the existing body of work on acoustic signal processing and classification (e.g., [24] [34] [21] [14]) and takes a systems design approach to realizing the first such system for a mobile phone. A key design goal of SoundSense is the scalability of classification to a large population. Specifically, the contributions of this paper are as follows: (i) we propose an architecture and a set of algorithms for multistage, hierarchal classification of sound events on mobile phones; (ii) we address the scaling problem through the introduction of an adaptive unsupervised learning algorithm to classify significant sound events in individual users' environments; and finally (iii) we implement the SoundSense system architecture and algorithms on the Apple iPhone, profile resource consumption in terms of CPU and Memory usage, and evaluate the performance of the algorithms using real world audio data sets and through the implementation of two proof of concept people-centric sensing applications.

The paper is structured as follows. Section 2 discusses the design considerations of implementing SoundSense on resource limited mobile phones. Section 3 presents the SoundSense architecture based on a multi-staged, hierarchal classification model. This is followed in Section 4 by a detailed presentation of the SoundSense classification algorithms. Section 5 discusses the implementation of SoundSense on the Apple iPhone and Section 6 presents a detailed evaluation of the system and two proof-of-concept applications. Section 8 presents some concluding remarks.

2. DESIGN CONSIDERATIONS

In this section, we discuss the technical considerations that underpin the design of SoundSense while the detailed design is presented in Section 3.

2.1 Scaling Sound Classification

People live in very different environments, containing a wide variety of everyday sounds. Sounds heard by individuals on a daily basis are largely influenced by lifestyle and environment. For example, sounds encountered by a bus driver in Auckland would likely be different to that of a stock broker in Beijing. The idea of collecting large annotated sound samples and training generic classifiers for all the possible sounds in the world is simply not feasible nor, we argue, is necessary. It is impossible to acquire such a large data set, particularly when individual users are only exposed to a small subset of such a data set in their everyday lives. We describe this challenge as the audio sensing scalability problem. Prior work in this area typically avoids this hard problem by restricting the operating domain to a narrow range of possible acoustic settings. The SoundSense system is designed to specifically attempt to make progress toward addressing this important scalability problem. In essence, SoundSense uses different strategies when dealing with different sounds. Some sounds are general in people's life, such as voice and music, which are also well studied and can be modeled with generic models [14] [21] [24] [34]. For other sounds, which are more specific to an individual, unsupervised learning techniques are used to discover and learn new sound types as the system is exposed to more sound events. Through interaction with end-users, SoundSense labels new sound signatures in a meaningful manner. As a result, SoundSense is able to *customize* the unique set of sounds a user is exposed to. SoundSense adopts a hierarchical classification architecture. In the first stage, sound is classified as one of the three coarse sound categories: voice, music, ambient sound (i.e., everything else). In the second stage, further analysis is applied according to the category of the sound. In the case of voice and music, finer discrimination (e.g., gender classification in the case of speech) can be done to meet specific application needs. The ambient sound is also learned over time by the unsupervised adaptive learning and classification process in the second stage. Importantly, SoundSense does not attempt to learn all sounds encountered, rather, it identifies those sounds considered *significant sounds* in the lives of end-users; that is, in terms of frequency of encounter and duration of its occurrences. When SoundSense determines a new sound to be significant, it prompts the end-user to either provide a textual description (i.e., a label) or rejects the sound as unimportant or sensitive in terms of privacy. In this manner, SoundSense's audio processing is customized to the particular user without any prior assumptions about sounds he/she will encounter. SoundSense's design decision to base coarse classification on supervised training and ambient soundclassification on unsupervised learning, directly addresses the need to design for scale. In Section 6, we demonstrate through implementation and experimentation that with relatively few user interactions (for labeling), SoundSense can quickly learn new sounds that are significant in people's everyday lives.

2.2 Phone Context

Mobile phones are ubiquitous and ideal for capturing the sound events we experience around us in our everyday lives. However, phones are primarily designed for voice communication and present a number of practical limitations. People carry phones in a number of different ways; for example, in the pocket, on a belt, in a purse or bag. The location of a phone with respect to the body, where a phone is used and the conditions under which it is used is collectively referred to as the phone context. The phone context presents a number of challenges to building a robust sound sensing system because sound can be muffled, for example, when the phone is in the pocket or backpack. A goal of SoundSense is to support robust sound processing and classification under different phone context conditions, which vary the volume level. To get a sense of some of the practical challenges that phone context presents to the design of SoundSense on a mobile phone we performed a controlled experiment using an Apple iPhone sampling a distinct sound within a lab environment inside the Computer Science building at Dartmouth College. Figure 1 shows the same sound source, a Macbook Pro playing constant volume white noise, being sampled from the same distance during a variety of different phone contexts with the phone being placed in different locations or with the person facing different directions. The figure shows that the root mean square (RMS) value of the sound, which is a good approximation of the average volume, deviates by more than 30% within the range of different contextual conditions. Although energy based features are shown to be effective for some types of classification, they are sensitive to volume and have typically been used in controlled settings rather than in the wild. Due to the uncertainty of phone context and its use under real world conditions and not laboratory controlled conditions, we avoid using these types of features and adopt features that are more robust to volume variations and phone context.

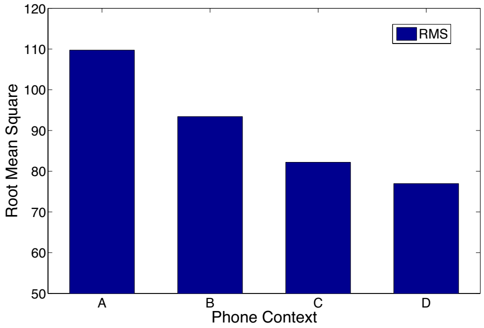


Figure 1: Phone context alters the volume (showed as RMS) of the same sound event significantly. The context represented above is: (A) in the hand of the user facing the source, (B) in the pocket of the user facing the source, (C) in the hand of the user facing away from the source, (D) in the pocket of the user facing away from the source.

2.3 Privacy Issues and Resource Limitations

Privacy is an important issue in the design of mobile sensing applications. People are sensitive about how audio data captured by the phone, particularly conversational data, is used by the system. Users need to be convinced that their privacy is safeguarded. It would be problematic if an application sends audio samples from the phone to a remote server for processing. With this in mind the SoundSense system is designed to run locally on the phone without any server interaction. SoundSense only uses raw sound samples in the case of the feature extraction step, as discussed in Section 3. After this step is complete only features are retained and the raw samples are discarded. All the audio data is processed on the phone and raw audio is never stored. When a new type of sound is discovered, users are given the choice to either provide a label for the new sound or mark the new sound rejected in which case the system does not attempt to classify such an event in the future. The user has complete control over how the results of classification are presented either in terms of being visualized on the phone screen or pushing them to external applications, such as, social networks. [27]. Note, that in our current implementation results can only be displayed locally on the user's phone.

Another pressing challenge when designing continuous sensing applications such as SoundSense is that mobile phones are resource limited and applications need to be designed with this in mind. The microphone on a phone is typically designed for capturing the human voice, not ambient sounds, and typically sample at 8 KHz. According to the Nyquist-Shannon sampling theorem [38], the microphone cannot capture information above 4 KHz, and, as a result, important information is lost, for example, high frequency components of music. Beyond sampling issues, mobile phones do not lend themselves to the implementation of computationally complex signal processing and classification algorithms. In SoundSense, sounds need to be analyzed efficiently such that real-time classification is possible while not overwhelming the CPU and memory of the phone. Therefore, the designer has to consider the accuracy and cost trade off. This is a significant challenge when designing classification algorithms that have to efficiently run on the phone, without impacting the main function of the phone, i.e., voice communications. When implementing applications on the phone, it is therefore necessary that there is always sufficient resources (e.g., CPU, memory usage needs) maintained so the phone remains responsive to calls or other phone functions. In SoundSense, we manage energy, CPU, and memory usage by performing "frame admission control" to incoming audio samples, as discussed in Section 4. Only when we estimate a sample is of adequate quality for classification does the full SoundSense classification pipeline start. To reduce the computational needs of SoundSense, we implement lightweight classification algorithms (as discussed in Section 4) capable of compensating for any errors introduced by imperfect sampling and the lightweight classification process.

Sound events in the real world (i.e., not in a controlled laboratory experiment) are complex and typically include multiple sound sources. SoundSense does not attempt to separate these different sources but rather takes the mixture of sound as the signature of the sound event. Therefore, the classification result is mainly determined by the dominant sound in the sound mix, i.e., the sound with the highest energy level. SoundSense also does not differentiate different kinds of sound source (e.g., a real conversation and speech on TV are simply recognized as human voice).

3. SOUNDSENSE ARCHITECTURE

In this section, we provide an overview of the SoundSense architecture. We describe each architectural component in turn, presenting a high-level view of how the system works in unison to provide scalable sound sensing. We present a detailed discussion of the SoundSense algorithms and system implementation in Section 4 and Section 5, respectively. Figure 2 shows the SoundSense architecture and its components. The architecture is implemented solely on the mobile phone, which in our work is the Apple iPhone. The SoundSense architecture comprises the following key components:

Preprocessing. Sound processing usually starts with segmenting the audio stream from the microphone into frames of uniform duration. Features for classification are extracted during processing either from an individual frame or from a window that is n frames long. In our system, classification is done with respect to the complete n frame window and not on individual frames. Not all frames are considered for processing. Spectral Entropy and energy measurements are used to filter frames that are silent or are too hard to classify accurately due to context (e.g., far away from the source or muffled in backpack).

Coarse Category Classification. The classification occurs in a hierarchical fashion with multiple layers of classifiers being applied. Initially, features are extracted from the frames. Following this, they are fed into the coarse category classifier - the stage one classifier. The principle role of this component is to make an assessment of the coarse-grain category of the sound: voice, music, and ambient sound (i.e., everything other than voice and music). This assignment dictates which further classification stages should be applied based on the category that is inferred. Some applications may only require coarse level classification. If this is the case, then processing ceases at this point and the inferred category is provided to the application. Alternatively, stage two classification occurs.

Finer Intra-Category Classification. The purpose of the stage two intra-category classifiers is to allow additional

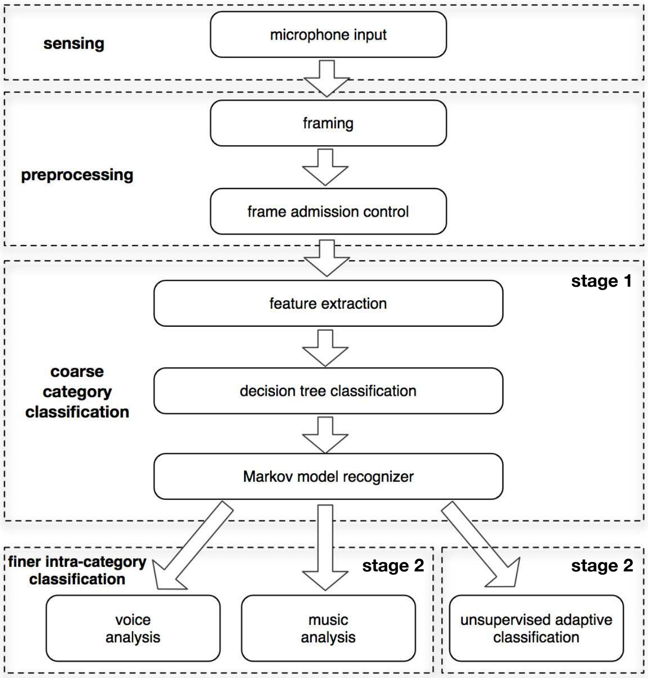


Figure 2: The SoundSense Architecture

levels of details to be inferred from the sound sample, whose category is already classified. For instance, beyond just inferring the sound contains music, further analysis can assess the genre of music [41]. Performing these types of finer grain distinctions can only happen after the type of the sound is known. In this stage, we can apply techniques from prior work, and in particular make use of work that limits the scope of usage to only some particular type of sound (e.g., voice) to perform domain specific inferences (e.g., gender classification, see Section 5 for details). For efficiency only the relevant module is activated for each sound event, as shown in Figure 2: (a) one of intra-category classifiers are applied (i.e., voice analysis or music analysis) based on the event class; or, (b) an unsupervised adaptive ambient sound classification is applied.

Unsupervised Adaptive Classification. Building supervised classifiers to recognize all types of ambient sound is not a scalable approach, as discussed earlier. The potential scope of ambient sound is vast and can change over time. Rather, we use unsupervised adaptive classification, which is itself is a special case of a two stage two classifier, to make sound processing adaptive to the set of sounds individual phone users most experience. Sounds that are recognized by the category classifier as ambient sound (i.e., not voice and music) in stage one classification are handled by this component. The design objective of this component is to learn sound clips that are significant in a person's life over time in terms of frequency of encounter and duration of the event, and to adapt the classifier itself to recognize these sounds when they recur. In Section 6, we show that many sounds, such as the turning signal of a car, vacuum cleaners, car engines are considered significant and discovered by SoundSense, while SoundSense itself has no prior knowledge about these sound events. This is the reason why we term this approach *adaptive*. The algorithm used to discover and model significant sounds is unsupervised. However, the user is prompted to provide a textual label to associate with the sound.

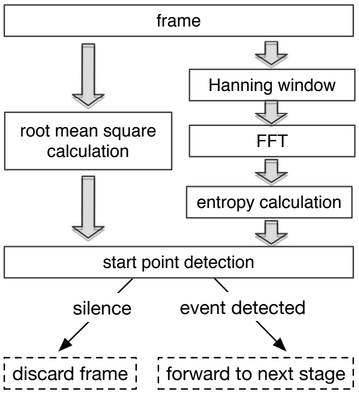


Figure 3: Work flow of the preprocessing component

4. SOUNDSENSE ALGORITHMS

In what follows, we present the detailed design of the SoundSense algorithms implemented on the phone as part of the SoundSense architecture.

4.1 Preprocessing

This preprocessing component, as shown in Figure 3, is responsible for segmenting the incoming audio stream into frames and performing frame admission control by identifying when frames are likely to contain the start of an acoustic event (e.g., breaking glass, shouting) that warrants further processing.

4.1.1 Framing

Segmenting the audio stream into uniform frames is common practice for feature extraction and classification. The frame width (i.e., duration) needs to be short enough so that the audio content is stable and meanwhile long enough to capture the characteristics signature of the sound. Existing work [26] use frames that overlap each other since overlapping frames are able to capture subtle changes in the sounds more precisely. However, this can cause overlapping pieces of audio data to be processed multiple times. Given the resource constraints of the phone we use independent non-overlapping frames of 64 ms. This frame width is slightly larger than what is typically used in other forms of audio processing (e.g., speech recognition) where the width typically ranges between 25-46 ms. In addition to enabling a lower duty cycle on the phone, a frame width of 64 ms is sufficient for capturing the acoustic characertistics of environmental sounds.

4.1.2 Frame Admission Control

Frame admission control is required since frames may contain audio content that is not interesting (e.g., white noise) or is not able to be classified (e.g., silence or insufficient amount of the signal is captured). These frames can occur at any time due to phone context; for example, the phone may be at a location that is virtually silent (e.g., library, home during night) or where the sounds that are sampled are simply too far away from the phone to be sufficiently captured for classification. Alternatively, the phone position (e.g., deep inside a backpack) may also prevent the microphone from acquiring the sound clearly enough for classification.

Frame admission is done on the basis of energy level and spectral entropy. Low energy level indicates silence or undesirable phone context, which prevents meaningful classification. Spectral entropy gives cues about the frequency pattern of the sound. A flat spectrum (silence or white noise) results in high entropy. A low entropy value indicates a strong pattern in the spectrum. The energy of a frame is evaluated by taking the RMS of the amplitude of the audio content it contains. To compute spectral entropy, we need to perform the following three steps: (i) apply a hanning window to the frame, which suppresses the frame boundaries and thus reduces the known effect of Fast Fourier Transform (FFT) spectral leakage [18]; (ii) calculate the FFT spectrum of the frame; and finally, (iii) normalize the spectrum, treat it as a probability density function, and finally obtain the spectral entropy, H_f, by,

$$H_f = -\sum_{i=1}^{n} p_i \log p_i \tag{1}$$

Acoustic events captured by the phone's microphone should have reasonable high RMS values, which means the volume of the sound sample is not too low. However, this is not sufficient. The phone's context may lower the RMS of the sound but the received sound may still contain enough information that warrants processing as long as the sound signature is sufficiently captured. So we use spectral entropy to differentiate informative low volume samples from non-informative ones. Frames that contain mainly silence have a uniform distribution of energy in the frequency domain. In contrast, a frame that contains acoustic events has multiple energy peaks in the spectrum that correspond to the pitch and harmonics of the sound. Therefore, its spectral entropy tends to be lower than silent frames.

Two thresholds, $rms_{\text{threshold}}$ and $entropy_{\text{threshold}}$, are used for frame admission control. For frames to be admitted they must have either an RMS above $rms_{\text{threshold}}$, or if below they must have a spectral entropy below $entropy_{\text{threshold}}$. Frames that meet this criteria are kept and processed, or otherwise discarded. One complication of this approach is that certain acoustic events such as conversations and music are not continuous. Speech contains frames of unvoiced sounds or pause periods. Similarly, music may stop for periods and then resume. Frame admission control takes this condition into account and does not reject "in-between" frames (i.e., low RMS or high entropy frames within a non-continuous sound event). If these in-between frames were considered independently they would fail the admission criteria. To cope with this admission issue we admit frames for a short period of time (i.e., five seconds) when engaged in an ongoing acoustic event regardless of the content of these frames. During this period, each frame associated with an event will ensure that the next five seconds of data is admitted. If no new event frames are received within five second, frame admission ceases.

4.2 Coarse Category Classification

Coarse category classification classifies the current acoustic event to one of the coarse sound types (i.e., voice, music, ambient sound). In what follows, we first discuss the features we extract from the frame window provided by preprocessing and then describe the how classification is performed.

4.2.1 *Feature Extraction*

The selection of features is critical in building a robust classifier. Being conscious of the limitations of our platform to deal with issues such as the clipped response of the microphone (typically clipping at the 8 KHz phone quality level) and the variability in the energy of the sound sampled by the microphone, we consider a large range of features, both in the temporal and spectral domains. All of the features we use are insensitive to volume. Through experimentation we discovered that the drawback of the clipped frequency response is compensated by using multiple spectral features. In what follows, we discuss temporal features we use in SoundSense:

Zero Crossing Rate (ZCR). We use both ZCR variance and a count of the number of ZCR peaks. ZCR [34] is defined as the number of time-domain zero-crossings within a frame,

$$ZCR_f = \frac{\sum_{i=0}^{n} |sign(s_i) - sign(s_{i-1})|}{2} \tag{2}$$

where the $sign()$ function is 1 for positive arguments and -1 for negative arguments. ZCR correlates with the frequency content of a signal. Human voice consists of voiced and unvoiced sounds. The voiced and unvoiced frames have low and high ZCR values, respectively. Therefore, human voice shows a higher variation of ZCR. Typically, music does not have this variation in ZCR, although some music does (e.g., rock music that contains a lot of drumming). Ambient sounds are usually fairly stable in frequency, the variance of ZCR remains low.

Low Energy Frame Rate This is the number of frames within a frame window that have an RMS value less than 50% of the mean RMS for the entire window [35]. For voice there are more quiet frames corresponding to unvoiced sound, so this measure will be higher for speech than for music and constant noise [34]. For all spectral features the DC component of the spectrum (i.e. the first FFT coefficient), which indicates the average volume of the sound, is discarded, and the values of all the frequency bins are normalized in order to remove the influence of the volume. We now describe the spectral features we use. Note, that our definitions frequently refer to $p_t(i)$ which is indicating the normalized magnitude of the ith frequency bin of the computed FFT spectrum:

Spectral Flux (SF). SF [35] is defined as the L2-norm of the spectral amplitude difference vector of two adjacent frames.

$$SF_t = \sum_{i=1}^{n} (p_t(i) - p_{t-1}(i))^2 \tag{3}$$

where $p_t(i)$ and $p_{t-1}(i)$ are referring to the current frame and the last frame, respectively. SF measures the change in the shape of the spectrum. Speech generally switches quickly between voice and unvoiced sound, altering its shape rapidly. Music does not typically have this characteristics and usually has a lower SF value, but it occasionally goes through dramatic frame to frame changes, which result in SF values that are much higher than the value of voice.

Spectral Rolloff (SRF). SRF [21] is defined as the frequency bin below which 93% of the distribution is concentrated.It can be calculated by,

$$SRF_f = max\big(h|\sum_{i=1}^{h} p(i) < \text{threshold}\big) \quad (4)$$

It is a measure of the skewness of the spectral distribution, the value is larger for right-skewed distributions. Music signals, which contain a greater number of higher frequency components tend to have high SRF values.

Spectral Centroid (SC). SC [21] is the balancing point of the spectral power distribution.

$$SC_f = \frac{\sum_{i=1}^{n} i \cdot p(i)^2}{\sum_{i=1}^{n} p(i)^2} \quad (5)$$

Music usually involves high frequency sounds which push the spectral mean higher.

Bandwidth. Bandwidth [21] is the width of the range of the frequencies that the signal occupies. It makes use of the SC value and shows the spectrum is concentrated around the centroid or spread out over the whole spectrum.

$$BW_f = \frac{\sum_{i=1}^{n} (i - SC_f)^2 \cdot p(i)^2}{\sum_{i=1}^{n} p(i)^2} \quad (6)$$

is a measure of the "flatness" of the FFT spectrum. Most ambient sound consists of a limited range of frequencies, having a small value. Music often consists of a broader mixture of frequencies than voice and ambient sound.

Normalized Weighted Phase Deviation. This feature introduced in [15] shows the phase deviations of the frequency bins in the spectrum weighted by their magnitude,

$$nwpd_f = \sum_{i=1}^{n} p(i) \cdot \phi_i'' \quad (7)$$

where ϕ_i'' is the second derivative of the phase of ith frequency bin. Usually the ambient sound and music will have a smaller phase devation than voice.

Relative Spectral Entropy (RSE). This feature introduced in [7] differentiates speech and other sounds. However, in [7] RSE is defined as the KL (Kullback-Leibler) divergence between the frequency domain of the current window of frames and previous 500 frames. Given the resource constraints of mobile phones we represent the historical patterns in the frequency domain for each frame as:

$$m_t = m_{t-1} \cdot 0.9 + p_t \cdot 0.1 \quad (8)$$

where m_t is the mean spectrum vector of time t, p is the normalized spectrum vector, and the relative spectral entropy is given by:

$$RSE_f = -\sum_{i=1}^{n} p(i) \log \frac{p(i)}{m_{t-1}(i)} \quad (9)$$

For all spectral features described above, the variance is also calculated over an n frame window and used as longer term features.

4.2.2 Multi Level Classification

The stage one coarse classification itself occurs as a two step process in which we couple the output of a decision tree classifier to a collection of markov models, one for each sound event category. The design combination meets our requirement for lightweight processing on the mobile phone, while being robust to the complications of the real world sound sensing.

The use of a decision tree classifier evolved out of a study investigating the use of several other classification algorithms that have proven to be effective in the audio processing including: Gaussian Mixture Models (GMMs) [40] [26], k-nearest neighbors(KNN) [30], decision trees [44] and Hidden Markov Models (HMMs) [44]. We construct classifiers using all of these algorithms with an identical set of features (see Section 4.2.1) and the same training set (see Table 1 for details). As observed by others [40] [44] we did not find substantial performance differences between them. This is not unexpected given the function of the category classifier is to segment the feature space into large grain categories of sound categories. Given the feature space is segmented into wide segments, much of the differences between the algorithms is nullified. Finally, we use a decision tree generated using the J.48 algorithm [43] as it is simple to construct and execute while being equivalent in performance to the alternatives.

When we consider the decision tree classifier in isolation it provides reasonable accuracy (i.e., approximately 80%, as discussed in Section 6). However, the decision tree classifier makes a category assignment for each window locally and in isolation of other assignments, as a result sometimes the output is choppy (see Table 2). Some real-world scenarios would cause the classifier to enter a less accurate oscillating state when the sound alternates from one category to another. When the sound remains in one category the classifier's performance remains high but during the switching periods the performance of the system degrades, for instance, a conversation or presentation with a sequence of short pauses over time, a piece of music switches quickly between smooth and intensive parts. We cope with these issues by adding smoothing to the output of the decision tree. We build simple first-order Markov models for each of the three categories. They allow us to make soft category assignments from the output sequence of the tree classifier. The Markov models are trained from the output sequence of the decision tree, which are the category assignments of the sound sample. Examples sequences are given in Table 2. Each model uses the same structure, one state for each of the three sound categories. The models are trained to learn the pairwise transition probabilities $p^i(s_t|s_{t-1}), s \in \{0, 1, 2\}$ for each category i. For a given class sequence $\{s_1, ..., s_L\}$ of length L passed by the decision tree classifier the final class assignment i is determined by finding the model of category i that maximized the probability,

$$C = arg \max_i \sum_{t=1}^{L-1} \log p^i(s_{t+1}|s_t) \quad (10)$$

We train the classifiers in our implementation using a training data set we built up over three months. We detail the composition of the data set in Table 1. Each sound category contains a variety of different examples that are gathered under a wide range of scenarios. The audio samples are WAV files recorded by iPhone and Nokia N95 in the same 8Khz 16 bit Mono PCM format, which is also the format we use during the execution of the SoundSense system. All 559 sound clips (about 1 GB in total) are manually labelled, see Table 1. Because the voice data is collected from real world scenarios and only roughly labelled, most of them contain a

Category	Activies	Note
Speech	reading, meeting, chatting, conference talks, lectures,	8 female and 32 male
Music	classical, pop, new age ,flute, rock, folk saxophone, guitar piano,trumpet, violin	played by CD player, ipod, radio, and laptop
ambient sound	driving in city, elevator highway driving, walking airplane, crowd of people vacuuming, fan, shower clapping, toilet flushing,rain climbing stairs, wind, faucet	samples collected from multiple scenarios

Table 1: Description of the training data set

Sound Type	Output from Layer Two
ambient sound	00001000001000000000010002000
Music	11111021111112111111111110011
Speech	22222101212200221222212212222

Table 2: These numbers (i.e. category assignments) are the output of the first layer decision tree classifier. They can be interpreted as follows: 0 is ambient sound, 1 is music, and 2 is voice. The output are provided directly to the Markov model based second layer as part of the category classification process.

considerable amount of pauses/silence. We clean this training set by removing all the silent pieces lasting longer than 1 second. Music and ambient sound samples were not altered.

4.3 Finer Intra-Category Classification

The purpose of finer intra-category (i.e., category-specific) classification is to allow further analysis of sound events. Much of the previous work on audio signal processing is performed using audio input containing data only from one audio category. Once the category of the sound event is identified by category classification, detailed analysis can be done to provide type specific information of the input signal, according to the requirements of the application. For voice and music input, techniques such as speech recognition [32], speaker identification [33], or music genre classification [41] can be applied. The features we adopt are widely used by other audio and music classification [26]. More powerful yet computationally demanding features, for example, Mel Frequency Cepstral Coefficient (MFCC) can be calculated from the FFT spectrum which is already available at this stage. Although a lot of research has been done in the speech and music domain, little work has focused on the classification of the everyday ambient sound, which is different from user to user. In what follows, we discuss our unsupervised learning algorithm for processing the ambient sound.

4.4 Unsupervised Ambient Sound Learning

The unsupervised adaptive classification component copes with all the ambient sounds. The objective of this component is to discover over time environmental sounds that are significant in every life in addition to voice and music, so that the SoundSense system is able to classify these sounds correctly when they recur.

4.4.1 Overview

Classifiers in previous sections are trained offline with labeled data. They can only differentiate types that they are trained with and have no ability to identify new types of sound. In this component, we deal with sound classified as ambient sound by the previous stage in a different manner. The algorithm described below is unsupervised, and sound is processed on-the-go.

In this stage, we use MFCC as feature, which is designed to mimic human perception [45]. MFCC provides fine details in the frequent bands to which the human ear is sensitive, while they also capture coarser information over other spectral ranges. MFCC is recognized as one of the most important feature sets for audio signal processing [31] [26]. In our experiments, MFCC performs well, however, our learning algorithm is agnostic to features in use. The modification of the feature set does not require changes to the rest of the algorithm.

The FFT spectrum of each incoming frame is further converted into MFCC features vectors.We shows how we determine frame lengths for our implementation, in Section 5. Feature vectors are classified into *bins*, with each bin representing one type of sound, as discussed in Section 4.4.2. This operation is computationally expensive, so it runs on a window basis,and the MFCC vectors for each frame are averaged over the window. There are a large number of possible sounds in a person's life. Due to the limited resources of the platform, it is not feasible to keep infinite number of bins on the phone. SoundSense ranks all bins in terms of encounter frequency and summed duration and less significant bins are expunged when necessary, as discussed in Section 4.4.3. The assumption here is that frequency and duration of a sound indicates its importance. The system will prompt the user for meaningful labels of significant sounds discovered by the algorithm. The whole pipeline of the unsupervised algorithm described in this section is summarized in Fig. 4.

4.4.2 Unsupervised Learning of Sound Events

We use a simple Bayes classifier [8] with equal priors for each class to represent different ambient sound events (e.g., using a washing machine, driving a car). We assume each MFCC feature vector of window i, denoted as $\mathbf{w}_i$, is from one of B multivariate Gaussian distributions, $\mathcal{N}(\mathbf{w}_t; \mu_b, \Sigma_b), b \in \{1, ..., B\}$ (referred in this paper as *bins*). B is the total number of *bins*. For each MFCC feature vector $\mathbf{w}_i$, its label is denoted as random variable $p_i \in \{1, ..., B\}$, which has the following distribution:

$$\text{Prob}(p_i = j) = \begin{cases} 1 & \text{if } j = \arg\max_b \mathcal{N}(\mathbf{w}_i; \mu_b, \Sigma_b) \\ & \text{and } \mathcal{N}(\mathbf{w}_i; \mu_j, \Sigma_j) > \epsilon_{\text{threshold}} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Where $\mathcal{N}(\mathbf{m}; \mu, \Sigma)$ is the probability density function for the multivariate Gaussian distribution and $\epsilon_{\text{threshold}}$ is the confidence level. We make hard decisions in Eq. 11 to limit the computational cost of the algorithm: for each $\mathbf{w}_i$, the probability it comes from bin b is either 1 or 0. The parameter $\epsilon_{\text{threshold}}$ ensures that a bin can not grow too wide in size and cover irrelevant sound events. The trade-off here is that a larger $\epsilon_{\text{threshold}}$ will split similar sounds of the same sound event to different classes, while a smaller $\epsilon_{\text{threshold}}$ will merge different sounds to the same cluster. We set this parameter to 1×10^{-13} for the 12-dimensional MFCC feature vectors based on experimental results. This value gives best performance over our training set.

The model evolves over time according to the sounds that the user experiences. It is updated as new evidence is observed. During this online process, for every incoming feature vector w_t at time t, our model parameters are updated by maximizing the likelihood over all the feature vectors $w_1, ..., w_t$ sensed since the system started. In other words, the mean and covariance for bin b is updated by the mean and covariance over all previous feature vectors that come from the bin as described in Eq.12 and Eq.13. It should be noted that both Eq.12 and Eq.13 can be updated incrementally without storing previous $\mathbf{w}_i$ on the phone.

$$\mu'_b = \frac{\sum_{i=1}^{t} \text{Prob}(p_i = b)\mathbf{w}_i}{\sum_{i=1}^{t} \text{Prob}(p_i = b)} \tag{12}$$

$$\Sigma'_b = \frac{\sum_{i=1}^{t} \text{Prob}(p_i = b)(\mathbf{w}_i - \mu'_b)(\mathbf{w}_i - \mu'_b)^T}{\sum_{i=1}^{t} \text{Prob}(p_i = b)} \tag{13}$$

In practice, we assume that all covariance matrices are diagonal matrices and we simplify Eq. 13 in our implementation to reduce the computational cost.

We use a HMM model to smooth the output of the multivariate Gaussian classifier. The use of HMM model is similar to the reason we use HMM in the previous stage, as discussed in Section 4.2.2. We define the sequence of $\{s_1, ..., s_m\}, s_i \in \{1, ..., B\}$ as the hidden states for the sequence $\{p_1, ..., p_m\}$. In the next section, we will use s_i as the smoothed label for audio feature $\mathbf{w}_i$ instead of p_i to rank all sounds. We define the observation emitting probabilities and transition probabilities as:

$$\text{Prob}(p_i|s_i) = \begin{cases} \gamma & (s_i = p_i) \\ \frac{1-\gamma}{B-1} & (s_i \neq p_i) \end{cases} \tag{14}$$

$$\text{Prob}(s_i|s_{i-1}) = \begin{cases} \pi & (s_i = s_{i-1}) \\ \frac{1-\pi}{B-1} & (s_i \neq s_{i-1}) \end{cases} \tag{15}$$

We use $\gamma = 0.1$ and $\pi = 0.999$, which maximize the performance over the training fraction of the dataset. We only apply Viterbi smoothing to every C windows for computing efficiency.

4.4.3 SoundRank and Bin Replacement Strategy

There are a large number of sounds that a user may encounter, and we cannot give each sound a Gaussian bin due to limited resources on the phone. Thus, we define SoundRank as a way to rank sounds and only maintain bins of the most interesting sound events observed.

We define sound events to be interesting if they recur frequently within a time window ($[t^L_{int}, t^H_{int}]$) and have a minimum duration of t_{dur}. A simple function that captures these requirements for our sound rank value for bin b is as follows:

$$SR(b) \quad = \quad (\sum \rho_i + \sum \gamma_i) \tag{16}$$

where ρ_i is each occurrence duration that is larger than t_d and γ_i is the reward score for each interval between two consecutive occurrences that is within $\left[t^L_{int}, t^H_{int}\right]$, which is used to control the granularity of the event classification. The intuition here is if two instances of same sound come close in time (less than t^L_{int}), they might come from the same occurrence of this sound event, rather than two consecutive occurrences. On the other hand, if a sound does not happen often (within t^H_{int} interval), it might not be important. For example, if we would like to only detect sounds that occurs on roughly a 24 hour interval, we can set $\left[t^L_{int}, t^H_{int}\right]$ to $[20hours, 28hours]$, allowing some variance. In our experiments, we set $t^L_{int} = 40min$ and t^H_{int} to ∞ and γ to 100. This is only one of many possible ways to define the SR function. Different applications with different requirements can use different SR functions. Bins are ordered based on $SR(b), \forall b \in \{1, ..., B\}$. After all bin slots are occupied, if a new unknown sound are captured, i.e., $\forall b, \text{Prob}(P_i = b) = 0$, we then discard the bin b which has the least SR, and use $\mathbf{w}_i$ as the new μ'_b and a default covariance matrix as Σ'_b.

```
Intialize $\Sigma_b$ and $\mu_b$ for all $b \in \{1, ..., B\}$.
$t = 0$
$c = 0$
while retrieve a new window MFCC vector $\mathbf{w}_t$ from sensing component at time $t$ do
  Compute the distribution of $p_t$
  if $\mathbf{w}_b$ belongs to one existing bin $b$ then
    Update parameters for bin $b$ according to Eq. 12 and Eq. 13.
  else
    Find the bin $b$ with the least $SR(b)$
    Use $\mathbf{w}_t$ as the mean of bin $b$, set the new $SR(b)$ to zero
  end if
  if $c = C$ then
    Apply HMM Viterbi algorithm to trace $\{p_{t-C}, ..., p_t\}$ and get $\{s_{t-C}, ..., s_t\}$
    $c = 0$
    Update $SR(b), \forall b$ according to $\{s_1, ..., s_t\}$
  else
    $c = c + 1$
  end if
  $t = t + 1$
end while
```

Figure 4: Pseudo-code for the algorithm to bootstrap and update the model.

4.4.4 User Annotation of New Sounds

When the sound rank associated with a certain bin exceeds a given threshold, the user is prompted to provide a label for that sound event. The user can choose to provide a label or ask the system to ignore that sound in the future if they consider the sound to be either unimportant or an invasion of privacy. In the later case, the system will still keep the bin but the recognition will be suppressed; otherwise, deletion of the bin would force the sound to be learnt again.

5. IMPLEMENTATION

The SoundSense prototype system is implemented as a self contained piece of software that runs on the Apple iPhone. In this section, we describe our software prototype and system parameters.

We validate the SoundSense architecture and algorithms through a prototype implementation on the Apple iPhone. Our current version is approximately 5,500 lines of code and

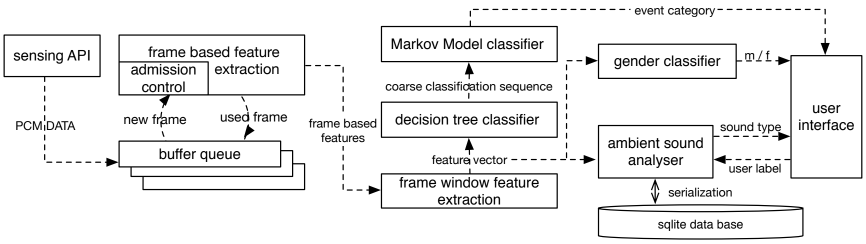


Figure 5: Implementation component diagram.

is a mixture of C, C++ and objective C [3]. Our signal processing and core classification algorithms are mainly written in C and C++. Objective C is necessary to build an Apple iPhone application which allows us to access hardware (e.g., microphone) and construct a GUI. The complete binary package is approximately 300 KB including resource files. The SoundSense prototype implements a high-level category classifier that differentiates music, speech, and ambient sound; an intra-category classifier is applied to the voice type and distinguishes male and female voices; and finally, the unsupervised adaptive classifier uses 100 bins for the purpose of modeling ambient sounds. Under the current Apple iPhone SDK [5] we are not allowed to run the prototype as a daemon process that is necessary for continuous sampling. This is not a technical limitation but commercially driven decision by Apple. For experiments that require continuous sampling we run our SoundSense prototype on jail-broken iPhones that allow background process without modifying our software.

The individual software components in our implementation and the interactions between them are shown in Figure 5. Using the standard iPhone SDK API, we collect continuous 8 kHZ, 16-bit, mono audio samples from the microphone. The PCM formatted data is placed in a three-buffer circular queue, with each buffer in the queue holding an entire frame (512 samples). Once the buffer is full, it is provided to the preprocessor which makes a frame admission control assessment. Our SoundSense prototype is optimized for CPU usage and we exchange additional memory usage for lower CPU load. Memory blocks are pre-allocated and initialized as much as possible when the application is loaded. If there is a lack of an acoustic event, the system enters into a long duty cycle state in which only one frame in every ten frames is processed (i.e., every 0.64 seconds). If a frame is accepted by the frame admission control, which means an event has been detected, then processing becomes continuous. The KissFFT library [9] is used to calculate the FFT. Once all frame based features are calculated, the buffer is released back to the circular queue for future use. Frame based features are calculated once n frames that are necessary for a frame window are accumulated. The long term features, such as, variance, are calculated over this frame window. As discussed earlier, we do not use overlapping windows. Rather, windows are classified as discrete chunks. We use log-probability for all of our classifiers to avoid arithmetic underflow. Once a full window frame is collected, window based features are calculated and pushed to the high-level category classifier. The first step in the coarse-level category classification is to apply the decision tree to the incoming feature vector. Based on our training sets (see Table. 1) we learned a 17-node tree with a depth of 6 levels. The output of the tree is one of three category assignments (music, voice, ambient sound) which is inserted into a FIFO queue. The queue is provided to the Markov model classifier with three pre-trained models to smooth the output. Depending on the output of the high-level category classifier either an intra-category classifier or the unsupervised adaptive classifier is used. Where appropriate previously calculated features are provided to the subsequent classifiers (e.g., the FFT spectrum). In our current implementation, only a single intra-category classifier is available. This classifier (available for the voice category) tries to distinguish male and female voices. To construct the classifier we simply re-used existing components. For the features we use MFCC feature extraction routines and for the classifier we use the multivariate Gaussian implementation both of which are required for the unsupervised adaptive classifier, which analyizes ambient sound. This component uses the SQLite database to keep the necessary statistics of the required bin structure and label/bin mappings acquired from users.

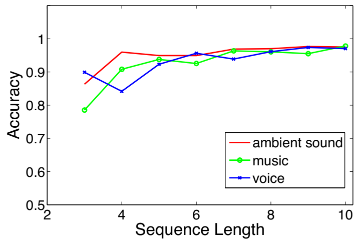


Figure 6: Classification accuracy vs. sequence size

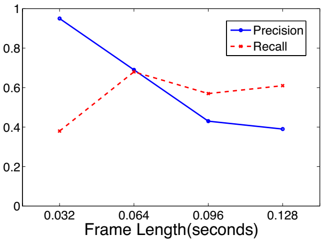


Figure 7: Recall and precision vs. frame length.

The choice of system parameters also has an impact on the performance of SoundSense. As discussed in Section 4, the output of the decision tree classifier is buffered in a FIFO queue before input to the Markov model recognizer. A larger buffer causes a longer sequence length to be provided to the Markov model which increases the accuracy but at the expense of the system's responsiveness to state changes. To determine a suitable buffer size, we perform experiments on

status	cpu usage	memory usage
GUI only	less than 1%	2.79MB
Silence	1%-5%	3.75MB
Music/Speech	8%-20%	3.80MB
Ambient Sound	11%-22%	3.85MB∼5MB

Table 3: CPU and memory benchmarks of prototype software running on the iPhone

the training data discussed in Section 4.2.2. The results of these experiments are shown in Figure 6. The figure shows that the gain of larger buffer size is marginal beyond a buffer size of five. This observation is independent of the category of sound. Therefore, we adopt a buffer size of five for the SoundSense implementation, where the tradeoff of latency and accuracy are suitably balanced.

The length of each MFCC frame used in the unsupervised adaptive ambient sound learning algorithm impacts the computational costs and classification accuracy. We examine this tradeoff with a number of experiments on the Intel MSP (Mobile Sensing Platform) data set [12] collected as part of earlier work. The MSP data set contains fifty-two 25-minute-long sound clips collected by the MSP device, which senses audio signals with a microphone similar to those on the mobile phones. For each clip, one of the participants carries the devices performed several daily activities such as walking, riding elevators, and driving. Each activity in the clips is manually labeled. We first apply our unsupervised learning algorithm to form bins over all the clips and for each activity find the most relevant bin by selecting the sound bin which is encountered most frequently during the duration of that activity. We use these sound bins to segment audio sequences into different classes and compare the results with the ground truth labels by calculating the precision [1] and recall [2]. In Figure 7, we see the changes on recall and precision as the MFCC frame length varies. These results suggest that a larger frame length tends to increase recall but decrease precision. We observe that an MFCC frame length of 0.064s is a good trade-off point considering both precision and recall.

6. EVALUATION

In this section, we discuss the evaluation of the SoundSense system. We first discuss a number of benchmarks for the Apple iPhone based prototype system. Following this, we present the detailed performance evaluation of the SoundSense multi-stage classification process and discuss two proof-of-concept people-centric sensing applications built on top of SoundSense on the Apple iPhone.

6.1 CPU and Memory Benchmarks

Table 3 presents results of our CPU and memory benchmarks. If the environment is quiet, the system adopts a long duty cycle and the CPU usage is less than 5%. Once acoustic events are detected and processing begins, the CPU usage increases to about 25%. We measure the elapsed time for processing a frame (64 ms) to be around 20 to 30 ms, depending on the particular path through the processing workflow (e.g., ambient sound requires more processing than music, which currently has no additional intra-category classifiers). The SoundSense process together with all other system processes on the iPhone consumes less than 60% of the CPU in total. Memory consumption is potentially more dynamic and depends on how many bins are in use by the unsupervised adaptive (ambient sound) classifier. Even with all bins being used (currently capped at 100), the total memory usage is about 5 MB, comparing to the 30 MB (128MB in total) memory available for 3rd party applications on the iPhone. These results indicate that our software preserves enough resources for other 3rd party applications or further SoundSense extensions, such as more intra-category classifiers.

Category	Num of Clips
Ambient Sound	47
Music	18
Speech	49(19 females,30 males)

Table 4: Testing data set

Actual\Classified as	Ambient Sound	Music	Speech
Ambient Sound	0.9159	0.0634	0.0207
Music	0.1359	0.8116	0.0525
Speech	0.0671	0.1444	0.7885

Table 5: Confusion matrix for the decision tree classifier

6.2 Classification Performance

In this section, we discuss the quality of the inferences made by each classification component.

Coarse Category Classifier. We explore (i) the effectiveness of the decision tree subcomponent; and (ii) the advantages of the secondary Markov model layer for smoothing of the coarse category classifier. For this experiment, we use a test data set that is collected from real world setting over a two month period and is distinct from the training set discussed in Section 4.2.2. The test set contains sound clips from different categories of sounds (music, speech, and ambient sound) that the classifier is designed to recognize. Table 4 provides a breakdown of the number of clips for each category. The length of these clips range from as little as 5 seconds (e.g., clapping) to as much as 10 minutes (e.g., speech). Each clip is annotated manually with a label. The types of sound in each category are identical to the types presented in Table 1.

Table 5 shows the confusion matrix representation of the result for the classification process using only the decision tree component. The ambient sound is identified correctly 90% of the time. Speech and music are more difficult to recognize, and are both classified correctly around 80% of the time. This is due to the dynamic nature of these two categories and the limitation of the sampling rate. Some of the slow and smooth sections of music are mistaken for ambient sound, and some allegro tunes are confused with speech. Similarly, around 14% of the speech samples are misclassified as music. The pause-continue-pause pattern is often found in everyday conversation and the frames near the boundary of voice and pause sections are prone to be mistaken for music. The decision tree is only the first step

[1] Precision for an event is the number of frames that are correctly classified as this event divided by all frames classified as this event.

[2] Recall for an event is the defined as the recognized occurrences of the event divided by the number of overall occurrences of this event in the audio sequence.

Actual\Classified as	Ambient Sound	Music	Speech
Ambient Sound	0.9494	0.0402	0.0104
Music	0.0379	0.9178	0.0444
Speech	0.0310	0.0657	0.9033

Table 6: Confusion matrix for the decision tree classifier with Markov model smoothing

Actual\Classified as	Female	Male
Female	0.7428	0.2572
Male	0.2807	0.7193

Table 7: Confusion matrix for the gender classifier

of the coarse category classification process. We enhance it with the Markov model smoothing process. We repeat the same experiment discussed above but now incorporate the Markov model component. We find significant improvements in our results, as shown in the confusion matrix in Table 6. In particular, the classification accuracy is improved by approximately 10% for music and speech. There is a minor 3% performance gain for ambient sound.

Finer Intra-Category Classifier. We currently implement only a single intra-category classifier – the gender classifier. Future work will extend this initial implementation. The classifier is fairly simple in comparison to other examples found in the literature (e.g., [37] reports 90% accuracy) so we do not expect high accuracy for this classifier. We perform an experiment using the voice clips in the test data set. According to this data set, we achieve 72% classification accuracy, as shows in Table 7.

Unsupervised Adaptive Ambient Sound Learning. We here show the performance of the unsupervised algorithm in real world scenarios. We evaluate the system using a data set collected by users over a number of work days of a week, during this period the users wore a SoundSense iPhone around their neck. Each day one hour of audio is recorded in the morning. The total size of the data set is about 300 minutes in length. The interesting and significant sounds discovered are shown in Table 8. Multiple bins are associated with the car driving sound. The algorithm segmented the event of car driving into different sub-events due to the variations of car engine sounds (e.g., high pitch sound when driving fast on a highway and low pitch when driving slowly on local roads). Arguably, users may find this type of segmentation desirable. As part of our future work, we intend to fuse other types of contextual information (e.g., GPS) to address this issue.

Table 8 shows that the sound with the highest sound rank (SR) is the ambient sound when the phone rubs against the person's clothing. During the experiment, the phone is attached to a necklace and hangs around the neck. In this case, SoundSense relies on the user to reject this uninteresting sound when the system prompts the user to ask for a contextual label of the sound. It is worthwhile to point out that how users respond to these system prompts largely impact the effectiveness of SoundSense. All the labels are given by the userand the user decides whether to accept or reject sound events discovered by the system. The semantic labels are also provided by the user and therefore meaningful to the user. Users may, for example, attach the same label to different bins, (e.g., the different driving events can all be labeled "driving"). Therefore, it is the user's obligation to make the labels understandable to themsleves and possibly others people. Sourcing labels from user populations is a complex research topic in its own right [19]. While this is an interesting systems issue it is out of scope of this paper, but certainly a topic we intend to study in future.

SR Rank	Event
1st	Phone scraping against Clothes
2nd	Highway Driving
3rd	Walking Outdoor(with wind sound)
4th	Driving/Accelerating
5th	In-Town Driving
6th	Walking Indoor
7th	Taking Out Trash(Plastic rustling Sound)

Table 8: SR rank calculated by the Unsupervised Adaptive Classifier

Event	Precision	Recall
Walking	93%	53%
Driving Cars	100%	100%
Riding Elevators	78%	80%
Riding a Bus	25%	90%

Table 9: Unsupervised Adaptive Classifier performance

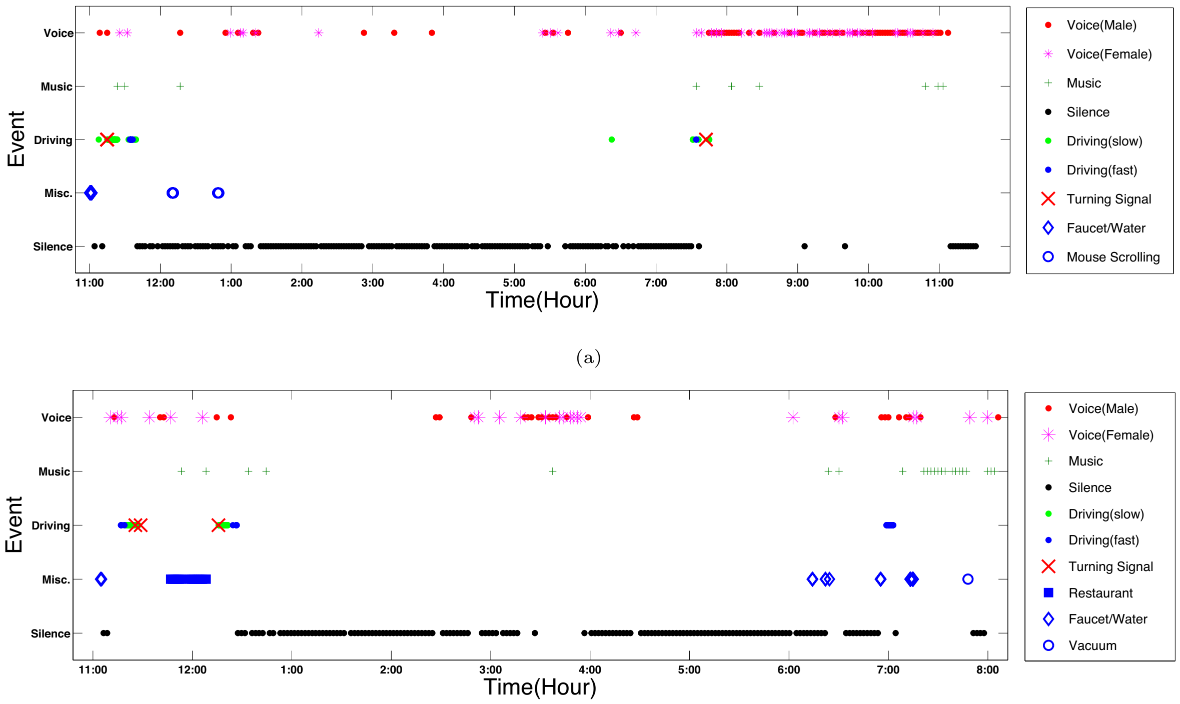
Using the MSP data set [12], we evaluate the effectiveness of the unsupervised adaptive learning algorithm in terms of precision and recall of the recognized events. In the MSP data set, there are 11 labeled human activities includes walking, walking down stairs, walking up stairs, taking elevator down, taking elevator up, jogging, sitting, etc. We merged classes such as taking elevator up and taking elevator down to one class because these activities can not be distinguished by audio characteristics. We skip activities such as sitting and standing because they have no obvious acoustic signature. We end up with four different human activities and use them to test the performance of our algorithm. Table 9 shows that the precision and recall of the four activities. We noticed that the precision of riding a bus is quite low. The reason is that there is no reliable stable sound that represents the bus riding sound well (i.e., the bus engine volume is low) in the data set. Thus, other less relevant sounds (i.e., the sound of the crowd of people on a bus) are used to identify the riding bus event.

6.3 SoundSense Applications

Finally, we implemeted two proof-of-concept applications on the iPhone using the SoundSense software; these are: (i) an *audio daily diary* appliction of everyday events based on opportunistic sensing; and (ii) a *music detector* application based on participatory sensing. We do not claim that these applications are new but use them to evaluate the ability of SoundSense to support such application on the iPhone.

Audio Daily Diary based on Opportunistic Sensing. In this application, sound is continuously sampled which produces a time series log of classified acoustic events that occur during the day to day life users. Users of the application can then perform queries against this log of sound events determining for instance how much time they spent in their cars, etc. There has been a number of applications similar to this proposed in the literature [42] [20].

To experiment with this application we had a single participant carry an iPhone to collect raw audio data over the



(a)

(b)

Figure 8: The participant's activities recognized by SoundSense on a Friday (a) and a Saturday (b).

course of two weeks. Figure 8(a) and Figure 8(b) visualize the sound events from a work day (Friday) and a weekend day (Saturday). The X axis is the time of day and the Y axis indicates different activities. A marker is drawn when the system successfully recognizes a certain category of sound. For example, a red dot is plotted when the system discovers speech events. The figures show that the subject went to a restaurant on one Saturday, and because of a loud refrigerator/vender machine in the restaurant, SoundSense is able to identify this particular location. Notably, the system discovers the "click-click"sound when the driver uses the turning signal when driving. As can see from Figure 8(b), the person uses turn signals of his car in the morning, but does not use turn signals on his way back home late at night. It was interesting that the person was unaware of this behavior until the SoundSense system highlighted it!

In general, we found that the system is good at discovering long duration events. However, there are significant sounds in our lives that are of shorter duration (e.g., door bells). We found that supporting these short duration events by simply reduce the duration threshold will make the system generate too much false significant events and thus put a heavy burden on the user to filter them. In our experiment, we also noticed that a few voice or music samples that are incorrectly classified as ambient sound are processed by the unsupervised learning algorithm and sometimes trigger a prompt. Right now, we rely on the users to filter them out by marking them as unwanted and plan to tackle this issue in our future work.

Music Detector based on Participatory Sensing. We also consider a different type of application. The ability to recognize a broad array of sound categories opens up interesting application spaces for example within the domain of participatory sensing [10]. We explore one such application built on SoundSense on the iPhone. In this particular case, we use the sound category of music and a deployment within Hanover, a small New England town where Dartmouth College is located. The goal of the application is to provide students with a way to discover events that are associated with music being played (e.g., parties, concerts, music in a room). In this simple application, people who opt-in to be part of the system are prompted via their iPhone to take a picture when music is detected by the SoundSense system. These images are displayed on a community web portal along with the location where the music is detected. Other students can visit the web portal and view the stream of submissions from SoundSense users, the location lets students know where the event is happening and the images allow them to determine if they want to join the event or not. We show some of images that were taken in Figure 9. The figure shows a screenshot of the locations where the music are detected along with icons on the map that indicate the location of the sound.

7. RELATED WORK

There is growing interest in mobile phones research for pervasive computing [29] [22] and urban sensing [11] [10] [1] applications. Although early work [36] mentions the use of microphone as a clue to context, most work found in the literature focus on the camera, accelerometer [6], and GPS [23] as sources of sensor data.

There has been significant work on audio analysis and signal processing. The basic problem of sound classification has been as an active area of research [35] [34] [21] [16] including some of the challenges we overcome with Sound-

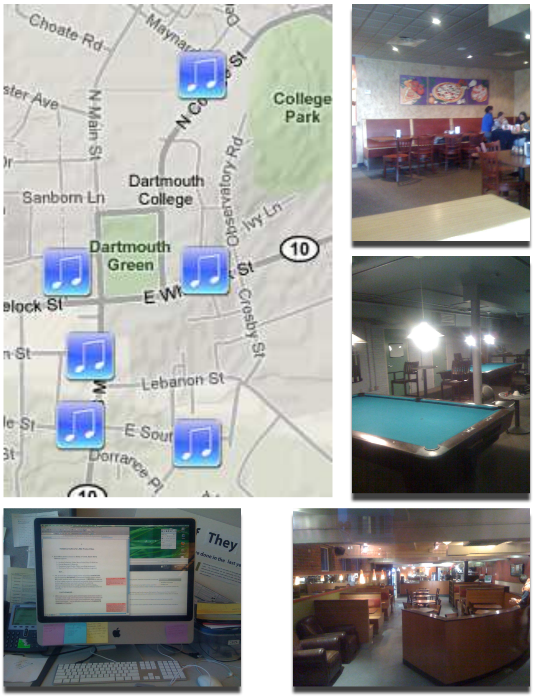


Figure 9: A screenshot of the location where music is detected during our experiment of the participatory sensing application. Surrounding the map are images that were taken according to the prompts.

Sense. However, what makes our work novel in comparison to the literature is that we address a broad set of real-world challenges in the implementation of a scalable sound sensing system on resource limited mobile phones. This has not been previously achieved, such as the unsupervised adaptive classification scheme discussed in Section 4.4.

Existing work that considers problems such as sound recognition or audio scene recognition do not prove their techniques on resource limited hardware. One exception is [14] which is focused on performing sound processing using wearable computers. However, the authors collect data from wearable devices and do offline analysis. In [30] 17 everyday environments are classified (i.e., streets, public transportations, office, living room) with an accuracy of 63.4% using the GMM algorithm; again, the classification process is offline. SoundSense is designed for real-time analysis through the design of feature extraction and scalable classification that is capable of running online on resource limited phones. This has not be achieved before in the literature and distinguishes our contribution.

In [24] [25] the authors use a HMM based strategy capable of classifying 10 environments with good performance using samples of only 3 second duration. This framework provides a way to build new classifiers. However, [24] [25] achieve this by transferring training samples to a centralized sever, which in essence, is simply repeating conventional training steps.This approach neglect the important challenge of automatically discovering interesting new sound events. Instead the system relies on user triggered re-training, which we think over simplifies many of the problems. In practice these approaches are hard to scale due to the heavy burden on user interaction and the energy consumption and privacy concerns of communicating raw audio samples and features to central servers for retraining. In [39], the authors demonstrate a system able to recognize a number of sounds using mobile devices but again this system makes extensive use of off device computational resources offered by back-end infrastructure. SoundSense works solely on the mobile phone and does not rely on offline analysis or back-end interaction. This represents another part of our contribution and one of our design goals.

We also benefited from audio processing research that considers problems other than sound classification. For example, work on speech recognition [32], speaker identification [33], and music genre classification [41]. We build on this body of work as part of the design of the intra-category classification.

8. CONCLUSION

In this paper, we presented SoundSense, an audio event classification system specifically designed for resource limited mobile phones. We described the hierarchical classification architecture that is light-weight and scalable yet capable of recognizing a broad set of sound events. In contrast to traditional audio context recognition systems that are offline, SoundSense performs online classification at a lower computational cost but yields results that are comparable to offline systems. The ambient sound learning algorithm adatively learns a unique set acoustic events for each individual user, and provides a powerful and scalable framework for modeling personalized context. SoundSense carries out all the sensing and classification tasks exclusively on the mobile phone without undermining the main functions of the phone (e.g., making and receiving calls, surfing the web, using IM). We believe the flexiblity and scalability of SoundSense makes it suitable for a wide range of people-centric sensing applications and present two simple proof-of-concept applications in this paper.

9. ACKNOWLEDGMENTS

This work is supported in part by Intel Corp., Nokia, NSF NCS-0631289, NSF IIS-0845683, and the Institute for Security Technology Studies (ISTS) at Dartmouth College. We would like to thank Peter Boda and Chieh-Yih Wan for their support of this work. ISTS support is provided by the U.S. Department of Homeland Security under award 2006-CS-001-000001, and by award 60NANB6D6130 from the U.S. Department of Commerce. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of any funding body. A special thanks goes to our shepherd Bill Schilit.

10. REFERENCES

[1] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.

[2] O. Amft, M. Stäger, P. Lukowicz, and G. Tröster. Analysis of chewing sounds for dietary monitoring. In M. Beigl, S. S. Intille, J. Rekimoto, and H. Tokuda, editors, *Ubicomp*, volume 3660 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2005.

[3] Apple. Introduction to the objective-c 2.0 programming language. Website, 2008. `http://developer.apple.com/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/chapter_1_section_1.html`.

[4] Apple. iphone. Website, 2008. `http://www.apple.com/iphone/`.

[5] Apple. iphone sdk. Website, 2008. `http://developer.apple.com/iphone/`.

[6] L. Bao and S. S. Intille. Activity recognition from user-annotated acceleration data. In A. Ferscha and F. Mattern, editors, *Pervasive*, volume 3001 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 2004.

[7] S. Basu. A linked-HMM model for robust voicing and speech detection. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, 2003.

[8] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2006.

[9] M. Borgerding. Kiss fft. Website, 2008. `http://sourceforge.net/projects/kissfft/`.

[10] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and Srivastava. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSW): Mobile Device Centric Sensor Networks and Applications*, 2006.

[11] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. People-centric urban sensing. In *WICON '06: Proceedings of the 2nd annual international workshop on Wireless internet*, page 18, New York, NY, USA, 2006. ACM.

[12] T. Choudhury, G. Borriello, S. Consolvo, D. Haehnel, B. Harrison, B. Hemingway, J. Hightower, P. Klasnja, K. Koscher, A. LaMarca, et al. The Mobile Sensing Platform: An Embedded System for Capturing and Recognizing Human Activities. *IEEE Pervasive Computing Special Issue on Activity-Based Computing*, 2008.

[13] T. K. Choudhury. Sensing and modeling human networks. Technical report, Ph. D. Thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, 2003.

[14] B. Clarkson, N. Sawhney, and A. Pentl. Auditory context awareness via wearable computing. In *In Proceedings of the 1998 Workshop on Perceptual User Interfaces(PUI98)*, pages 4–6, 1998.

[15] S. Dixon. Onset Detection Revisited. In *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx06), Montreal, Canada*, 2006.

[16] J. Foote. An overview of audio information retrieval. *Multimedia Systems*, 7(1):2–10, 1999.

[17] Google. Android. Website, 2008. `http://code.google.com/android/`.

[18] F. Harris. On the use of windows for harmonic analysis with the discrete Fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[19] N. D. Lane, H. Lu, S. B. Eisenman, and A. T. Campbell. Cooperative techniques supporting sensor-based people-centric inferencing. In J. Indulska, D. J. Patterson, T. Rodden, and M. Ott, editors, *Pervasive*, volume 5013 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2008.

[20] M. L. Lee and A. K. Dey. Lifelogging memory appliance for people with episodic memory impairment. In H. Y. Youn and W.-D. Cho, editors, *UbiComp*, volume 344 of *ACM International Conference Proceeding Series*, pages 44–53. ACM, 2008.

[21] D. Li, I. Sethi, N. Dimitrova, and T. McGee. Classification of general audio data for content-based retrieval. *Pattern Recognition Letters*, 22(5):533–544, 2001.

[22] K. A. Li, T. Y. Sohn, S. Huang, and W. G. Griswold. Peopletones: a system for the detection and notification of buddy proximity on mobile phones. In *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, pages 160–173, New York, NY, USA, 2008. ACM.

[23] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *Int. J. Rob. Res.*, 26(1):119–134, 2007.

[24] L. Ma, B. Milner, and D. Smith. Acoustic environment classification. *ACM Transactions on Speech and Language Processing (TSLP)*, 3(2):1–22, 2006.

[25] L. Ma, D. Smith, and B. Milner. Context Awareness Using Environmental Noise Classification. In *Eighth European Conference on Speech Communication and Technology*. ISCA, 2003.

[26] M. McKinney and J. Breebaart. Features for audio and music classification. In *Proc. ISMIR*, pages 151–158, 2003.

[27] E. Miluzzo, N. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. Eisenman, X. Zheng, and A. Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM New York, NY, USA, 2008.

[28] Nokia. N95. Website, 2008. `http://nseries.nokia.com`.

[29] D. J. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, and H. Kautz. Opportunity knocks: A system to provide cognitive assistance with transportation services. In *UbiComp 2004: Ubiquitous Computing*, volume 3205 of *Lecture Notes in Computer Science*, pages 433–450, Berlin / Heidelberg, 2004. Springer.

[30] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa. Computational Auditory Scene Recognition. In *IEEE International conference on acoustics speech and signal processing*, volume 2. IEEE; 1999, 2002.

[31] V. Peltonen, J. Tuomi, A. Klapuri, J. Huopaniemi, and T. Sorsa. Computational Auditory Scene Recognition. In *IEEE Intl. Conf. on Acoustics Speech and Signal Processing*, volume 2. IEEE; 1999, 2002.

[32] L. Rabiner and B. Juang. *Fundamentals of speech recognition.* 1993.

[33] D. Reynolds. An Overview of Automatic Speaker Recognition Technology. In *IEEE International Conference on Acoustics Speech and Signal Processing*, volume 4, pages 4072–4075. IEEE; 1999, 2002.

[34] J. Saunders, L. Co, and N. Nashua. Real-time discrimination of broadcast speech/music. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 2, 1996.

[35] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/musicdiscriminator. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, 1997.

[36] A. Schmidt, K. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. In *Handheld and Ubiquitous Computing: First International Symposium, Huc'99, Karlsruhe, Germany, September 27-29, 1999, Proceedings*, page 89. Springer, 1999.

[37] I. Shafran, M. Riley, and M. Mohri. Voice signatures. In *Automatic Speech Recognition and Understanding, 2003. ASRU'03. 2003 IEEE Workshop on*, pages 31–36, 2003.

[38] C. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

[39] D. Smith, L. Ma, and N. Ryan. Acoustic environment as an indicator of social and physical context. *Personal and Ubiquitous Computing*, 10(4):241–254, 2006.

[40] M. Spina and V. Zue. Automatic transcription of general audio data: preliminary analyses. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, 1996.

[41] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *Speech and Audio Processing, IEEE Transactions on*, 10(5):293–302, 2002.

[42] S. Vemuri, C. Schmandt, W. Bender, S. Tellex, and B. Lassey. An audio-based personal memory aid. In N. Davies, E. D. Mynatt, and I. Siio, editors, *Ubicomp*, volume 3205 of *Lecture Notes in Computer Science*, pages 400–417. Springer, 2004.

[43] I. Witten, U. of Waikato, and D. of Computer Science. *Weka: Practical Machine Learning Tools and Techniques with Java Implementations*. Dept. of Computer Science, University of Waikato, 1999.

[44] T. Zhang and C. Kuo. Audio-guided audiovisual data segmentation, indexing, and retrieval. In *Proceedings of SPIE*, volume 3656, page 316. SPIE, 1998.

[45] F. Zheng, G. Zhang, and Z. Song. Comparison of Different Implementations of MFCC. *Journal of Computer Science and Technology*, 16(6):582–589, 2001.