Smartphone Audio Based Distress Detection

Anil Sharma, Sarthak Ahuja, Mayank Gautam, and Sanjit Kaul
IIIT-Delhi, India
{anils, sarthak12088, mayank10047, skkaul}@iiitd.ac.in

ABSTRACT

We investigate an unobtrusive and 24×7 human *distress detection and signaling system*, *Always Alert*, that requires the smartphone, and not its human owner, to be on alert. The system leverages the microphone sensor, at least one of which is available on every phone, and assumes the availability of a data network. We propose a novel two-stage supervised learning framework, using support vector machines (SVMs), that executes on a user's smartphone and monitors natural vocal expressions of fear — screaming and crying in our study — when a human being is in harm's way. The challenge is to achieve a high distress detection rate while ensuring that the false alarm rate is a manageable overhead, while a typical smartphone user goes about living life as usual. We train the learning framework with carefully selected audio fingerprints of distress and of varied environmental contexts. The audio is used to tune the learning framework to obtain a *desirable* distress detection rate and false alarm rate (FAR). The ability of the proposed framework to detect distress in rather challenging audio environments is demonstrated. Exploiting the time contiguous nature of false alarms further allows us to reduce the FAR. We show the feasibility of using our framework anytime and anywhere by testing it over many hours of audio fingerprints recorded by volunteers on their smartphones, as they went about their daily routines. We are able to achieve high distress detection rates at an average overhead that is equivalent to about 1 facebook post every 3 to 4 hours.

1. INTRODUCTION

Security systems (burglar alarms, intrusion detection systems) abound for residential and commercial establishments. A breach of security leads to an alarm that is followed by a visit from law enforcement agencies. Often, the visit is preceded by an attempt to confirm that a breach had indeed taken place, that is the alarm was not a *false alarm* [10].

While crimes against individuals are probably as old as humanity, anytime anywhere security for a typical individual going about life as usual is fairly new. It has been made possible by smart devices (for example, smartphones and watches) and wide network coverage. However, the current approaches leave a lot to be desired. Most approaches require the smartphone (or device) owner to be alert enough to raise panic using the phone, often by what is equivalent to pressing a *panic* button. The human needs to be constantly on guard. While this is desirable when passing through a place known for its notoriety, it is unnatural and onerous to be on guard always, especially when in familiar places and amongst familiar faces [1, 5].

We propose *Always Alert* (A^A), a distress detection system, in which a person's smartphone is always on guard. A^A receives audio inputs from the phone's microphone and processes them to detect the presence of screaming and crying, which are known natural responses to distressful circumstances [13]. A^A needs to address two challenges before it can be deployed in the real world. Firstly, it must do its primary job of detecting distress with high accuracy. It must achieve a good accuracy in real world settings in which the background often contains other sounds. These other sounds form the *environmental context* of the phone (and its user). As a result, audio inputs received by a user's phone are not as *clean* as recordings made under controlled and quiet settings. Also, a user may place the phone at various locations, for example, in hand, or in a pocket, or in a bag, further affecting the quality of audio received by the phone's microphone. Secondly, it must trigger a very small number of false alarms. Specifically, the overheads on law enforcement due to false alarms must be minimal.

There are many other works on scream detection [18, 31, 35]. To our knowledge none of them address detection 24×7 over varying environmental contexts. Also, they often use fixed sensors and training over carefully chosen non-scream audio samples. For example, in [18] the authors use microphone arrays and train their learning framework to distinguish between a scream, and sounds created on applause, laughter, crying, glass break, and clap. In contrast, to exemplify the challenges that A^A must address, it must be able to detect screams when a microwave or any other home appliance is on in the background. Also, A^A must not falsely categorize sounds from a home appliance as screams and forward the false detections (alarms) to law enforcement.

A^A follows a four stage approach to distress detection. The first stage, *Speech Filter*, detects distress with high probability. *Speech Filter* is trained using clean scream, crying, and normal speech samples. However, *Speech Filter* leads to a significant (about 10%) false alarm rate. Audio that is accepted as a scream by *Speech Filter* is then processed by *Context Filter*. This stage is trained to understand environmental contexts. It brings down the false alarm rate by an order of magnitude (to about 1%) and has a negligible impact on the true scream detection rate. A FAR of 1% is about 15minutes of false alarms generated by a user of A^A over 24 hours. This is further reduced by *Temporal Analysis*, which exploits the fact that false alarms often occur contiguously in time. The environmental context is responsible for them and often does not change for extended periods of time. All audio samples that pass the above stages as a distress sample are sent to friends-in-the-detection loop before escalation to law enforcement. Evaluation of A^A using volunteer data suggests that the false alarms amount to an average overhead of one (facebook©) post every 3 to 4 hours to friends-in-the-detection loop.

Our specific contributions are as follows.

1. To our knowledge, we are the first to propose an entirely smartphone audio based 24 × 7 distress detection system. All stages starting with recording of audio samples, classification, and raising of alarm, execute on the phone in real time. We provide an outline of the implementation of the application and a thorough evaluation of its suitability to execute on a smartphone.

2. We propose and evaluate a novel multistage distress detection and false alarm rejection framework that is able to achieve high distress detection rates and low false alarm rates, even in the presence of a variety of sounds from environmental contexts. The framework allows selection of a desirable tradeoff between false alarm rate (FAR) and detection rate, across a range of harsh environmental contexts.

3. Evaluation of the feasibility of the use of friends-in-the-detection-loop to further reduce the FAR so as not to create an unnecessary burden on law enforcement.

4. Extensive evaluation of our proposals on many hours of volunteer data that was collected by 16 volunteers using smartphones and going about their usual daily routine. For volunteer collected data we show that overheads due to false alarm rate are on an average equivalent to a facebook post every 3-4 hours.

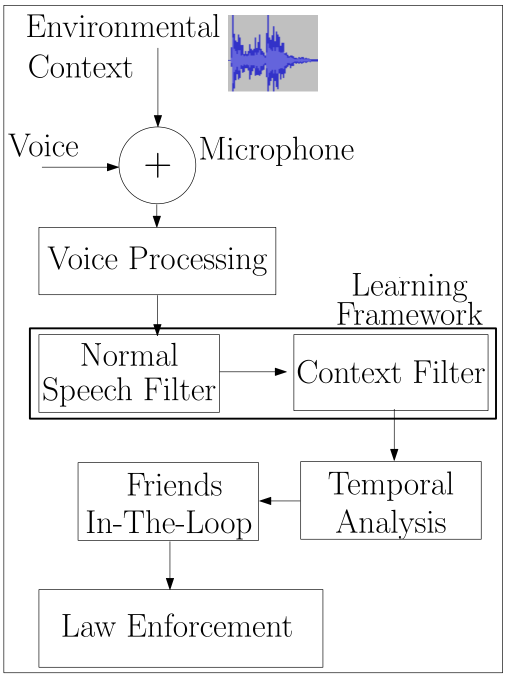


Figure 1: Architecture of A^A.

Rest of the paper is organized as follows. Section 2 describes the architecture of A^A. Sections 3 and 4 describe data collection and processing. Section 5 describes our learning framework, which is followed by a description of temporal analysis in Section 6. Evaluation methodology is described in Section 7. Section 8 and 9 detail the performance evaluation of A^A. Section 10 describes our implementation on Android and its energy footprint. We discuss our limitations and a few possible approaches in Section 11. Related works are described in 12. We conclude in 13.

2. ARCHITECTURE

Figure 1 shows the various blocks that constitute A^A. Note that all of the blocks other than friends-in-the-loop and law enforcement execute as a smartphone application. A^A senses the environmental context and any human audio signals using a phone's one or more microphones. The resulting audio is sent through the voice processing block, which converts the input audio stream into smaller chunks, samples of 2 seconds length, that are each converted into (MFCC) feature vectors. The feature vectors are input to the learning framework that consists of the blocks Normal Speech Filter and Context Filter. The phone categorizes a sample to belong to the distress category only if both the Normal Speech Filter and the Context Filter categorize the sample as distress.

Once a sample is categorized as distress, the phone sets its state to *alarm raised*. The alarm is sent to friend(s)-in-loop together with a snapshot of audio around the time of alarm, who for this study are subscribers to a facebook page. Temporal Analysis ensures that alarms that are contiguous in time and hence are very likely to be raised due to similar reasons are not forwarded again to the friend(s)-in-loop. It also exploits time information in the recorded audio to reject likely false alarm culprits like music, as we will show later.

The friend(s)-in-the-loop act as a final filter. Humans can be very good at rejecting alarms due to environmental context as they can easily detect the lack of a human scream in audio snippets of reasonable clarity [12]. The friend(s)-in-loop can reduce the overhead of false alarms on law enforcement. They may directly escalate to law enforcement or may confirm to the phone application whether the sample detected as distress is in reality distress or not. Of course, they will introduce delays. Later, we will look at the expected delays between the phone posting an alarm and a friend-in-loop responding to it.

The rest of the paper will describe and evaluate all the blocks (except law enforcement). While we only consider sampling audio information, in case an alarm is confirmed to *not* be a false alarm, the phone may send any other required information, for example, location, to law enforcement.

3. DATA COLLECTION

We collected audio that is representative of varied environmental contexts and that captures two expressions of distress, namely screaming and crying. Data collection drives were both *controlled* and *uncontrolled*. Uncontrolled data collection involved 16 volunteers recording data using smartphones while going about their daily routines. On the contrary, controlled data collection was done by selected people and was targeted towards collecting specific kinds of audio data. Also, effort was made to get as clean a recording as possible. For example, the collection was done with phone in hand so that recordings were of good fidelity.

All distress audio data was carried out in a controlled manner. We also collected sounds from a set of environmental contexts in a controlled manner. All collected data is recorded at 44.1KHz and is encoded at 16 bits per sample.

3.1 Controlled Data Collection

Distress Audio Data: In this work we consider only screaming and crying. Screams and cry data was collected from the Internet [7], TV serials, movies, and some data was collected from student actors of age group 18 to 22 years. Silence periods in the beginning and at the end of all recordings were pruned. The pruned audio files were further split into 2 second long audio samples. We have a total of 340 two second long audio samples that represent distress. Out of 340 samples, 315 were from females and the rest few from males.

Normal Human Speech Data: For normal speech data we collected conversations from TV serials (without background music) and student actors. We have a total of 580 two second samples of normal speech data.

Environmental Context Data: We sample audio from five different categories. They are described below.

1. Indoors: This includes the various sounds that are generated when a person is engaged in activities at home and office. Conversations with family members, with people in a meeting at office, dining at home, and any other sounds excluding those generated by equipments and machines found in a typical home or office. We have 8714 two seconds samples in this category. This data was collected by 4 different people using different phones.

2. Outdoors: Outdoors include all sounds when a person is outside. It includes sounds due to vehicular traffic, sounds while commuting in various modes of transport like bus, rail, and car. Other sounds like honking are included. We also were able to collect about 5 minutes of rainfall. In total we have 4513 two second samples under this category. All data was collected using phones.

3. Machinery: This category includes sounds from machines that are commonly used at home and office. Collected sounds include that of Vacuum Cleaners, Hair Dryers, Mixers, Microwave Ovens, Chimney, Flush, Utensils, Shaving Machine, Washing Machine, Rinsing Bowl, fans, air conditioners, and exhausts. The data for this category was collected by a single individual on a mobile and in total we have 3566 two second samples for this category. A small percentage of samples were collected from sources on the Internet.

4. TV: This category includes artificial human sounds and music generated by televisions and radios. We have a total of 3712 two second samples.

5. Gathering: This category includes sounds in situations where many people are together. Large number of people laughing, a crowd cheering, restaurants, malls, markets, seminars, applause by a crowd, and people talking loudly in a metro, are a few examples. This category contains 1641 two-second samples in total. Samples were collected using a phone and from sources on the Internet.

The sounds collected under different categories have small unavoidable overlaps with each other. For example, some of the Home samples had sounds from a TV and music in the background. Similarly, the *Indoors* category will have human speech data in it.

3.2 Uncontrolled Data Collection

Sixteen volunteers (6 females and 10 males) helped us with audio data collection as they went about their daily routines. This data collection was uncontrolled and no instructions about what must be collected were given. The collection was done using smartphones that the volunteers were using as their personal phones. The volunteers were provided with an application that helped them annotate their environmental context. The application nudged the volunteer to annotate as and when a change in sound intensity of greater than 10dB was detected by it. The volunteers had the option of pausing the recording at will. A total of about 250 hours of volunteer data was collected.

4. DATA PROCESSING

The learning framework cannot process audio samples. Data processing involves converting the audio samples into feature vectors that can be processed by the learning framework. We convert every 2 second audio sample into a 12 element vector that consists of Mel Frequency Cepstral Coefficients (MFCC).

MFCC [27] is widely used in many speech and speaker recognition systems. The mel scale transforms linear frequency into mel-frequency, which represents the frequencies as perceived by the human ear. Small variations in frequencies are not noticed in the mel scale. There are a total of 39 cepstral coefficients. We do not include the 0^{th} cepstral coefficient and coefficients 13 onwards as including them did not give us good classification performance. Our feature vector contains cepstral coefficients 1 to 12.

5. LEARNING FRAMEWORK

Our supervised learning framework comprises of the algorithms *Speech Filter* and *Context Filter*. Both the algorithms use support vector machines to classify any input MFCC vector. *Speech Filter* uses a SVM model created using the *distress-and-speech* training set, while *Context Filter* uses a model created using the *distress-and-context* training set. The former set consists of the two categories human speech and scream. The latter consists of a total of 7 categories. The creation of the sets is described in detail in Section 7.

Speech Filter is shown in Algorithm 1. It takes as parameters the audio sample that needs to be classified in the form of the corresponding MFCC feature vector, the distress-and-speech model, and a distance measure. The algorithm returns the sample for further escalation if it is categorized as distress. Else, it discards the sample and returns an empty vector. Note that the *Speech Filter* model contains just the two categories of distress and speech, and thus only one hyperplane [21] that separates speech and distress. As a result the distance estimate $\hat{d}$ returned by the library function *svmpredict*() is a scalar. If $\hat{d} > 0$ for the input sample, the function classifies it as distress. Otherwise, it classifies it as normal speech. The absolute value $|\hat{d}|$ is an indicator of the confidence with which the classification was made.

Speech Filter takes an input parameter, the threshold distance D_T, which allows us to choose the values of confidence for which a test sample must be classified as distress. A threshold distance $D_T < 0$ will mean that *Speech Filter* will also accept as distress, test samples that were earlier classified as speech with low confidence. While this will increase the distress detection rate, it will also increase the false alarm rate. Later in Section 8, we show the tradeoff between the detection rate and the false alarm rate (commonly known as the ROC curve [8]) for $-2 \leq D_T \leq 4.3$. A suitable choice will be arrived at by running the algorithm on the *validation* and verifying it on the *test* data sets.

Algorithm 1 *Speech Filter* algorithm

```
Require: DSM, $V_I$, $D_T$.          ▷ DSM is the distress-and-speech model, $V_I$ the input MFCC vector, and $D_T$ the threshold distance.
Ensure: The output vector $V_O$.
1: $\hat{d}$ = svmpredict(DSM, $V_I$)  ▷ Predict category of $V_I$.
2: if $\hat{d} > D_T$ then             ▷ $\hat{d}$ is the distance from 0.
3:     $V_O \leftarrow V_I$
4: else
5:     $V_O \leftarrow \phi$
6: end if
7:
8: return $V_O$
```

Context Filter is described in Algorithm 2. It uses the distress-and-context model that distinguishes between seven categories, one of them being distress. In general, for K categories the model uses $K(K-1)/2$ hyperplanes, one hyperplane separating each pair of categories. Thus, our model has 21 hyperplanes. For every input sample (MFCC vector) the SVM model outputs a predicted category and also a distance measure vector, say $\hat{\mathbf{d}}$, of length 21. Assume the categories are labeled from $1, 2, \ldots, K$. For ease of exposition, let $K = 3$. Then $\hat{\mathbf{d}}(1)$ stores the distance measure corresponding to the categories 1 and 2, $\hat{\mathbf{d}}(2)$ stores the measure for the categories 1 and 3, and $\hat{\mathbf{d}}(3)$ stores the measure for categories 2 and 3.

If a distance measure corresponding to categories i and j is positive, then we say that i *wins* over j, that is the input feature vector is more likely to belong to category i than to category j. Else, if the distance measure is negative, j wins over i. Having calculated the winning category for each of the $K(K-1)/2$ distance measures, we sort the K category indexes in decreasing order of the number of times they won. Call this sorted list of category indexes as I. SVM (*svmpredict*()) by default returns $I(1)$ as the category of the input sample. That is if $I(1)$ does not store the index $LABEL\_DISTRESS$ of the distress category, SVM will not classify the sample as distress. *Context Filter* allows for flexibility by using the column relaxation parameter C_R, such that if the index of the distress category is amongst the first C_R elements of I, it classifies the input as distress. A value of $C_R > 1$ therefore allows for a larger distress detection rate at the expense of a larger false alarm rate. If *Context Filter* classifies an input as distress, it returns the vector, else it returns an empty vector.

Note that, given our set of $K = 7$ categories, C_R can take the values $1, 2, \ldots, 7$. Also, in *Speech Filter* we will vary the threshold distance D_T over $(-2, 4.5)$. When *Speech Filter* and *Context Filter* are used together, we end up with a large number of choices that can be made over the set of values in the Cartesian product $\{C_R\} \times \{D_T\}$. We will see in Section 8, that the space allows us to choose a good tradeoff between distress detection rate and false alarm rate. In fact, neither *Speech Filter* nor *Context Filter* but the two in series — *Context Filter* placed after *Speech Filter* — gives us the best detection rate and FAR tradeoff.

Table 1 shows an example of how different choice of C_R and D_T help us choose a better detection rate and FAR tradeoff. For example, $D_T = -1.1, C_R = 1$ gives a detection rate of 87.38% and FAR of 1.38%. This tradeoff can be improved by selecting $D_T = 1.9, C_R = 5$. At this point, the detection rate improves while the FAR is unchanged.

D_T/C_R	1	2	5	6	7
-1.1	1.38	2.78	9.03	14.1	57.57
	87.38	91.18	98.88	99.82	100.00
0.4	0.28	1.01	2.75	5.08	5.72
	86.83	90.48	97.81	98.61	98.62
1.9	0.11	0.57	1.38	2.41	2.43
	85.97	88.95	95.83	96.49	96.5
4.3	0.06	0.29	0.73	1.14	1.14
	82.61	84.67	90.70	91.09	91.1

Table 1: An example of how different values of the detection threshold D_T and the column relaxation C_R allow us to choose different detection rate and FAR tradeoff(s). The first row for every value of D_T lists the FAR(s) and the second row lists the detection rates. The values are a subset of the tradeoff(s) in Figure 2b when using *In Series* learning framework.

6. TEMPORAL ANALYSIS

The temporal analysis (TA) block of A^A maintains a state variable which is set when the learning framework classifies an audio sample as distress. The state variable is unset if no new audio samples are classified as distress for a predefined $TIMEOUT$ period. TA has a very simple role. It does not forward any audio samples classified as scream by the learning framework to friends-in-the-loop if the state variable is set. This ensures, for a reasonable setting of $TIMEOUT$, that friends-in-the-loop are not flooded by redundant alarms.

Algorithm 2 *Context Filter* algorithm

```
Require: DCM, $V_I$, $C_R$.                    ▷ DCM is
    the distress-and-context model, $V_I$ the input MFCC
    vector, and $C_R$ is the column relaxation.
Ensure: The output vector $V_O$.
 1: $K \leftarrow$ Number of Categories         ▷ We have 7
    categories.
 2: $\hat{\mathbf{d}}$ = svmpredict(DCM, $V_I$)   ▷ $\hat{\mathbf{d}}$ is the distance
    measure for the $K(K-1)/2$ hyperplanes.
 3: $k \leftarrow 1$
 4: for $i = 1$ to $K - 1$ do
 5:     for $j = i + 1$ to $K$ do
 6:         if $\hat{\mathbf{d}}(k) > 0$ then
 7:             $C(i) \leftarrow C(i) + 1$
 8:         else
 9:             $C(j) \leftarrow C(j) + 1$
10:         end if
11:         $k \leftarrow k + 1$
12:     end for
13: end for
14: $I \leftarrow \{i_1, \ldots, i_N \in [1, N] : C(i_1) > \ldots > C(i_N)\}$
15: for $col = 1$ to $C_R$ do
16:     if $I(col) == LABEL\_DISTRESS$ then
17:         $V_O \leftarrow V_I$ return $V_O$
18:     end if
19: end for
20: $V_O \leftarrow \phi$
21: return $V_O$
```

7. METHODOLOGY

The data collected using controlled collection (see Section 3) is used to create two training sets, namely, the *distress-and-speech* set and the *distress-and-context* set. The distress-and-speech set includes 50% of all distress audio samples and also 50% of all normal human speech samples. In addition, the distress-and-context set includes 50% of the samples that were collected under the five categories representative of environmental context that we described in Section 3.

The samples that are not a part of the training sets are used to create the *validation* sets. We create four validation sets. The validation sets are used to find desirable points of operation (the (D_T, C_R) pairs) of our learning framework.

Audio samples that are not part of any training set and belong to human speech and the five environmental context categories are added to each of the four validation sets. Each sample is added as is (its *Clean* recording), and also at SNR(s) of 40dB, 20dB, and 10dB. A given SNR is achieved for a sample by adding Additive-White-Gaussian-Noise (AWGN) to it in a proportion so as to achieve the said SNR.

The four validation sets differ from one another in the SNR of the distress audio samples added to them. One of the four validation sets includes *clean* distress samples (no noise added), and the others include distress samples at 40dB, 20dB, and 10dB respectively. Note that unlike the other categories that appear in every validation set at a mix of SNR(s), all distress audio samples in a selected validation set are either clean or have an SNR of 40 or 20 or 10dB.

Also, most importantly, the distress samples are added to samples from the environmental context and human speech and not to AWGN noise to achieve the said SNR. This addition is done in an exhaustive manner. That is, if there are n distress samples and a total of m samples of context and human speech, then we get a total of nm samples of distress each of which is added to one of the m other sounds at the selected SNR.

The different SNR(s) of distress across different validation sets simulate the possibly different relative energies of vocal expression of distress by an individual and that of the surrounding context as sensed by the phone. Note that it is almost impossible to get real data for all possibilities. So we use multiple validation sets to simulate varied environmental contexts, very quiet (high distress SNR) to very harsh (low distress SNR).

8. RESULTS

First, we quantify the performance of *Speech Filter* and *Context Filter* over our validation sets using different SNR(s) as explained in Section 7. The performance evaluation motivates our learning framework, which has *Speech Filter* and *Context Filter* in series. Next, having chosen our learning framework, and a set of desirable column relaxation and distance threshold, (C_R, D_T), pairs, we evaluate the performance of the framework on test data collected by volunteers. This is followed by quantifying the impact of temporal analysis on the false alarm rate obtained from volunteer data.

Note that the learning framework categorizes each input feature vector, which is generated from a 2 sec audio sample. False alarm rate (FAR) is the percentage of such feature vectors, amongst the total non-scream feature vectors, that are wrongly classified as distress. As we go from the learning framework to friends-in-the-loop, we will, instead of stating the FAR, state the average number of posts per hour received by friends-in-the-loop that are false alarms.

8.1 Validation set evaluation and selection of the learning framework

Our summary observations are as follows.

1. Both *Speech Filter* and *Context Filter* do well individually when *Clean* distress audio samples are used.

2. For an SNR of 40dB and less, *Context Filter* leads to very low distress detection rates.

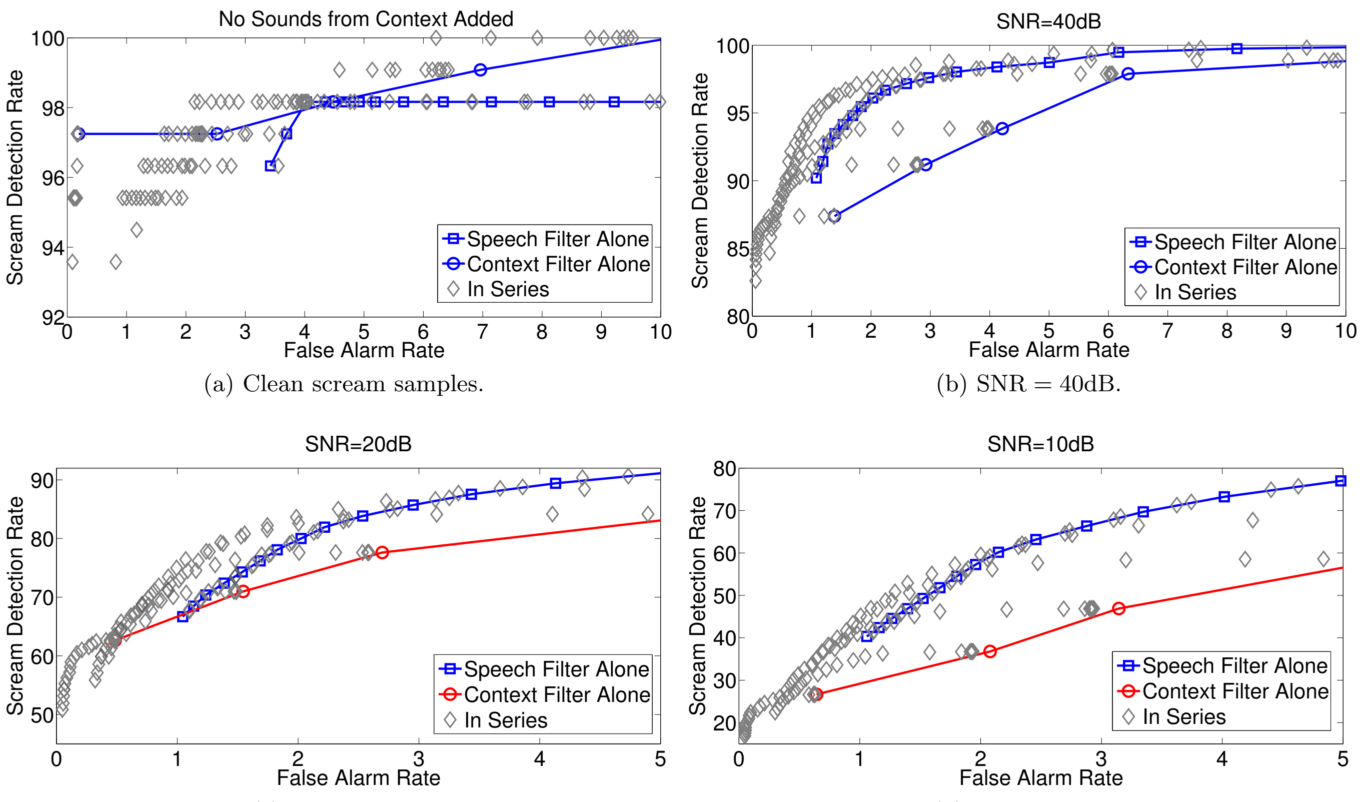
3. *Speech Filter* achieves high detection rate. However, the best tradeoff between detection and false alarm rate is achieved when an audio sample is processed by *Speech Filter* and *Context Filter in series.* Specifically, every sample is first processed by *Speech Filter* and in case *Speech Filter* classifies the input to belong to the distress category, the input is processed by *Context Filter.* The input is declared to be of the distress category only if *Context Filter* also classifies it to be distress.

The detection rate and FAR tradeoff(s) achieved are plotted in Figure 2. Figures 2a, 2b, 2c, and 2d, show the tradeoff for distress samples that are *Clean* (no noise added), have 40dB, 20dB, and 10dB SNR, respectively. For each SNR we show the tradeoff achieved when only *Speech Filter* is used, only *Context Filter* is used, and *Context Filter* follows (is used in series with) *Speech Filter.* When only *Speech Filter* is used the different detection rate and FAR tradeoff(s) are obtained by varying the distance threshold D_T. When only *Context Filter* is used, the different tradeoff(s) are achieved by varying the column relaxation C_R.

As is seen in Figure 2a, there is not much to choose between the three possibilities when the distress samples are clean. For all other SNR(s) *Speech Filter* alone achieves much higher detection rates than *Context Filter* alone. However, note that the best tradeoff between detection rate and FAR is obtained when using the two *In Series.* For example, consider SNR=40dB (Figure 2b). At about a false alarm rate of 1%, using *In Series* can give a detection rate of 5% more than using *Speech Filter* alone. Larger gains in detection rate are seen for 20dB SNR (Figure 2c). At SNR=10dB, detection rate is very low. However, for all the SNR(s), the choice of *In Series* leads to detection rate and FAR tradeoff points that are to the left and top of the points obtained when using *Speech Filter* and *Context Filter* alone. Thus, *In Series* gives a better tradeoff.

8.2 Choosing a set of desirable points of operation from $D_T \times C_R$

Each of the tradeoff(s) corresponding to *In Series* in Figure 2 is obtained by a specific setting of D_T and C_R. We now choose pairs of values such that a detection rate of about 80% is achieved at a FAR of about 1%. For



(a) Clean scream samples.

(b) SNR = 40dB.

(c) SNR = 20dB.

(d) SNR = 10dB.

Figure 2: The false alarm rate and detection rate tradeoff curves obtained when using each of *Speech Filter* and *Context Filter* alone, and *In Series*. The curves were plotted for distress samples and context related samples in the validation set for distress sample SNR(s) of 10, 20, 40dB, and the case (*Clean*) when no sounds from the environmental context are added to the distress samples.

10dB, a detection rate of 80% cannot be achieved at a FAR of 1%. We select two points P_5 and P_6, where P_5 achieves a detection rate of about 80% at a high FAR of about 6% and P_6 achieves a detection rate of 50% at FAR of about 1%. The selected pairs are shown in Table 2. The selections for the *Clean* distress samples are able to achieve FAR of about 1% even at very high detection rates.

8.3 Evaluation over volunteer data

Our evaluation shows that the false alarm rate is brought down to an average of about one message every three hours using the learning framework and temporal analysis. Assuming that an average individual spends about 12 hours every day outside home, 3 to 4 messages or posts to friends-in-the-loop are the overhead that A^A creates in lieu for always staying alert to distress.

Note that Table 2 lists the FAR and detection rate as achieved for the validation data set. We now evaluate performance over the test data set, which consists of data collected by volunteers' phones as they went about their daily routine. The test data set does not include any real screams.

	Clean	Clean	40dB	20dB	10dB	10dB
Name	P_1	P_2	P_3	P_4	P_5	P_6
D_T	4.0	3.7	2.2	1.9	0.4	2.2
C_R	3.0	2.0	4	5.0	7.0	5.0
FAR	1.29	0.99	1.02	1.35	5.71	1.29
DR	96.33	95.41	95.03	79.30	79.62	50.81

Table 2: Selected points of operation $P_1, \dots, P_5$. For each point we list the value of the distance threshold D_T, the column relaxation C_R, and the achieved FAR and detection rate. These points are used to evaluate the FAR obtained from volunteer data. We choose two sets of values for *Clean* and 10dB.

We evaluate the FAR over volunteer data for the points of operation P_1 to P_6. The FAR is plotted for each volunteer in Figure 3. The FAR for the six points of operation is stacked over one another. For all volunteers, the settings P_1 and P_2 give very small FAR. P_3

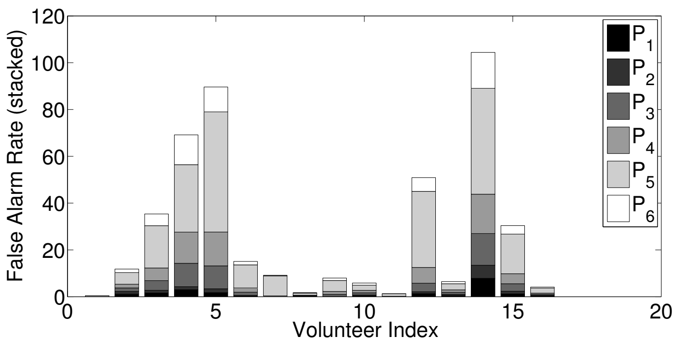


Figure 3: FAR experienced by volunteers for the selected points of operation P_1 to P_6.

works well for some of the volunteers. P_5 leads to really large false alarm rates. Note that the variation of FAR with the point of operation is not surprising (also see discussion in Section 11) and was also seen in Figure 2. The higher FAR, for example when using P_5, is because the chosen detection rate and FAR tradeoff leads to a very high detection rate, however at a very large FAR. At 40dB SNR, P_5 leads to a detection rate of 98.62% and a FAR of 5.72% when using *In Series*. Instead, when P_3 is selected we get a detection rate of 95.03% and FAR of 1.02%.

For all further evaluation of volunteer data we assume the point of operation to be P_1. The median FAR over all volunteers for this choice is 0.45% and the mean FAR is 0.8978%. The median and mean drop to 0.58% and 0.388% when we exclude volunteer 14. As is seen in Figure 3, volunteer 14 generates a large percentage of false alarms, for point of operation P_1, in comparison to other users. Our investigation of volunteer 14's recordings revealed music and television to be a big reason for the high FAR. On the flip side, this also leads to the false alarms being contiguous in time. Temporal analysis brings about a big reduction in the number of false alarms that are forwarded to friend(s)-in-the-loop. Row 6 of Table 3 shows the reduction for volunteer 14. A total of 1194 alarms generated by the learning framework are reduced to merely 11.

8.4 Reduction in FAR on temporal analysis

Table 3 shows the number of false alarms generated per volunteer assuming contiguous alarm rejection with timeout values of 30, and 60 minutes respectively. If we consider volunteers for whom more than 10 hours of data was recorded, on an average we see a false alarm every 2.5 and 3 hours for timeout settings of 30 min and 60 min respectively.

9. FRIENDS IN THE LOOP

We chose 3 facebook pages and 2 groups to evaluate the delay between a new post and the first few responses (comments) to it. This delay is likely to be a good indicator of delays that A^A will suffer due to the requirement that friends-in-the-loop verify an alarm to be a genuine alarm before it is forwarded to law enforcement. Since the humans that constitute the friends-in-the-loop for any individual is likely to be a small community, the worst case delays may be up to 15 minutes. Note that these delays may reduce in case a directed message is sent to a priori selected group of people, for example, the family of the individual likely to be in distress.

We polled the most recent 300 posts from each page. From each post we obtained the earliest 5 comments. Only posts with at least 1 comment were considered for this experiment. For each post and each of the 5 earliest comments on it, we calculate the difference in the time of posting of the post and the comment. Table 4 summarizes the median of the obtained time differences. Page 3 and page 4 have a very limited number of members and we see median time delays of about 15 minutes between a post and the very first comment on it.

	False alarms sent to friends-in-loop		
Tot. Hours	No Timeout	30 min	60 min
57.28	304	20	18
39.10	55	3	3
19.27	466	8	7
18.49	25	3	3
18.33	7	2	2
13.51	1197	14	11
11.09	49	8	6
3.15	12	2	2
0.97	30	3	2
0.69	35	1	1

Table 3: Each row corresponds to a different volunteer. The first column is the total number of hours of recorded audio from each of the volunteers. The second columns lists the number of false alarms sent to friends-in-the-loop assuming that we do not exploit the time contiguous nature of the false alarms. Third and fourth column show the number of forwarded false alarms for timeout values of 30 and 60 minutes, respectively.

10. SMARTPHONE IMPLEMENTATION AND ENERGY CONSIDERATIONS

Figure 4 summarizes the implementation of the A^A Android application. We used the CoMIRVA [32] library for MFCC computation. The library libsvm [11] was used for SVM based classification. We now enumerate the challenges faced and the solutions we have implemented.

Buffer overflow: There are two Android libraries, AudioRecord and MediaRecorder, which allow recording audio from the microphone. MediaRecorder only

Comment	Pg. 1	Pg. 2	Pg. 3	Pg. 4	Pg. 5
1	1.233	0.892	15.52	14.48	2.2
2	1.933	1.325	42.71	16.57	2.917
3	2.817	1.683	80.13	22.02	3.833
4	3.65	2.025	112.6	39.43	4.542
5	4.742	2.517	124.2	54.47	5.333

Table 4: For each page, we tabulate the median time between a post and the first to the fifth comment on it. Page 1 has 320000 followers. Page 2 has 3.6 million followers. Page 5 has 68000 followers, page 3 has 622 members, and page 4 has a mere 79 members.

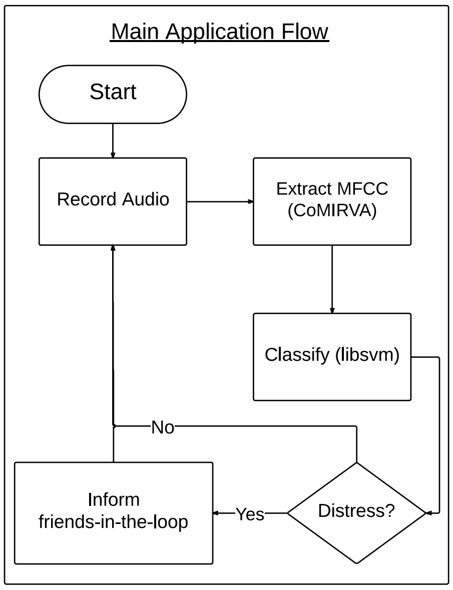


Figure 4: The different functional blocks of the A^A Android application.

provides audio in compressed format. As we need to acquire raw PCM data, we used AudioRecord. The AudioRecord API uses the polling approach to acquire PCM frames from the audio buffer, which overflows quickly if not polled. We had to ensure that the thread we created for recording worked in tandem with AudioRecord's thread that polls the buffer so as to ensure that all recorded audio is saved.

Multi-core implementation: On phones that have multiple cores, we realized that the recording thread was slower than the polling thread since the threads were running on different cores. This led to creation of null frames, which would get saved as a series of 0s and hence corrupt the recording. To make sure these null frames do not get saved, we put each frame through a test to check for null frames.

User privacy: We did not want any audio file saved by the application to show up in the media player. Note that the application may store about 5MB of audio every minute, and the file will need to be saved on the phone's external storage. To hide the audio file we use a combination of three techniques. We do not insert the header into the file, we place a "nomedia" marker file in the folder, and make the folder a hidden system folder.

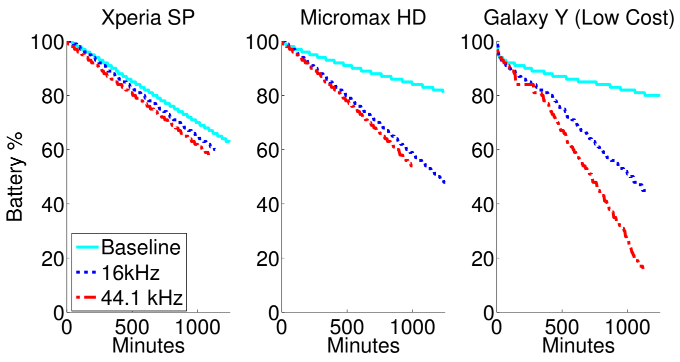


Figure 5: Energy consumed for recording at different sampling rates, by three different phones.

10.1 Energy consumption

We measured the energy consumption of recording audio from the microphone and classification using SVM on phone. The experiment was conducted on three Android phones, namely, Samsung Galaxy Y (Android 2.3.6), Sony Xperia SP (Android 4.1.2) and Micromax Canvas HD (Android 4.2.1). Amongst these phones, Galaxy Y is a 2011 model and has low cost hardware, whereas the other two are 2013 models with fairly powerful hardware. The experiments were conducted in two stages, one for recording and the other for classification using libsvm.

Figure 5 summarizes the energy consumption during recording. The more powerful phones with newer versions of Android tend to perform better in terms of energy consumption. The sample rate at which the audio was recorded did not create much difference in the energy consumed on Xperia and Micromax. On Galaxy Y it created a huge difference.

The figure does not show the results of the classification experiment. There wasn't a significant difference in terms of energy consumption and classification took merely $3-5\%$ battery over 10 hours, making classification using libsvm of not much concern with respect to energy consumption. While recording is energy intensive new phones are more optimized for it. In fact, optimizations that improve the audio potential of smartphones are likely to arrive sooner than later.

11. LIMITATIONS AND A FEW OBSERVATIONS

On the false alarms: An average of a post every three hours can soon become a nuisance, especially if the posts are broadcast to a large community. Also, because almost all of these posts will be false alarms. One may instead choose to send the posts to a smaller group of known people, for example, parents or a spouse. This can lead to privacy concerns, however.

It is worth investigating the ability of the phone to guess the environmental context in which an alarm was raised. Certain contexts may then be treated as safe and alarms raised may be ignored. Of course, the user of A^A may be nudged and only in case of a previously decided response from the user, the alarm can be discarded by A^A.

Throughout this work we have only used inputs from the microphone to detect distress. Inputs from other sensors like accelerometer can help better gauge context and reduce false alarms. GPS or WiFi based location information together with audio fingerprints can help reject repeat false alarms that are tied to a particular location. An example location is a children's park where screaming children lead A^A to raise false alarms with certainty.

There are other scenarios too. They are not very likely to occur, however, at least not on a daily basis. The framework cannot distinguish between a happy scream (no cause for alarm) and a sad scream, at least not without involving friends-in-the-loop. It cannot currently distinguish between a scream on TV and a real human scream.

Why choose facebook: We chose facebook to evaluate the delays that friends-in-the-loop may lead to. Chat messengers like WhatsApp will likely lead to smaller delays. However, unlike facebook, we do not have access to API(s) needed to get the required information. Also, our calculations of delays assume that people will over time respond to posts made by A^A.

12. RELATED WORK

In this section, we summarize related works on distress detection using audio on mobile phones and wearable devices. We will also discuss a few alert dissemination techniques.

12.1 Distress detection

Distress detection from atypical events, using audio and video information, is found in [25, 36, 37]. Works that only use audio to identify normal and abnormal events include [13, 18]. For detection of distress, scream is chosen expression of distress [18, 31, 35].

[18] uses the categories of scream, cry, laugh, applause, clap, knock, and glass break to train SVM and GMM with the aim of distinguishing between normal and abnormal events in a home setting. They use a Linux based system (pentium Mobile CPU 2G, 400M front Bus) and a microphone array.

[35] identifies screams and gunshots using a microphone array and also localizes the abnormal audio event by pointing a camera to the exact location of the event using the Time Difference of Arrival technique. Two parallel GMM(s) were trained to discriminate noise from scream and noise from gunshot. They used a sophisticated feature extraction and selection method to include MFCC along with periodicity, spectral slope and centroid. [31] also detects gunshot, explosion, and scream in a metro and subway environment. The above works provided a good motivation for using scream as a category of interest for our distress detection framework. They discriminated between threatening and non-threatening situations using GMM and HMM. These works show high accuracy for scream detection. However, they have use a dedicated and static hardware for their framework. Also, unlike A^A they cater to very select environments.

In our work, we carry out a thorough evaluation of scream detection across varied environmental context and different degrees of sounds from the environment. All related works have considered a specific contexts [15, 22, 23, 29, 31].

Related works that have used a multi-stage framework include [13, 15, 22, 29, 30, 31]. These systems have limited context information, whereas we consider a wide range of possible contexts.

[28] attempts to identify the emotion of the user using GMM and HMM. Emotion Recognition [19, 20, 24, 33] is used to classify six basic type of emotions suggested by Paul Ekman [17]. The emotion recognition work has also been used in fear detection [13]. Their database [14] contains abnormal, non-verbal, and dangerous situations. It was used to make a text-and-speaker-independent fear detection framework. Their accuracy is not high, which is not useful for a system like A^A. Emotion recognition is found in other applications as well [9], but these work within a specific context.

12.2 Context detection

[12] emphasizes the need of only audio based systems instead of both video and audio. But the performance shown was limited to non-speech and non-music data, hence a real time deployment of this work is still a question. People have used Audio in Human Activity recognition tasks [22] to recognize illegal activities like trespassing and hunting. The accuracy is very low in presence of noise. [30] uses wavelet transform at its first stage of classification and has considered many context audios. They used a dedicated hardware, however. Audio fingerprints have been used in elderly care to detect falls using mobile and wearable devices [16, 26]. [26] uses many sensors along with a microphone on a wearable device to identify depression using silence and speech periods. [34] uses a microphone and other sensors to adapt to user context.

12.3 Distress Detection using mobile phones

To our knowledge, there is no work on audio based automated distress detection using mobile phones. Related works on distress detection using mobile phones provide a manual trigger to the user which needs to be pressed when the user is in distress [2, 3, 4, 6]. [2] provides an alert dissemination framework which shows alerts from recent past so that a user is warned in advance about a harmful place.

13. CONCLUSIONS

We proposed an entirely smartphone audio based 24×7 real time distress detection system. We provided an outline of the implementation of the application and a thorough evaluation of its suitability to execute on a smartphone. We proposed a novel framework, that uses two stage learning, temporal information, and ability of humans to reject false alarms based on audio clips to achieve a high distress detection rate and a reasonable false alarm rate. Extensive evaluation on many hours of volunteer data was used to show that the overheads due to false alarm rate are on an average equivalent to a facebook post by the user of A^A every 3-4 hours.

14. REFERENCES

[1] http://www.feminist.com/antiviolence/facts.html.

[2] http://www.fightbackmobile.com/welcome.

[3] http://www.indianexpress.com/news/now-smartphone-app-to-report-sexual-harassment/1158165/.

[4] http://www.pressdontpanic.com/.

[5] `http://164.100.47.134/intranet/Crimeagainstwomen.pdf`.

[6] `https://play.google.com/store/apps/details?id=com.startv.gumrah\&hl=en`.

[7] `http://www.sounddogs.com/results.asp?Type=1,\&CategoryID=1033\&SubcategoryID=24`.

[8] R.o. duda, p.e. hart, and d.g. stork, pattern classification, new york: John wiley & sons, 2001, pp. xx + 654, isbn: 0-471-05669-3. *J. Classif.*, 24(2):305–307, Sept. 2007.

[9] F. Al Machot, A. Mosa, K. Dabbour, A. Fasih, C. Schwarzlmuller, M. Ali, and K. Kyamakya. A novel real-time emotion detection system from audio streams based on bayesian quadratic discriminate classifier for adas. In *Nonlinear Dynamics and Synchronization (INDS) 16th Int'l Symposium on Theoretical Electrical Engineering (ISTET), 2011 Joint 3rd Int'l Workshop on*, pages 1–5, 2011.

[10] E. A. Blackstone, A. J. Buck, and S. Hakim. Evaluation of alternative policies to combat false emergency calls. *Evaluation and Program Planning*, 28(2):233 – 242, 2005.

[11] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[12] S. Chu, S. Narayanan, and C.-C. Kuo. Environmental sound recognition using mp-based features. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1–4, 2008.

[13] C. Clavel, I. Vasilescu, L. Devillers, G. Richard, and T. Ehrette. Fear-type emotion recognition for future audio-based surveillance systems. *Speech Communication*, 50(6):487 – 503, 2008.

[14] C. Clavel, I. Vasilescu, L. Devillers, G. Richard, T. Ehrette, and C. Sedogbo. The safe corpus: illustrating extreme emotions in dynamic situations. In *The Workshop Programme Corpora for Research on Emotion and Affect Tuesday 23 rd May 2006*, page 76.

[15] M. d A. Sehili, B. L., M. Vacher, F. Portet, D. Istrate, B. Dorizzi, and J. Boudy. Sound environment analysis in smart home. *Third International Joint Conference, AmI 2012, Pisa, Italy, November 13-15, 2012. Proceedings*, 7683(13-15):208–223, November 2012.

[16] C. Doukas and I. Maglogiannis. Human distress sound analysis and characterization using advanced classification techniques. In *5th Hellenic Conference on AI, SETN 2008, Syros, Greece, October 2-4, 2008. Proceedings*, pages 73–84, 2008.

[17] P. Ekman. *Basic Emotions*, pages 45–60. John Wiley & Sons, Ltd, 2005.

[18] W. Huang, T.-K. Chiew, H. Li, T. S. Kok, and J. Biswas. Scream detection for home applications. In *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, pages 2115–2120, 2010.

[19] K. Krishna Kishore and P. Krishna Satish. Emotion recognition in speech using mfcc and wavelet features. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 842–847, 2013.

[20] X. Mao, L. Chen, and L. Fu. Multi-level speech emotion recognition based on hmm and ann. In *Computer Science and Information Engineering, 2009 WRI World Congress on*, volume 7, pages 225–229, 2009.

[21] S. Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[22] S. Ntalampiras and I. Potamitis. Detection of human activities in natural environments based on their acoustic emissions. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1469–1473, 2012.

[23] K. Okutani, T. Yoshida, K. Nakamura, and

K. Nakadai. Outdoor auditory scene analysis using a moving microphone array embedded in a quadrocopter. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3288–3293, 2012.

[24] S. Pathak and A. Kulkarni. Recognizing emotions from speech. In *Electronics Computer Technology (ICECT), 2011 3rd International Conference on*, volume 4, pages 107–109, 2011.

[25] Q.-C. Pham, A. Lapeyronnie, C. Baudry, L. Lucat, P. Sayd, S. Ambellouis, D. Sodoyer, A. Flancquart, A.-C. Barcelo, F. Heer, F. Ganansia, and V. Delcourt. Audio-video surveillance system for public transportation. In *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*, pages 47–53, 2010.

[26] M. Rabbi, S. Ali, T. Choudhury, and E. Berke. Passive and in-situ assessment of mental and physical well-being using mobile sensors. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 385–394, New York, NY, USA, 2011. ACM.

[27] L. R. Rabiner and R. W. Schafer. Introduction to digital speech processing. *Found. Trends Signal Process.*, 1(1):1–194, Jan. 2007.

[28] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. Emotionsense: A mobile phones based adaptive platform for experimental social psychology research. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, pages 281–290, New York, NY, USA, 2010. ACM.

[29] J.-L. Rouas, J. Louradour, and S. Ambellouis. Audio events detection in public transport vehicle. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 733–738, 2006.

[30] J. E. Rougui, D. Istrate, and W. Souidene. Audio sound event identification for distress situations and context awareness. In *Enginee ring in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 3501–3504, 2009.

[31] N. S., I. Potamitis, and N. Fakotakis. On acoustic surveillance of hazardous situations. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 165–168, 2009.

[32] M. Schedl, P. Knees, K. Seyerlehner, and T. Pohle. The comirva toolkit for visualizing music-related data. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'07, pages 147–154, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[33] C. Sezgin, B. Gunsel, and C. Hacioglu. Audio emotion recognition by perceptual features. In *Signal Processing and Communications Applications Conference (SIU), 2012 20th*, pages 1–4, 2012.

[34] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong. Sensay: a context-aware mobile phone. In *Wearable Computers, 2003. Proceedings. Seventh IEEE International Symposium on*, pages 248–249, 2003.

[35] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti. Scream and gunshot detection and localization for audio-surveillance systems. *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, 15(5):795–825.

[36] V.-T. Vu, F. Bremond, G. Davini, M. Thonnat, Q.-C. Pham, N. Allezard, P. Sayd, J.-L. Rouas, S. Ambellouis, and A. Flancquart. Audio-video event recognition system for public transport security. In *Crime and Security, 2006. The Institution of Engineering and Technology Conference on*, pages 414–419, 2006.

[37] H. Wang and P. Chu. Voice source localization for automatic camera pointing system in videoconferencing. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 1, pages 187–190 vol.1, 1997.