# Crowd++: Unsupervised Speaker Count with Smartphones

**Chenren Xu**[†], **Sugang Li**[†], **Gang Liu**[§], **Yanyong Zhang**[†]
**Emiliano Miluzzo**[*], **Yih-Farn Chen**[*], **Jun Li**[†], **Bernhard Firner**[†]
[†]WINLAB, Rutgers University, North Brunswick, NJ, USA
[*]AT&T Labs - Research, Florham Park, NJ, USA
[§]Center for Robust Speech Systems, University of Texas at Dallas, Richardson, TX, USA

## ABSTRACT

Smartphones are excellent mobile sensing platforms, with the microphone in particular being exercised in several audio inference applications. We take smartphone audio inference a step further and demonstrate for the first time that it's possible to accurately estimate the number of people talking in a certain place – with an average error distance of 1.5 speakers – through unsupervised machine learning analysis on audio segments captured by the smartphones. Inference occurs transparently to the user and no human intervention is needed to derive the classification model. Our results are based on the design, implementation, and evaluation of a system called *Crowd++*, involving 120 participants in 10 very different environments. We show that no dedicated external hardware or cumbersome supervised learning approaches are needed but only off-the-shelf smartphones used in a transparent manner. We believe our findings have profound implications in many research fields, including social sensing and personal wellbeing assessment.

**ACM Classification Keywords**
C.3 Special-Purpose and Application-Based Systems: Miscellaneous

**General Terms**
Algorithms, Design, Experimentation, Human Factors

**Author Keywords**
Audio Inference, Smartphone Sensing, Speaker Count

## INTRODUCTION

The most direct form of social interaction occurs through the spoken language and conversations. Given its importance, for decades scientists have proposed diverse methodologies to analyze the audio recorded during people's conversations to distill the various attributes that characterize this particular social interaction. In addition to the most obvious attributes of a conversation, i.e., its content [28], several types of contextual cues have also received attention including speaker identification, conversation turn-taking, and characterization of a social setting [9, 22, 13]. We, however, note that one of the most important contextual attributes of a conversation, namely, speaker count, has been largely overlooked. Speaker count specifies the number of people that participate in a conversation, which is one of the primary metrics to evaluate a social setting: how crowded is a restaurant, how interactive is a lecture, or how socially active is a person [27, 24]. In this paper, we aim to accurately extract this attribute from recorded audio data directly on off-the-shelf smartphones, without any supervision, and in different use cases.

Most of the previous studies that focused on the extraction of conversation features all share a common thread: they often require specialized hardware – such as microphone arrays, external dongles pairing with mobile phones, or video cameras – and complex machine learning algorithms built upon supervised training techniques requiring the collection of large and diverse data sets to bootstrap the classification models. The support of powerful backend servers is also often needed to drive these algorithms.

Given that smartphones are becoming increasingly powerful and ubiquitous, it is natural to envision new social monitoring architectures, with the smartphones being the only sensing and computing platform. In pursuit of these goals, we design a system called *Crowd++*, where we exploit the audio from the smartphone's microphone to draw the social fingerprints of a place, an event, or a person. We do so by inferring the number of people in a conversation – but not their identity – as well as their interactions from the analysis of the voices contained in the audio captured by the smartphones, *without any prior knowledge of the speakers and their speech characteristics*. Audio inference from smartphones' microphones has been previously used to characterize places and events by picking up different sound cues in the environment [3]. However, for the first time, we show how to infer the number of speakers in a conversation through voice analysis using the audio recorded on off-the-shelf smartphones.

Crowd++ is unique given its number of contributions: (i) it is entirely distributed, with no infrastructure support; (ii) it applies completely unsupervised learning techniques and no prior training is needed for the system to operate; (iii) it

*UbiComp'13*, September 8–12, 2013, Zurich, Switzerland.
Copyright © 2013 ACM 978-1-4503-1770-2/13/09...$15.00.

is self-contained, in that, the sensing and machine learning computation takes place entirely and efficiently on the smartphone itself as shown by our implementation on four different Android smartphones and two tablet computers; (iv) it is accurate, as shown by experiments where Crowd++ is used in challenging environments with different audio characteristics – from quiet to noisy and loud – with the phone both inside and outside a pocket, and very short audio recordings; and (v) it's energy and resource-efficient.

In spite of Crowd++ not being perfect and potentially affected by limitations – the count is based on active speakers and noise can possibly impact the count accuracy – we still believe that ours is a competitive approach in many different application scenarios. In the social realm for example: people are often interested in finding "social hotspots," where occupants engage in different social behaviors: examples are restaurants, bars, malls, and meeting rooms. What if we could know in advance the number of people in a certain bar or restaurant? It might help us make more informed decisions as to which place to go.

While Crowd++ may be deemed only as an initial step, we show that faithful people count estimates in conversations can nevertheless be achieved with sufficient accuracy. We implement Crowd++ on four Android smartphones and two tablet computers and collect over 1200 minutes of audio over the course of three months from 120 different people. The audio is recorded by Crowd++ in a range of different environments, from quiet ones – home and office – to noisy places like restaurants, malls, and public squares. We show that the average difference between the actual number of speakers and the inferred count with Crowd++ is slightly over 1 for quiet environments, while being no larger than 2 in very noisy outdoor environments. We conjecture that this accuracy is adequate and meaningful for many applications – such as social sensing applications, crowd monitoring and social hotspots characterization just to name a few – that don't necessitate exact figures but only accurate estimates.

## MOTIVATION AND CHALLENGES

Speaker count is an important type of contextual information about conversations. Crowd++ is able to infer the number of speakers in a dialog without requiring any prior knowledge of the speech characteristics of the involved people because of its unsupervised nature. We believe that the ability to capture this information can support different classes of applications, some of which are summarized below.

**Crowd Estimation and Social Hotspots Discovery.** With Crowd++ it would be possible to estimate the number of people talking in certain places, such as restaurants, pubs, malls, or even corporate meeting rooms. This information is useful to assess the occupancy status of these places.

One question that comes to mind is: Why do we need a solution like Crowd++ to infer the number of people in a place? Wouldn't be enough to simply count the number of WiFi devices associated with an access point, piggyback to a bluetooth scan result, measure co-location, use computer vision techniques to analyze the number of people in video images, or even use active methods that require the transmission and analysis of audio tones? The answers to these questions are quite straightforward: none of these techniques in isolation is the solution to the problem. In order to read the association table of an access point there is a need to have access to the WiFi infrastructure, which is often not allowed. Even if possible, a person with several WiFi devices may generate false positives. A count based on the result of a bluetooth discovery [34] is error-prone because of the likelihood of reaching out to distant devices. RF-based device-free localization techniques [36] require the support of an infrastructure of several radio devices. Acoustic-based counting engines as in [14] are error-prone because of surrounding noise and audio sensitivity to clothes. Counting people through computer vision techniques [7] requires customized infrastructure, suffers from privacy concerns, and is limited by lighting condition. Crowd++ inference is instead based on a much more localized event – speech – that can significantly scope the count inference to specific geographic regions. It's also passive, since no active sounds by the devices need to be played.

We generally assume that people usually engage in conversations in social public spaces such as restaurants, bars, or conference rooms. We also acknowledge that in other places, such as subway stations or movie theaters, silence is predominant, making it difficult for Crowd++ to properly operate. We, however, note that Crowd++ should not be deemed as a replacement of any of the existing approaches. Rather, it should be seen as a complementary solution that can be useful to boost the crowd count accuracy by working in concert with different techniques. Prior information about a certain place – such as the average number of people attending the place – combined with the properties of statistical sub-sampling can also be used to boost the final count accuracy.

**Personal Social Diary.** Doctors analyze their patients' social patterns to predict depression or social isolation and take early actions. Rather than using ad-hoc hardware as in [27], which could potentially perturb the quality of the measurements, Crowd++ is installed on the smartphones of people potentially affected by depression and operates unobtrusive monitoring in a much more scalable, and less invasive fashion. These patients' social pattern could in fact be drawn by the social engagement captured by Crowd++ as the patients go about their daily lives.

**Participant Engagement Estimation.** What if a teacher could assess, after a lecture, the level of engagement of their students by simply looking at the number of students participating in discussions during the lecture and the frequency of the discussions? This could be used as an indirect measure of the class engagement and of the teacher's effort in improving the quality of their teaching. Students would in turn be motivated to run Crowd++ on their devices in order to share with their friends, and in turn apprehend from other students, information on the most lively lecture on campus.

### Challenges

As in other smartphone audio inference applications, Crowd++ is affected by some challenges: the phone's location, e.g., in or out of a pocket or bag, smartphone's hardware constraints, and noise polluting the audio are the main limiting factors. Despite these limitations, we show through the development and evaluation of Crowd++ that the system is able to efficiently and accurately perform speaker count in a diverse set of environments and settings.

## PRIVACY

It is quite natural to raise privacy concerns when doing audio analysis. These concerns become more serious when the audio is captured with a smartphone, which is always with the user, even in private spaces. With this in mind, we take specific steps to make sure that users' privacy is preserved.

Speakers' identity is never revealed. Crowd++ isn't able to associate a voice fingerprint to a specific person and it's designed to only infer the number of different speakers in an anonymized manner. Crowd++ could potentially identify only the phone's owner if the algorithm was actively trained to recognize the owner's voice. Identification of the owner may be optionally added to either improve the speaker count accuracy or in personal social diary applications.

The audio analysis is always performed locally on devices in order to avoid sensitive data leaks. The audio is deleted right after the audio features computation. Should communication with backend be needed, the servers should be trusted and off-the-shelf encryption methods for the communications should be put in place. Only features extracted from the audio, rather than the raw audio itself, should be sent to the server.

To guarantee the user's privacy when the data is sent to a backend server and to prevent attacks that exploit the audio features to reconstruct the original audio, measures such as the ones proposed by Liu et al. [18] should be put in place. In this work, it is shown how to manipulate the features to a point that they are still effective for a machine learning algorithm to infer events while, however, obfuscating the underlying content of the raw audio.

Finally, by giving users the ability to configure the application's settings, Crowd++ should be allowed to work only in specific locations – say, in public places. Through geo-fencing technologies, the application could be automatically activated and deactivated as directed by the user's preselected policies: e.g., activate it in the office and in restaurants but not at home.

## SYSTEM DESIGN

Crowd++ estimates the number of active speakers in a group of people. It consists of three steps: (1) *speech detection*, (2) *feature extraction*, and (3) *counting*. In the speech detection phase, we extract the speech segments from the audio data by filtering out silence periods and background noise. In the feature extraction phase, we compute the feature vectors from the active speech data. In the counting phase, we first select the distance function that is used to maximize the dissimilarity between different speakers' voice, and then apply an unsupervised learning technique that, operating on the feature vectors with the support of the distance function, determines the speaker count. An overview of the Crowd++ pipelined approach is shown in Figure 1.

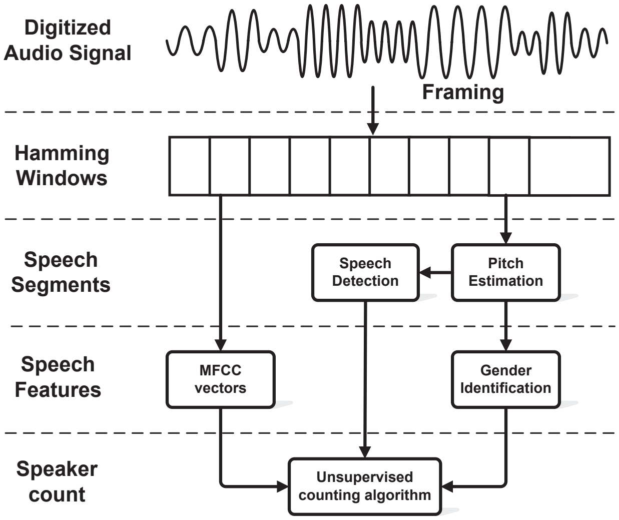


**Figure 1. Crowd++ sequence of operations.**

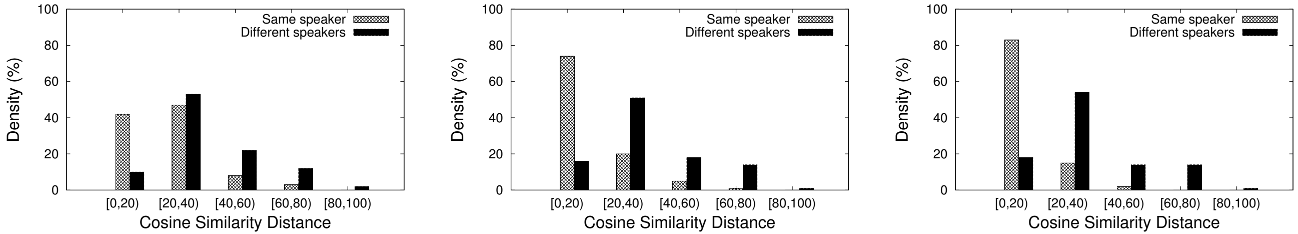
### Speech Detection

As soon as an audio clip is recorded, we segment the clip into smaller segments of equal length. Each segment, which is 3-second long, is the basic audio processing unit. Through experimentation we find this duration to be acceptable for the trade-off between inference delay and inference accuracy. It also captures adequately the turn-taking pattern normally present in everyday conversations [22]. This choice is also supported by previous studies showing that the median utterance duration of telephone conversations between customers of a major U.S. phone company and its customer service representatives is 2.74 seconds [31].

The result of the segmentation of an audio clip $S$ is a sequence of $N$ different segments, $S = \{S_1, S_2, ..., S_N\}$. Next we filter out segments containing long periods of silence or where noise is predominant. We use each segment's pitch value for this purpose.

Pitch [32] is directly related to the speaker's vocal cord, and therefore, by being intimately connected with the speaker vocal trait, it's robust against noise and other external factors. Pitch has been widely used in speaker identification [5] and speaker trait identification [20] problems. When estimated accurately, pitch information can be used to assist the voice activity detection task in a noisy acoustic environment. In this study, we select YIN [8], a time-domain pitch calculation algorithm based on autocorrelation. While some other pitch estimation algorithms, such as Wu [35] and SAcC [17], might exhibit better accuracy, YIN is simpler, more energy-efficient, and robust to noise – hence more suitable for mobile devices.

Traditionally, energy-based methods such as the ones discussed in [11] have been used for voice data detection, but they are unsuitable for processing audio collected by smartphones. When recording audio, smartphones are usually placed at a certain distance from the speakers. As a result, even in absence of speech, the ambient audio energy could be high enough to trigger false positives in energy-based algorithms. Pitch, on the other hand, is a better alternative be-



(a) One-second utterance (b) Two-second utterance (c) Three-second utterance

**Figure 2. Cosine similarity distance demonstrates better speaker distinguishing capabilities with longer utterance.**

cause human pitch is distinctly different than pitch obtained in absence of speech.

We then apply the pitch estimation algorithm on all the segments to only admit those where the pitch falls within the range of 50 to 450 Hz, the typical pitch interval for human voice [4]. In this way, we apply a filtering technique to remove all the segments with long periods of silence or background noise. We note that using pitch to detect speech is not always the best approach because of pitch being only associated with voiced phoneme. However, in our setting, each basic acoustic segment is 3 seconds long with a probability of lack of voiced parts in such a time frame being quite low. In our evaluation, we collected over 1200 minutes audio and verified that pitch is a feasible solution for our purposes.

### Speaker Distinguishing Features and System Calibration

Having filtered out the non-speech and background noise audio segments, our next step is to extract the features that can efficiently distinguish speakers. We have explored various feature sets that are largely used in the speech processing community, such as LPCC [23], RASTA [12], and different combinations of them. We find that MFCC [10] and pitch, when used together, provide the best inference results. In the following, we discuss the details on how these feature vectors are used in our counting algorithm.

#### *MFCC and its Distance Metric*

MFCC is one of the most effective and general-purpose features in speech processing [10]. In Crowd++, we use the coefficients between the 2nd and the 20th coefficient in order not to model the DC (direct current) component of the audio from the first coefficient. A 19-dimensional MFCC vector is then formed out of each 32 msec frame.

In order to perform the counting, we need to rely on a distance metric that allows Crowd++ to distinguish speech from different speakers by comparing MFCC vectors from different audio segments. An ideal distance metric should demonstrate a perfect discriminative capability when computed on data from two different speakers. After investigating several common distance metric options – e.g., Average Linkage and 2-Gaussian Mixture Model (GMM) Generalized Likelihood Ratio (GLR)[1] – we find that Cosine Similarity (CS) is the best candidate as it minimizes the computation overhead in terms of real-time factor (RTF), defined as the processing time per second, and the expected error probability (EEP) metric. The EEP is defined as:

$$\int_{-\infty}^{\tau} p\left(x|\omega_d\right) dx + \int_{\tau}^{\infty} p\left(x|\omega_s\right) dx,$$

where $p\left(x|\omega_s\right)$ and $p\left(x|\omega_d\right)$ represent, respectively, the probability density functions (pdfs) of the distance from the same speaker and different speakers, and $\tau$ is the data point where these two pdfs present the same value. Table 1 shows that the best performance for both the RTF and EEP metrics is achieved using CS. This confirms the superiority of the CS distance compared to a GMM approach, heavily used in the literature in audio processing applications.

| Distance Model | EEP | RTF |
|---|---|---|
| Cosine Similarity (CS) | 0.1687 | 0.003 |
| Average Linkage (AL) | 0.5787 | 0.01 |
| 2-Gaussian Mixture Model (GMM) | 0.5742 | 1.17 |

**Table 1. Cosine Similarity outperforms Average linkage and 2-Gaussian Mixture Model in terms of expected error probability (EEP) and real time factor (RTF) based on 3-second utterances.**

For the audio data processing, we partition the data into smaller segments, and assume the speech within a segment belongs to the same speaker. We then calculate the MFCC vectors for each segment and determine whether two segments belong to the same speaker by looking at their distance. We plot the cosine similarity distance density with different segment lengths (1, 2, 3 seconds) in Figure 2. We observe that the size of the overlap decreases as the length of the segment increases, which confirms the intuition that it is easier to distinguish multiple speakers when longer samples are collected. Finally, Figure 2 also provides hints about the best possible CS distance threshold that allows the differentiation of different speakers.

#### *Pitch and Gender Identification*

In addition to assisting the speech detection process as discussed above, pitch can also be used to identify the gender of the speaker because the most distinctive trait between male and female voices is their fundamental frequency or pitch. The average pitch for men falls between 100 and 146Hz, whereas for women it is usually between 188 and 221Hz, as demonstrated in [4]. By relying on gender identification, Crowd++ speaker count accuracy is increased because of its disambiguation role. For instance, if two participants (one male and the other female) present similar MFCC features, their pitch difference can help distinguish between the two.

[1] We use 2-GMM because a higher order GMM fails to converge in the parameter fitting phase.

### Crowd++ Counting Engine

The last step is about the computation of the speaker count. Having extracted $n$ different audio segments containing human voice, Crowd++ derives the feature vectors from each segment. Let $M_1, M_2, ..., M_n$ be the sequence of feature vectors for all the segments, where $M_i$ is the MFCC feature vectors for segment $S_i$.

Our counting algorithm involves two rounds. In the first round, we aggregate neighboring segments that produce similar features. Traditional speech processing methods use agglomerative hierarchical clustering [15] that requires the comparison between each segment with every other segment in the set, which incurs a computational complexity of $O(n^2)$. We instead employ a much more lightweight clustering method, i.e., forward clustering, which needs to visit all the segments only once. In forward clustering, we start from segment 1 (i.e., $S_1$), and compare it against $S_2$. If their MFCC features are close enough, i.e., $d_{CS}(M_1, M_2) < \theta_s$, we merge these two segments into a new $S_1$. Next we compare this new $S_1$ with $S_3$. If they are still similar, we will merge them too. Otherwise, we stop comparing with $S_1$, and begin to compare $S_3$ and $S_4$. In contrast with hierarchical clustering, forward clustering incurs much less computation and energy overhead given its linear time complexity $O(n)$. The rationale behind forward clustering is that there usually exists temporal correlation in speech – the likelihood of contiguous segments containing the same voice is high when the segments are short enough. After running the forward clustering algorithm, we have fewer and longer segments, to the result of the merging step. We also note that longer segments have better performance in distinguishing different speakers and further boost counting accuracy.

Let's now denote with $C$ the set of inferred speakers. When computing the distance $d_{CS}(i, j)$ between two different feature vectors $M_i$ (which is the MFCC vector from a new segment $i$) and $N_j$ (which is the MFCC vector of a previously inferred speaker, $C_j$) we have three possible outcomes:

- *Existing Speaker:* If $d_{CS}(i, j) < \theta_s$ and we infer a same gender, then we treat these two voice segments as belonging to the same person, namely $C_j$. In this case, we do not update $C$ by adding new inferred speakers, but only update $C_j$'s MFCC vector as $M_i$. If this condition is true for multiple existing speakers, we update the MFCC of the speaker that gives the lowest CS distance.

- *New Speaker:* If $d_{CS}(i, j) > \theta_d$ or different genders are inferred for all the members in $C$, we then tag this voice data as from a new speaker, the $|C| + 1$-th speaker, and add it to the admitted crowd $C$, where $|C|$ denotes the size of $C$.

- *Uncertainty:* If $d_{CS}(i, j) \geq \theta_s$ for all $j$'s but $d_{CS}(i, k) \leq \theta_d$ for some $k$ (both $j, k \leq |C|$), then we cannot decide whether this utterance is from an existing speaker or a new speaker. In this case, we discard this data point.

The $\theta_s$ and $\theta_d$ thresholds are empirically determined in the calibration phase before we conduct the evaluation. We note that the optimal threshold values may vary across different phone models because the microphones have different internal sensitivity levels. The choice of these two thresholds is driven by the desire to be conservative in the discovery of new speakers while minimizing the number of false positives.

To summarize, our counting algorithm is designed to be robust and resource-aware. To this end, we rely on an energy efficient and noise-resilient pitch estimation algorithm, and introduce the cosine similarity distance function, an efficient distance metric at the core of our counting engine.

## EVALUATION

A detailed description of the Crowd++ evaluation results is presented in this section.

### Crowd++ App Implementation

We have implemented Crowd++ on the Android platform using Java and installed it on multiple smartphones – HTC EVO 4G, Samsung Galaxy S2, S3, Google Nexus 4, – and tablets – Samsung Galaxy Tab 2 and Google Nexus 7. The raw audio is recorded at an 8 KHz frequency, 16 bit pulse-code modulation (PCM). We use 32 msec hamming window with 50% overlap for computing the MFCC, and the YIN pitch tracker. The code base of Crowd++ has been optimized to minimize the CPU processing time and energy consumption.

### Energy Considerations

In Table 2, we report the latency for processing 1-second audio segments in terms of MFCC and pitch computation, and the time needed to run the speaker count algorithm on the different devices. The results show that Crowd++ execution time is fast, topping 320 msec and only 171 msec on a Galaxy S3. In addition, we demonstrate Crowd++ energy efficiency in a continuous sensing scenario. We adopt the duty-cycling approach of recording for 5-minute followed by the speaker count algorithm and sleeping for $T$ minutes. We choose the Galaxy S2 phone and plot in Figure 3 the phone's battery duration as a function of the sleep time $T$ between consecutive recordings (similar results can be found for other devices). We observe that even with short sleeping intervals, i.e., 15 minutes, the phone can last up to 23 hours. All the measurements are collected with the WiFi service running in background on the phone. These battery durations are all compatible with the use of a phone in a normal daily routine. It has to be noted that these battery durations are achieved with a fixed duty-cycle policy, providing a performance lower bound. Given that Crowd++ would mostly run in public spaces only, longer sleeping intervals would extend the battery duration even further.

| Latency | HTC | Samsung | Samsung | Google | Google |
|---|---|---|---|---|---|
| (msec) | EVO 4g | Galaxy S2 | Galaxy S3 | Nexus 4 | Nexus 7 |
| MFCC | 42.90 | 36.71 | 24.41 | 22.86 | 23.14 |
| Pitch | 102.71 | 80.36 | 58.11 | 47.93 | 58.33 |
| Count | 175.16 | 150.47 | 89.01 | 83.53 | 70.23 |
| Total | 320.77 | 267.54 | 171.53 | 154.32 | 151.7 |

**Table 2. Average latency for processing 1-second audio for MFCC calculation, Pitch calculation, and speaker counting using different phone models.**

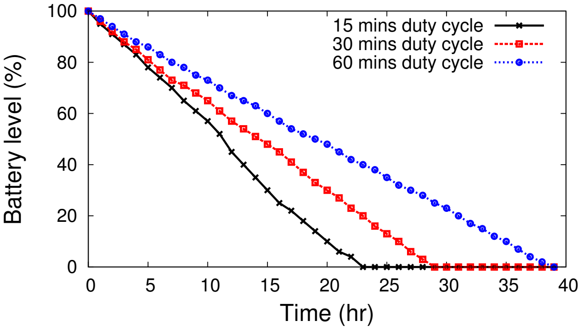


**Figure 3. A duty-cycle of 15 mins guarantees a one day battery life for the Samsung Galaxy S2.**

### Performance Metric

We define *Error Count Distance* as $|\hat{C} - C|$, where $C$ is the actual number of speakers and $\hat{C}$ is the estimated speaker count. The metric is calculated using the absolute value of the error to avoid the terms canceling out because of their positive and negative contributions. The average error count distance is a proxy for the Crowd++ count accuracy.

### System Calibration

Before the feature computation of an arbitrary speech segment, we first need to set appropriate values for the parameters required by the CS metric to properly operate. For this purpose, we have performed a preliminary calibration phase at the beginning of the study, where we collect audio from 10 participants (5 males and 5 females) from different countries with different accents. To guarantee robust calibration, we use different phone models mentioned earlier with different placements (on the table and in the pocket), different distances (in a range of 2 meters), and orientations with respect to the speaker. We empirically chose 15 and 30, respectively, for the $\theta_s$ and $\theta_d$ thresholds used by the cosine similarity distance metric introduced in the previous section. $\theta_s$ and $\theta_d$ are chosen as the median value from $p\left(x|\omega_s\right)$ and $p\left(x|\omega_d\right)$, which is a little off from $\tau$ mentioned earlier to filter out the speech containing overlap and pause.

### Performance with a Single Group of Speakers

We first conduct a set of controlled experiments to benchmark the performance of Crowd++. The experiment consists of 10 different sessions. The first session includes one speaker, and the number of speakers is incremented by 1 in each following session. As a result, the 10th session includes 10 speakers. In each session, every speaker sits at an oval table and speaks in turns as in a conversation. Figure 4 illustrates the experimental setting. We use 7 smartphones for the audio recording – one smartphone (phone 0) is placed at the center of the table; 3 smartphones (phones 1-3) are placed on the table at a distance of 0.5 m, 1 m, and 1.5 m from the center; 3 smartphones (phones 4-6) are placed inside speakers' pockets.

#### *Counting Accuracy vs. Phone Position*

During a conversation, phones are usually placed on the table. Therefore, we look at how the phone's position on the table affects the error count. The results are shown in Figure 5. The results show that Crowd++ is rather robust against various conversation group sizes and phone positions. The error count distance is usually within 1, sometimes 2, and very rarely 3 (in 2 out of 40 cases). From this set of results, we can draw the following conclusions: First, in a quiet indoor environment, Crowd++ gives accurate speaker count estimates. Second, the phone's position on the table does not have an obvious impact on the inference accuracy.

#### *Counting Accuracy when Phones on Table vs. in Pocket*

Figure 6 compares the mean error count distance for phones placed on the table (namely, phones 1-3) and phones placed inside a pants pocket (namely, phones 4-6). We find that in general, phones placed inside a pocket provide larger error count distances, similar to the trend observed in earlier smartphone-based audio sensing studies [25]. As a result, we suggest that in order to achieve accurate speaker count estimates, users should place their phones on the table to extend the sensing range of the microphone.

#### *Counting Accuracy with Different Aggregation Methods*

Given the proximity to the speakers, multiple phones record audio at the same time. We exploit this redundancy and compare different ways of aggregating the results. Specifically, we collect the speaker count estimates from all the 7 phones and show the mean, median and mode of all the samples in Figure 7. The results show that the median and mode value give better speaker count estimates because they are more robust to estimation errors, and mode is better than the median in most of the cases.

### Performance with Multiple Groups

We now investigate the performance of Crowd++ when operating in an environment where different groups of people are next to each other. This is the case of a restaurant for example, with each table occupied by a number of people. In this case, the speech from a nearby group could impact the results of the counting. In order to demonstrate that Crowd++ can work in such a scenario, we have conducted another benchmark experiment to mimic a restaurant setting by having two and three groups of people talking at adjacent tables in the same room. Each group entertains separate conversations occurring in parallel. For each group, two phones are deployed: the first one held by one speaker and the second one in another speaker's pocket. In the two-group scenario, each group has 5 participants, while in the three-group scenario each group has 3 participants. The groups are separated by a 3-meter gap. We record 3 audio clips in each scenario.

We show the estimated speaker count in Figure 8. It is interesting to see that when we have multiple groups talking at the same time, the phones in the pocket have a slightly better performance. This is because the phone in the pocket is still able to pick up the group members' voice while filtering out – for the clothing muffling effect – more distant sounds. Overall, both the phones in each group are able to accurately estimate the speaker count, with an average error distance of 1.5. It is important to realize that only 1 device is sufficient in a group of people to infer the speaker count.

In order to estimate the total number of people in a restaurant, our solution involves having each group estimate its size,

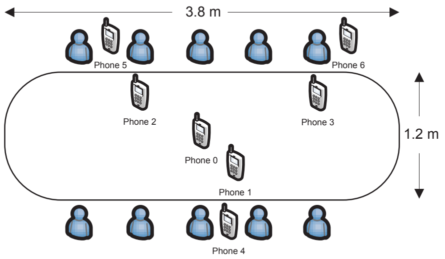


**Figure 4. The phone placement in the benchmark experiments.**

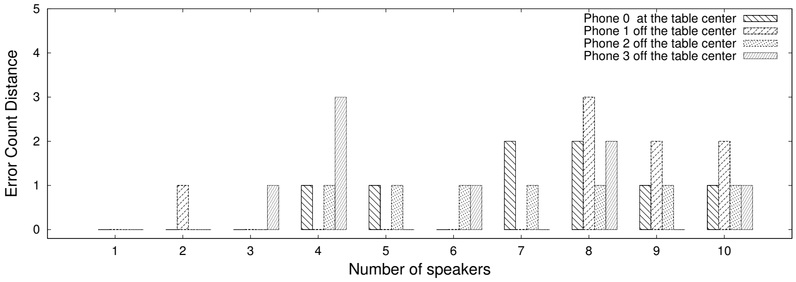


**Figure 5. The counting accuracy does not vary much with the phone position on the table.**

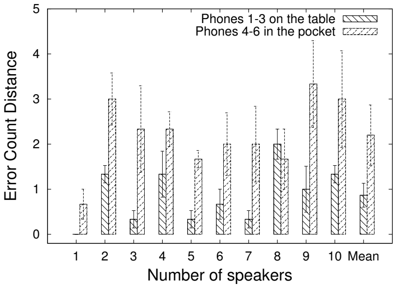


**Figure 6. The phones on the table present a better counting accuracy than the phones inside the pockets.**

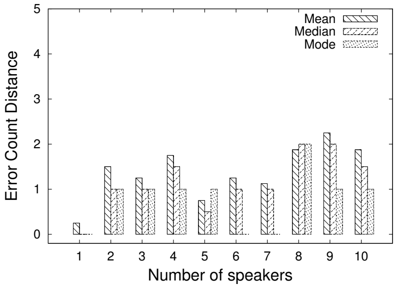


**Figure 7. We achieve better count results when using median or mode from all the devices.**

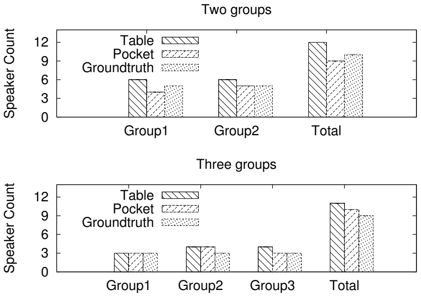


**Figure 8. The phones inside the pockets present better counting results when multiple groups of speakers are co-located.**

and then using the sum as the total people count. In this experiment, we also evaluate the performance of this solution, which is shown in Figure 8 as well. We find that the average error count distance is reported 1 and 2 for the phones placed in the pockets or on the table. As a result, we believe that our divide-and-conquer solution works well in practice, especially considering the privacy concerns involved in uploading the audio features to the cloud for aggregation.

### Performance with Various Conversation Parameters

In reality, many factors could impact the counting performance, such as utterance length, overlapping speech, and the duration of the recorded audio clip. Precisely controlling these parameters at the same time in real world experiments is often unfeasible. For this reason, we follow a common approach in the speech community and generate a separate dataset, as previously shown in [26]. Specifically, we collect audio recordings from 4 male and 4 female participants using a smartphone. We ask each speaker to talk for 3 minutes and record the audio clips. We then segment these clips into smaller segments of random lengths and assemble them to generate audio data. We model the utterance length as a random variable following a log-normal distribution with mean of $\delta$ and standard deviation of 1 according to the configurations used in [31]. By default, each generated audio clip has 2, 4, 6, or 8 speakers, is 8 minutes long, no overlap, and is assigned a value of $\delta = 3$.

#### *Counting Accuracy with Audio Clip Duration*

In this set of experiments, we vary the audio duration from 2, to 4, 6 and 8 minutes. We report the average error count distances with these different audio durations in Figure 9. The results show that to achieve a good counting accuracy, we need longer audio clips. As shown in the plot, 8-minute audio clips are usually long enough to achieve an average error count distance of 1. This is meaningful since we target the inference in social spaces, where usually people tend to remain for more than 8 minutes.

#### *Counting Accuracy with Overlapping Percentage*

Earlier studies [6] show that conversations are often characterized by interruptions of one speaker to another. In this set of experiments, we look at the impact of the percentage of the overlapping speech. We vary the overlapping percentage from 0%, 5%, 10%, 20% to 40% and show the results in Figure 10. We find that the overlapping percentage does not have a noticeable impact on the performance of Crowd++. Even with overlaps of 40% the average error distance of Crowd++ is about 1.

#### *Counting Accuracy with Utterance Length*

In daily conversations, utterance duration can vary according to the setting: people tend to be interrupted more frequently in casual chats and less in formal meetings. We then look at the impact of the utterance length, which we make it of 1, 2, 3, 5, and 8 seconds. The average error count distance is shown in Figure 11. We observe that we have slightly worse results when the utterance length is 1 or 2 seconds, shorter than the processing unit of 3 seconds. Even so, the average error distance is 1.5. When the utterance length is longer than 3 seconds, the average counting error distance decreases to 1.

### Large-scale Experiments

To demonstrate that Crowd++ can accurately count speakers in different conditions, we have installed the app in six

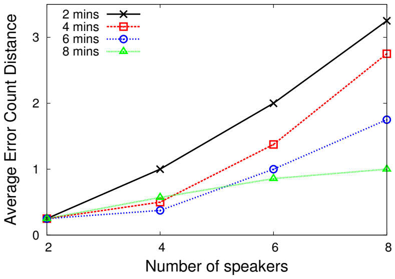


**Figure 9. Eight-minute audioclips are sufficient to achieve an error count distance of 1.**

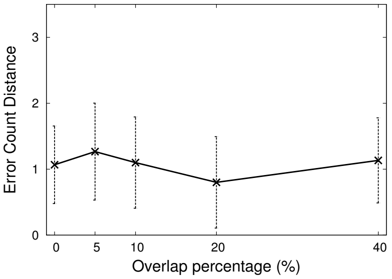


**Figure 10. The average counting error distance is around 1 with up to 40% overlap.**

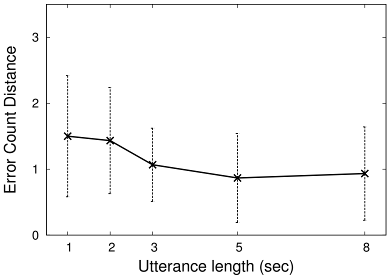


**Figure 11. Longer utterance lengths lead to slightly better counting performance.**

android devices and recruited six volunteers to collect 109 different audio instances with 120 people speakers for a total of 1034 minutes of recorded audio. The conversations are recorded during normal family and friend interactions. In each setting, the participants are within 1 meter from the phone. We can broadly group the audio clips into three categories based on the location of the recordings:

- *Private Indoor Environments:* In this category, audio clips are recorded in quiet indoor environments, including seminar rooms, office and home during, respectively, different events: meetings, lunch and home conversations. The phones are placed on the table during the recording. In spite of these conversations taking place in private indoor settings, background noise is still present, for example paper flipping, door's closing/opening, chair movement, etc.

- *Public Indoor Environments:* In this scenario, audio clips are recorded in different public indoor environments when participants are sitting in restaurants, food courts or moving in supermarkets and shopping malls. The phones are placed in the pocket. The background noise in these environments is mainly generated by surrounding people, music, and various service operations.

- *Outdoor Environments:* The last class of recordings are collected in outdoor places such as parking lots and restaurant outdoor seats, where the background noise mainly comes from cars, wind and other activities. The phones are placed in the pocket during recordings.

Table 3 summarizes the signal-to-noise ratio (SNR) estimation [16], average error count distance (AECD) and the average error count percentage (AECP)[2] from all the experiments. We observe a lower SNR and a higher AECD and AECP, when we move from private indoor, to public indoor and outdoor environments. We also observe that the error count increases when the crowd becomes larger because of more conflicting audio sources. The maximum AECD in private indoor scenarios is 2. In more challenging environments, e.g., public indoors and outdoors, the accuracy degrades. Being Crowd++ designed to infer social hotspots mainly indoors, we conjecture that the indoor error range can be considered adequate for many applications.

**Private Indoor Environments**

| Speaker # | Sample # | Place | SNR | AECD | AECP |
|---|---|---|---|---|---|
| 2 | 4 | Home | 21 | 0 | 0% |
| 3 | 11 | Office | 24.6 | 0.82 | 27.3% |
| 4 | 8 | Office | 21.4 | 1.25 | 31.3% |
| 5 | 7 | Kitchen | 20.9 | 1.28 | 25.6% |
| 6 | 10 | Kitchen | 17.6 | 2 | 33% |
| Overall | 40 | Quiet Indoor | 21.5 | 1.07 | 23.4% |

**Public Indoor Environments**

| Speaker # | Sample # | Place | SNR | AECD | AECP |
|---|---|---|---|---|---|
| 2 | 2 | Restaurant | 8.6 | 0 | 0% |
| 3 | 6 | Food Court | 13.2 | 1.5 | 50% |
| 4 | 7 | Coffee Shop | 8.2 | 1.86 | 46.5% |
| 5 | 17 | Shopping Mall | 12.2 | 1.82 | 36.4% |
| 7 | 12 | Super Market | 13.8 | 1.58 | 22.6% |
| Overall | 44 | Noisy Indoor | 11.2 | 1.35 | 31.1% |

**Outdoor Environments**

| Speaker # | Sample # | Place | SNR | AECD | AECP |
|---|---|---|---|---|---|
| 2 | 4 | Plaza | 16.8 | 0.5 | 25% |
| 3 | 6 | Parking Lot | 16.6 | 1.2 | 40% |
| 4 | 7 | Plaza | 13 | 2.29 | 57.3% |
| 5 | 2 | Parking Lot | 12.2 | 2.5 | 50% |
| 6 | 6 | Patio | 13.9 | 2.67 | 44.5% |
| Overall | 25 | Noisy Outdoor | 14.5 | 1.83 | 43.4% |

**Table 3. The detailed breakdown of the error counts for all the audio clips. We observe that average error count distances and average error count percentage for private indoor is less than in public indoor, and outdoor environments.**

## Crowd++ Use Cases

Finally, to demonstrate the utility of Crowd++, we have implemented three proof-of-concept use cases where knowing the number of speakers in a conversation is important.

### *Where Is the Most Crowded Restaurant?*

Crowd++ can be exercised to find the most crowded restaurant in the area. To provide such a service, we envision there is at least one smartphone at each table running Crowd++, which counts the number of people talking at the table. Then we calculate the total number of people in the restaurant by summing up the people at each table.

In this study, we have recruited participants to record audio at four different restaurants, including formal restaurants, coffee shops, food courts in a mall and in a university student center. The results are shown in Table 3 in the public indoor environments section, where AECD is 1.3. We believe that this level of accuracy should be adequate for this use case since, again, our goal is to infer an estimate of the count in a lightweight manner. More accurate results could be achieved in cooperation with other techniques.

[2] An alternative counting performance metric, defined as $\frac{|\hat{C}-C|}{C}$.

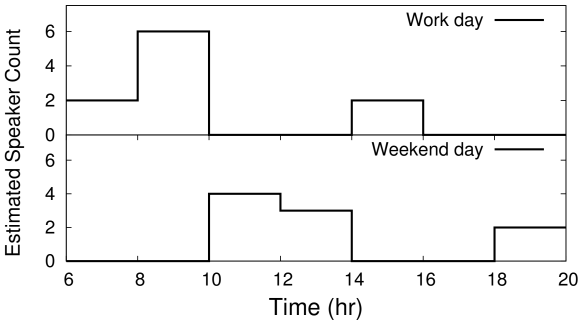


**Figure 12. The social diary of a participant shows that he has different social patterns on work days and weekends.**

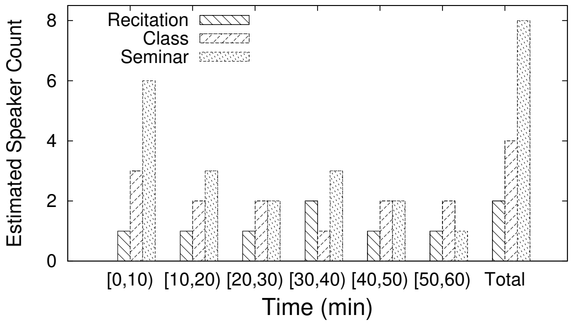


**Figure 13. Seminars and classroom lectures have different interaction patterns over the time.**

*Are you a social person?*

In the second use case, Crowd++ can be used to build a person's social diary – how many people the person talks to, and at what time – which is particularly useful for seniors or people with clinic depression.

In this study, we have recruited three participants, a teacher, a student, and a company employee, who have been using the SocialDiary app to record their conversation log for a week. The SocialDiary app records audio every 2-hour for 8 minutes. We show the log for the student participant on a weekday and a weekend in Figure 12. From the social diary, we observe that the student's day starts much later on the weekend. Also, he talks more in the morning and early afternoon on a weekday and at more recreational events – lunch and dinner – during weekends.

*Is your audience engaged?*

In the third use case, we show that Crowd++ can be used to measure the level of interaction of a lecture or seminar. Such a measurement could allow parents to be aware of their kids' participation in a classroom for example. Or it can be used to annotate a seminar or lecture to fast forward to the part with more active discussions when watching a video or audio recording of the event for example.

In this study, we record 2 regular classes and 2 recitation sessions at a university campus and 4 seminars from an industry lab. Each recording lasts 60 minutes. We segment each audio into six 10-minute segments and estimate the speaker count in each segment, as well as the total speaker count for the whole period. We show how the speaker count varies as time progresses in Figure 13. We observe that the recitation involves less interaction of all – the instructor spends most of the time showing how to solve the homework problems on the blackboard. The regular class has a steady interaction level throughout the duration, while the seminar presents more questions at the beginning.

## DISCUSSION

A possible source of interference is voice generated by TV or radio equipment. Would these background voices cause Crowd++ to over-count? The answer is no. Given the audio modulation techniques applied to TV and radio broadcast it has been proven that audio segments dominated by TV or radio sources can be effectively filtered out [21]. When there is instead significant overlap between people's live voices and TV or radio audio, source separation can be performed [19].

We acknowledge that in some cases the accuracy of Crowd++ could be improved; however, Crowd++ has been designed to perform people counting on mobile devices with no infrastructure intervention and in an energy and resource-efficient manner. Because of this, Crowd++ doesn't rely on complex speech processing algorithms that would yield higher accuracy; its design favors efficiency and mobility support. We are currently exploring further optimizations, such as sparse sampling techniques to reduce the computation overhead. Moreover, noise cancellation achieved with the multiple microphones that can be found in most recent smarpthones would likely provide a further accuracy boost.

## RELATED WORK

We review the most relevant literature on smartphone audio inference applications and speaker count techniques.

### Audio Sensing and Inference on Smartphones

A large body of research demonstrates the use of the mobile phone's microphone to opportunistically analyze audio for event and context characterization. Examples of smartphone context-aware applications are Darwin [25] and SpeakerSense [22] to perform speaker identification. SurroundSense [3] analyzes audio events for place fingerprinting. The EmotionSense project [29] demonstrates the possibility to classify humans' emotions through audio analysis. Ambient noise is leveraged to improve indoor localization results in [33]. All these projects have often in common the use of cumbersome supervised learning approaches, the use of external hardware in some cases, and the need to rely on external servers to operate the learning process. In contrast, Crowd++ is entirely unsupervised, with sensing and processing entirely occurring on the mobile device itself.

### Speaker Counting

Some speaker counting techniques can be found in the literature. The closest related research to Crowd++ is [1] and [26]. Agneessens et al. [1] present a pitch estimation algorithm to recognize a single speaker from audio recordings containing two speakers with 70% of the times correctly estimate the speaker count (referred to as counting accuracy). Crowd++ goes beyond this binary classification approach by tackling a much harder problem, where the number of speakers is up to 10 or even more. Moreover, Crowd++ runs an unsupervised learning algorithm without taking any training data from the target speakers. Ofoegbu et al. [26] present 60% counting accuracy for 4 speakers (versus Crowd++'s 68% counting accuracy under the same conditions and settings) and a generalized residual radio algorithm with a computational complexity of $O(N^2)$ (versus Crowd++'s $O(N)$). Moreover, the data set in [26] is based on staged data from the HTIMIT database [30] containing transcribed speech of American English speakers. Crowd++'s focus instead is the analysis of audio recordings challenged by noise, mobility and obstacles as people go about their daily lives. Another relevant technique is speaker diarization [2], which essentially determines "who spoke when" in an audio recording that contains an unknown amount of speech and also an unknown number of speakers. However, the main objective of diarization is to cluster the homogeneous speech rather than determine the optimal number of clusters. In addition, it usually relies on computationally expensive models (GMM, HMM) and algorithms (BIC, MCMC), which are not suitable for off-the-shelf smartphones.

## CONCLUSION AND FUTURE WORK

In this paper, we presented Crowd++, a scalable and energy efficient speaker count application for smartphones based on the microphone's audio analysis. Crowd++ is novel in many dimensions: it is completely unsupervised and no prior models or external hardware are necessary to operate. It doesn't require any infrastructure and runs entirely on the mobile device. We implemented Crowd++ on different Android platforms and showed, through solid experimentation, that Crowd++ presents adequate inference accuracy in many diverse conditions, from quiet to noisy environments. In contrast to more complex and less scalable counting techniques, Crowd++ is a lightweight approach that can support many different application scenarios: from social sensing – to determine social hotspots – to personal wellbeing assessment and social diary, place characterization, and more accurate localization techniques.

## REFERENCES

1. Agneessens, A., Bisio, I., Lavagetto, F., Marchese, M., and Sciarrone, A. Speaker count application for smartphone platforms. In *Proc. of IEEE ISWPC* (2010).
2. Anguera Miro, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., and Vinyals, O. Speaker diarization: A review of recent research. *IEEE Transaction on Audio, Speech and Language Processing 20*, 2 (2012).
3. Azizyan, M., Constandache, I., and Roy Choudhury, R. Surroundsense: mobile phone localization via ambience fingerprinting. In *Proc. of ACM MobiCom* (2009).
4. Baken, R. *Clinical measurement of speech and voice*. College-Hill Press, 1986.
5. Carey, M., and et al. Robust prosodic features for speaker identification. In *Proc. of ICSLP* (1996).
6. Cetin, O., and Schriberg, E. Speaker overlaps and asr errors in meetings: Effects before, during, and after the overlap. In *Proc. of IEEE ICASSP* (2006).
7. Chan, A. B., Liang, Z.-S., and Vasconcelos, N. Privacy preserving crowd monitoring: Counting people without people models or tracking. In *Proc. of IEEE CVPR* (2008).
8. Cheveigné, A. D., and Kawahara, H. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America 111*, 4 (2002).
9. Choudhury, T., and Pentland, A. Sensing and modeling human networks using the sociometer. In *Proc. of IEEE ISWC* (2003).
10. Davis, S., and Mermelstein, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing 28*, 4 (1980).
11. Haigh, J., and Mason, J. Robust voice activity detection using cepstral features. In *Proc. of TENCON* (1993).
12. Hermansky, H., and Morgan, N. Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing 2*, 4 (1994).
13. Jayagopi, D. B., Hung, H., Yeo, C., and Gatica-Perez, D. Modeling dominance in group conversations using nonverbal activity cues. *IEEE Transactions on Audio, Speech, and Language Processing 17*, 3 (2009).
14. Kannan, P. G., Venkatagiri, S. P., Chan, M. C., Ananda, A. L., and Peh, L.-S. Low cost crowd counting using audio tones. In *Proc. of ACM SenSys* (2012).
15. Karypis, G., Han, E.-H., and Kumar, V. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer 32*, 8 (1999).
16. Kim, C., and Stern, R. M. Robust signal-to-noise ratio estimation based on waveform amplitude distribution analysis. In *Proc. of INTERSPEECH* (2008).
17. Lee, B. S., and Ellis, D. P. W. Noise robust pitch tracking by subband autocorrelation classification. In *Proc. of INTERSPEECH* (2012).
18. Liu, B., Jiang, Y., Sha, F., and Govindan, R. Cloud-enabled privacy-preserving collaborative learning for mobile sensing. In *Proc. of ACM SenSys* (2012).
19. Liu, G., Dimitriadis, D., and Bocchieri, E. Robust speech enhancement techniques for asr in non-stationary noise and dynamic environments. In *Proc. of INTERSPEECH* (2013).
20. Liu, G., Lei, Y., and Hansen, J. H. A novel feature extraction strategy for multi-stream robust emotion identification. In *Proc. of INTERSPEECH* (2010).
21. Liu, G., Zhang, C., and Hansen, J. H. A linguistic data acquisition front-end for language recognition evaluation. In *Proc. of Odyssey* (2012).
22. Lu, H., Brush, A. B., Priyantha, B., Karlson, A. K., and Liu, J. Speakersense: energy efficient unobtrusive speaker identification on mobile phones. In *Proc. of Pervasive* (2011).
23. Markel, J. E., and Gray, A. H. *Linear Prediction of Speech*. Springer-Verlag New York, Inc., 1982.
24. Matic, A., Osmani, V., and Mayora, O. Automatic sensing of speech activity and correlation with mood changes. *Pervasive and Mobile Sensing and Computing for Healthcare* (2012).
25. Miluzzo, E., Cornelius, C. T., Ramaswamy, A., Choudhury, T., Liu, Z., and Campbell, A. T. Darwin phones: the evolution of sensing and inference on mobile phones. In *Proc. of ACM MobiSys* (2010).
26. Ofoegbu, U. O., Iyer, A. N., Yantorno, R. E., and Smolenski, B. Y. A speaker count system for telephone conversations. In *Proc. of IEEE ISPACS* (2006).
27. Rabbi, M., Ali, S., Choudhury, T., and Berke, E. Passive and in-situ assessment of mental and physical well-being using mobile sensors. In *Proc. of ACM UbiComp* (2012).
28. Rabiner, L., and Juang, B.-H. *Fundamentals of speech recognition*. Prentice-Hall, Inc., 1993.
29. Rachuri, K. K., Musolesi, M., Mascolo, C., Rentfrow, P. J., Longworth, C., and Aucinas, A. Emotionsense: a mobile phones based adaptive platform for experimental social psychology research. In *Proc. of ACM UbiComp* (2010).
30. Reynolds, D. A. Htimit and llhdb: Speech corpora for the study of handset transducer effects. In *Proc. of IEEE ICASSP* (1997).
31. Rosenberg, A. E., Gorin, A., Liu, Z., and Parthasarathy, S. Unsupervised speaker segmentation of telephone conversations. In *Proc. of INTERSPEECH* (2002).
32. Sonmez, K., Shriberg, E., Heck, L., and Weintraub, M. Modeling dynamic prosodic variation for speaker verification. In *Proc. of ICSLP* (1998).
33. Tarzia, S. P., Dinda, P. A., Dick, R. P., and Memik, G. Indoor localization without infrastructure using the acoustic background spectrum. In *Proc. of ACM MobiSys* (2011).
34. Weppner, J., and Lukowicz, P. Collaborative crowd density estimation with mobile phones. In *Proc. of ACM PhoneSense* (2011).
35. Wu, M., Wang, D., and Brown, G. J. A multipitch tracking algorithm for noisy speech. *IEEE Transactions on Speech and Audio Processing 11*, 3 (2003).
36. Xu, C., Firner, B., Moore, R. S., Zhang, Y., Trappe, W., Howard, R., Zhang, F., and An, N. Scpl: indoor device-free multi-subject counting and localization using radio signal strength. In *Proc. of ACM/IEEE IPSN* (2013).