Distributed Discussion Diarisation

Pascal Bissig
ETH Zurich
bissigp@tik.ee.ethz.ch

Klaus-Tycho Foerster
ETH Zurich / Microsoft Research
foklaus@ethz.ch

Simon Tanner
ETH Zurich
simon.tanner@tik.ee.ethz.ch

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

***Abstract*—In this paper we present *Disca*, a tool to analyze discussions in terms of which person is speaking at what time. We rely on a set of smartphones collaborating in detecting the most likely speaker at every given moment in real time. Each pair of smartphones observes a time difference of arrival pattern that is caused by the location of the different participants. The set of observations between all pairs of smartphones is then used to identify speakers on-line. To achieve this, clock differences and clock drifts between devices are estimated and compensated. Ultimately, participants are found by clustering time difference of arrival measurements which are unique for distinct speakers. We implement the system as an Android application and show that for more than 90% of time windows the correct speaker can be identified. To cope with heterogeneous hardware of Android smartphones, the computational burden is dynamically distributed among all participating smartphones according to their performance.**

I. Introduction

Face to face communication is a vital part of our everyday lives. Discussions occur at work or with friends and family. However, even though we spend a lot of time talking to other people, it is hard to obtain objective data about these discussions. The lack of facts, be it during business meetings or in informal situations, makes it hard to improve discussions. Also, we cannot present our peers with facts when criticizing or trying to improve a conversation. Subjective criticism can be perceived as being offensive rather than helpful. In a corporate environment, inefficient communication and a bad work climate directly translate to added cost. To reduce these effects, companies organize team building events or even hire counselors. Using *Disca* in business or personal meetings can help obtaining objective statistics about a given discussion.

Disca is a distributed smartphone app which can distinguish the participants of a discussion and collect data about who is talking at what time. This information can be used to identify behavior that is keeping the discussion from being productive. For example, there might be a person hogging the conversation by not leaving any room for others. Or there might be someone who continuously interrupts others. Telling the culprit is usually difficult since there is no evidence and hence, the constructive criticism may be ignored or interpreted as a personal attack. By supplying objective data of such behavior, conversations can be optimized in an objective way.

We collect this data using a set of smartphones that collaborate to identify the current speaker. Since most people carry a smartphone nowadays, *Disca* can be applied in most everyday situations easily. Each smartphone records the conversation and exchanges chunks of recorded audio with the others. For each smartphone pair, the delay between the recordings is estimated using cross correlation. This leads to a vector containing delays for each smartphone pair which is then used to identify a speaker. Our method compensates offsets in the sampling rates of different smartphones and runs in real time on off-the shelf smartphones. A Markov model is used to reduce the effect of noisy measurements. The computational burden is distributed among the participating smartphones to avoid very slow devices being overburdened. The results are visualized in real time and archived so previous conversations can be aggregated or compared. Also, the results gained from the Markov chain allow to analyze if there are cliques of participants communicating mostly with each other.

Disca performs all computations in real time without sending any audio recordings to the cloud. Instead, all computations are performed locally such that no personal data has to be shared to obtain the results. To our knowledge there are no speaker diarization systems that can run in a fully distributed setting. This is mostly due to clock inaccuracies that prohibit tracking time difference of arrival measurements. In *Disca*, we show how clock inaccuracies can be overcome by coarsely synchronizing the clocks via network as well as tracking clock drifts using the recorded audio directly.

II. Related Work

Business meetings have been in the spotlight for being inefficient and frustrating as shown in a study by Romano et al. [1]. The process of distinguishing different speakers is called speaker diarisation and is extensively discussed in literature. The systems can be largely divided in two classes.

The first category uses acoustic features like Mel Frequency Cepstral Coefficients (MFCC) [2] and others [3] generated from one recording to identify the active speaker. These systems are especially useful if only one recording is available such as during like radio broadcasts and phone calls. However, we have observed that MFCC features perform poorly for voices that are similar. MFCC features are very suitable for authentication tasks when the spoken words are always the same. Changing the content introduces uncertainty that greatly reduces the performance of these features.

Lu et al [4] recently discussed continuous audio sensing to identify nearby speakers could improve life-logging applications. They use a single microphone to determine if a certain speaker is talking at the time. Note that their approach requires training for each speaker that is to be classified whereas our method does not require any training data. Similarly, Xu et

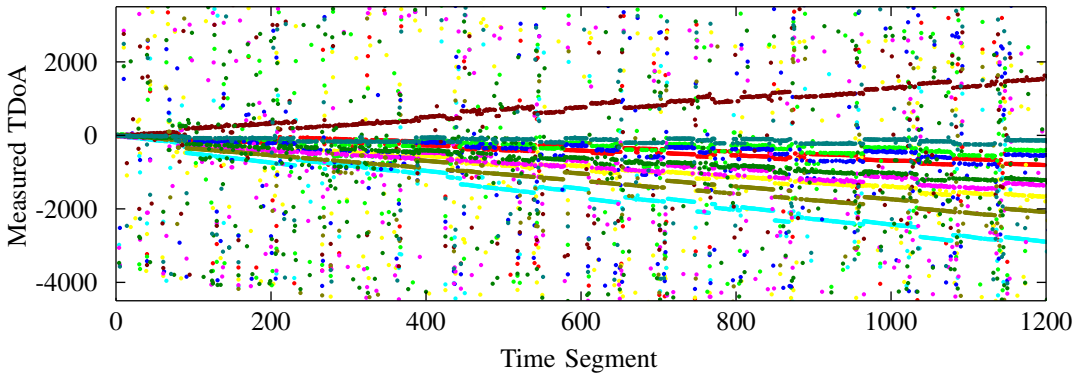


Fig. 1: Typical drift observed between all pairs of five different phones. Each time segment corresponds to 8192 samples and hence 1200 time segments correspond to 220 seconds. The clock drifts exceed the time difference of arrival measurements by far after a short period of time.

al. [5] showed that smartphone microphones can be used to count speakers in an unsupervised fashion.

The second class of systems takes advantage of multiple microphones. In contrast to methods relying on acoustic features, these methods generally require the microphones and speakers to remain approximately in the same location during a discussion, the speaker voices can be arbitrarily similar. Brandstein and Silverman [6] showed that microphone arrays can track active speakers. Similarly, Anguera et al. [7] use acoustic beamforming to enhance the signal from multiple distant microphones. However, the time difference of arrival (TDoA) data is not used to classify speakers. In their later paper [8], recordings from different microphones are compared to a reference and the timing data is used to infer active speakers. Note that they need reference microphone which can record each speaker well. Hence, all speakers have to be at a similar distance to the reference microphone. If this is not the case, the results will deteriorate since it will affect all the TDoA measurements from all other microphones. In addition to this, all the above methods are not robust against uneven sampling rates across different recording devices. We show that off-the-shelf smartphones are equipped with clocks that are prohibitively inaccurate for the above methods to work. More recently, Sur et al. [9] showed that smartphones can be accurately synchronized to perform beamforming. A central server is used to track clock drifts to achieve array gain for the microphones. Their speaker localization algorithms require the phones to be placed according to a given scheme. Also, a central server is required to achieve accurate synchronization. *Disca* does not require a central server or reference microphone and can accurately compensate for clock differences between recording devices.

Praviainen et al. [10] show how environmental sounds can be utilized to synchronize and localize off-the-shelf devices such as smartphones. Our system is similar because the recording setup of multiple smartphones is alike. Interestingly in [10], clock drifts are neither handled nor mentioned albeit in our experiments their impact on performance proved to be severe. Generally, the resulting sequence of clusters that best match the observations are post processed to reduce noise. For example, the Viterbi algorithm can be used to impose basic temporal properties of discussions as described by Anguera et al. [8].

III. Model

We aim to analyze discussions based on who was speaking at what time. To this end, a set of smartphones record the discussion. We assume that any two participants of the discussion have a unique set of distances to each microphone. If there are three microphones that are not arranged on a line, it is easy to see that, in a plane, there are no two locations with the same set of distances. Not every participant requires to provide a phone to obtain accurate results. The distances between the speaking participant and the microphones cause a propagation delay. If the locations of the microphones were known and their clocks would be synchronized, it would be possible to deduce the location of the speaker. Mostly because of the delay caused by the operating system which is not designed for such tasks, clock synchronization on such a high level of accuracy is infeasible on current smartphones.

We use the differences in propagation delays Δ_s to classify each speaker s. This difference is defined and may vary for each pair of smartphones. We assume speakers and smartphones to remain more or less in the same location throughout the discussion. In this case Δ_s is constant for all speakers s.

The time difference of arrival (TDoA) $d_k(i)$ observed in audio segment i for the k^{th} pair of smartphones is influenced by the difference in sampling rates of the two recording smartphones r_k. Also, clocks are not perfectly synchronized which leads to a constant offset c_k. The time difference of arrival observation $d_k(i)$ therefore relates to the difference in propagation delays $\Delta_{s,k}$ as follows:

$$d_k(i) = \Delta_{s,k} + i \cdot r_k + c_k + w \tag{1}$$

We account for Gaussian measurement noise with the term w. When n smartphones are used, the time differences of arrival $d_{i,k}$ are calculated for each of the $n(n-1)/2$ pairs of recordings

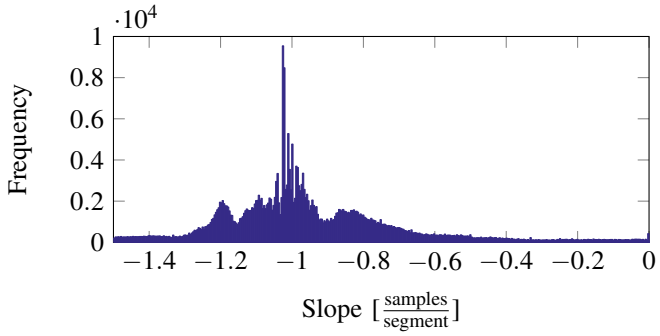


Fig. 2: Typical voting result for the most frequent slope aggregated in 500 time segments considering the past 500 segments.

in each audio segment. Combining all pairs of recordings we get the following:

$$D_i = \Delta_s + t \cdot R + C + W \tag{2}$$

In the following sections, we show how we estimate the difference in propagation delay for each observed audio segment. This information is then used to find each speaker's Δ_s. First, the smartphones are roughly synchronized such that audio segments from the individual smartphones that were recorded roughly at the same time can be compared. The time differences of arrival D_i are then calculated for each audio segment as described in the Time Difference of Arrival Section. The estimation of the difference in sampling rates R is explained in the Clock Sync Section. The resulting vectors of propagation delay differences Δ_s are then estimated by clustering and filtered as described in the Clustering Section.

Figure 1 shows the raw TDaA measurements performed for a set of five phones. Each pair of phones leads to a line that is sloped because of the clock differences r_k between the two participating devices. Also, the influence of the audio source Δ_s is apparent since the lines for each phone pair assume different levels as the speakers take turn.

A. Time Difference of Arrival

The calculation of the TDoAs requires the recordings of the different smartphones to be roughly synchronized. To find the corresponding position of one audio segment in an other recording, the time difference should be small. Otherwise the audio segment has to be compared to a long segment of the second recording.

Before starting with the recording, the phones exchange packets analogously to the Precision Time Protocol (PTP).

Using this synchronization method, the smartphones start recording at roughly the same time. The audio is then partitioned into segments of 8192 samples length that overlap the previous segment by 4096 samples. At a sample rate of 44.1 kHz one segment is 185.8 ms long, with one new segment being created every 92.9 ms. The corresponding position of this segment is then searched in the other recordings in a segment of 16384 samples.

The cross-correlation is used to find the time delay between the two signals x_1 and x_2.

$$R_{x_1,x_2}(n) = \mathscr{F}^{-1}(X_1(\omega)X_2^*(\omega)) \tag{3}$$

. where X_1 and X_2 are the Discrete Fourier Transforms of the signals x_1 and x_2. The TDoA is then the delay for which the cross-correlation R_{x_1,x_2} has the largest value. The Generalized Cross Correlation (GCC) [11] introduces weights in the frequency domain of the cross-correlation to make the calculation of the cross-correlation more robust against disturbing factors like noise and reverberations.

$$R^{GCC}_{x_1,x_2}(n) = \mathscr{F}^{-1}(X_1(\omega)X_2^*(\omega)\psi(\omega)) \tag{4}$$

One such weighting function that is used in conditions with reverberations is the Phase Transform (PHAT) [11]. It normalizes each frequency component and only uses the phase.

$$\psi_{PHAT}(\omega) = \frac{1}{\left|X_1(\omega)X_2^*(\omega)\right|} \tag{5}$$

This method is then called Generalized Cross Correlation with Phase Transform (GCC-PHAT). The TDoA d can the be calculated according to:

$$d = \arg\max_n R^{GCC}_{x_1,x_2}(n) \tag{6}$$

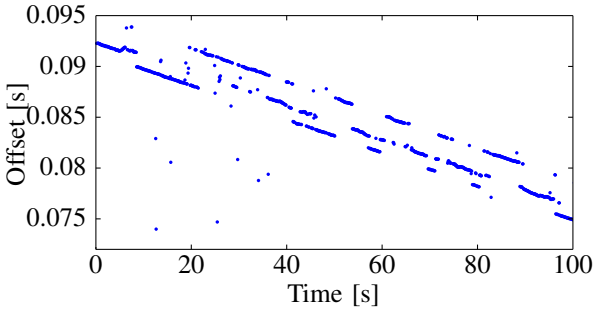
Figure 3a shows three different speakers taking turns in a discussion. The difference in the TDoA from time segments when one speaker is active to segments when another speaker is active are clearly visible. The slope is caused by the difference in the sampling rates of the two devices r_k.

B. Clock Drift

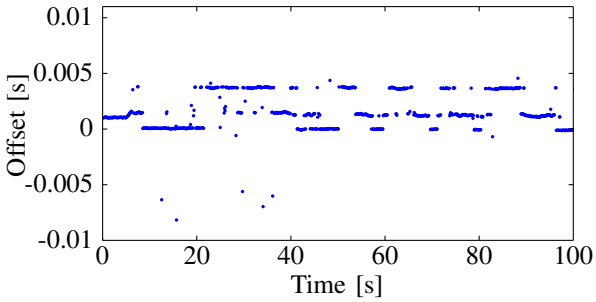
Experiments with different smartphones have shown that they do not record the audio at exactly 44.1 kHz. Figure 1 shows how quickly clock drifts aggregate to exceed the time difference of arrival values obtained rooms of regular size for meetings.

This is due to manufacturing tolerances and temperature differences. The differences measured are up to $\pm$ 15 samples per second. Without compensating these differences, two corresponding audio segments diverge and do not overlap anymore after a few minutes. Also, the TDoA vectors for any given speaker change over time if the difference in sampling rate is not compensated.

As a result of the difference in sampling rates, D_i lie on slopes as shown in Figure 1. The offset of these drifts is caused by different speakers being active at different times. To compute the actual propagation delay differences Δ_j that are used to detect the speakers, we need to compensate for the clock drift. Without knowledge of which speaker is active at what time, linear regression methods cannot be applied to estimate the slope. The presence of outliers makes least squares methods unsuitable. Instead, we compute the most likely slopes r using a voting scheme. For each newly D_i, we compute the slope to 500 D_j. The measurements D_j to which D_i is compared are cast from the last 2000. Each resulting slope casts a vote for the



(a) The raw offsets between both recordings. The slope in the graph is due to the difference in sampling rates r_k between the two participating devices.



(b) The offsets after compensating the difference in sampling rates r_k using our voting scheme. Each pair of recordings is drift compensated independently.

Fig. 3: One dimension of D_i from a recording of a discussion with 3 participants taking turns.

the actual slope. The binning then aggregates the most likely slope on-line by adding votes for each new TDoA measurement. The initial range that the binning spans is $-4\ldots4$ samples per time segment and contains 800 bins.

Figure 2 shows a typical binning showing a clear peak for a slope of roughly -1 samples per time segment containing 8192 samples. To more accurately estimate the slope, the range which the binning spans is reduced to accommodate the most likely slope values on-line. Previous measurements D_i are updates as the slope estimation becomes more accurate. Figure 3b shows the values of the slope compensated D_i ($D_i - t \cdot R$) and the corresponding to the raw D_i values in Figure 3a.

C. Clustering

After accounting for the clock differences, the differences in propagation delays Δ_s are the main unknown influences on the measurements from Equation 2. For each time window i we can compute l_i from the initial measurement D_i:

$$l_i := \Delta_s + W = D_i - t \cdot R - C \tag{7}$$

The values that l_i can assume are directly related to the time differences caused by different sound sources Δ_s and the measurement noise. Hence we do expect a user to cause values of l_i that are similar regardless of the frame number i. To find the actual set of Δ_s we cluster the results of the right side of Equation 7. The DBSCAN [12] algorithm clearly outperformed K-means clustering due to the large number of outliers that are present in the measurements. Running DBSCAN iteratively allows us to add new measurements at run-time.

By iteratively adding new data points, the density of noise points increases. This can lead to the merging of individual clusters which may represent different speakers. To avoid this problem, the number of data points is kept constant by removing the oldest data points. Clusters vanish when they have no data points left but the position of the previous clusters is stored. When a new cluster is created, it is compared to the position of the previous clusters and is connected to it if the positions are close. Therefore, speakers that were quiet for some time can be correctly detected when they start speaking again.

The measurements D_i often contain noisy components. Since the difference in propagation delay is short for all pairs of phones assuming that the recording area is limited, we can easily filter noisy measurements. After that, most Δ_s do not contain all components. Even so, the measurements should be clustered. To achieve this, the Partial Distance Strategy [13] is used to compute the distance between two data points using all components that exist in both data points. The distance between the vectors l_i and l_j each with N dimensions is calculated according to

$$d = \sqrt{N \frac{\sum_{k=1}^{N} (l_{i,k} - l_{j,k})^2 I_k^{i,j}}{\sum_{i=1}^{N} I_k^{i,j}}} \tag{8}$$

with $l_{i,k}$ being the k^{th} component of l_i and

$$I_k^{i,j} = \begin{cases} 1, & \text{if } k^{\text{th}} \text{ component is defined in } l_i \text{ and in } l_j \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

Additionally, the distance d is set to infinity when too few corresponding vector components are available after filtering. Since the geometry of the phones and the position of the speakers are not known, it is not possible to determine if these available components are sufficient to distinguish the different speakers. In the worst case clusters representing different speakers get merged. To avoid this problem, a high number *minPts* for the DBSCAN clustering is used and a penalty for measurements with only few components is introduced:

$$d' = d \frac{N}{\sum_{K=1}^{N} I_k^{i,j}} \tag{10}$$

D. Temporal Filtering With Markov Model

Time segments may be incorrectly classified as silence or as another speaker. These errors can occur because of short pauses or environment noise that caused the calculation of the TDoAs to give wrong results. Subsequently these time segments were associated with the wrong speaker in the clustering algorithm.

These errors can be corrected by assuming a structure for conversations that can be captured in a Markov model. For example, it is unlikely that speakers take turns 10 times per

second. The states in the model we use represent the active speaker and silence. It is assumed that only one speaker is active at the same time. The transitions between the states represent the probability of moving from one state to an other in one time segment. If a speaker is active in one segment, the same speaker will probably still be active in the next time segment 92.9 ms later. Therefore, the probability of staying in the same state is higher than the probability of changing to an other active speaker or to silence. For each state there are emission probabilities describing the probability of getting a certain observation when being in this state. The observations here are the different cluster assignments. The probability of observing the cluster assignment corresponding to the current state is highest while the probability of observing silence or another cluster is assignment is smaller. The Viterbi Algorithm is used to find sequence of states $x_1, ..., x_T$ of the Hidden Markov Model that matches the observations best.

E. Distributing the Workload

The smartphones used today vary in their processing power. Therefore, it is necessary to distribute the workload of processing the audio recording pairs to the participating phones such that all can complete their work in time. Since the step size of the processed audio segments is 4096 samples, one segment should be processed in 92.9 ms. On smartphones with multiple cores, multiple segments can be processed in parallel.

After starting the application, the audio recording pairs are distributed evenly to the participating smartphones through WiFi. Each pair is processed on one of the smartphones that is part of the pair. This helps to minimize the network bandwidth utilized by *Disca*. Also, each phone monitors the time required to process one segment. If the required time exceeds the available time of 92.9 ms, it requests its neighbors to take over the computation for their respective audio pair. So the workload allocation is handled in a fully distributed manner without chaning the network bandwidth requirements. After sending a neighbor a request to pass on the responsibility of processing the pair of recordings, the other smartphone accepts or refuses depending on the available processing time. If the transfer is rejected, we try to pass on one of the other pairs of recordings in which the overburdened smartphone is involved. We observed that even dated Android devices such as the Galaxy Nexus easily handle the computational burden. After calculating the TDoA for an audio pair, the smartphone transmits the calculated value to all other smartphones. When all measurements for one time segment are received, the clustering and classifying steps are executed on each smartphone.

IV. Evaluation

To evaluate our speaker detection system two setups have been used. Firstly, actual conversations with three speakers sitting around a desk have been recorded. In total, 4 different seating positions, rooms and combinations of people were recorded for a total of 20 minutes. The rooms were not chosen to be explicitly quiet and noise sources such as air conditioning or people moving and talking outside the open door were present. Each of these discussions was annotated manually.

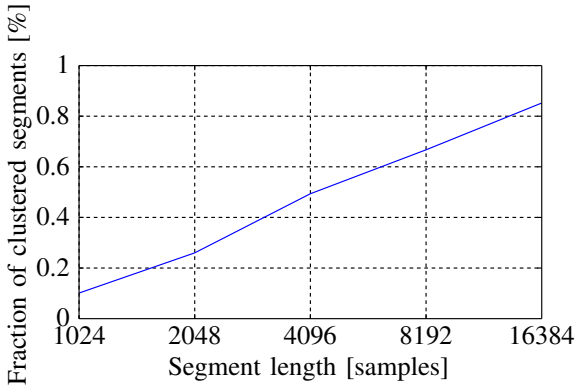


Fig. 4: Segments that could be clustered for segment lengths from 1024 to 16384 samples. With shorter segment length fewer segments could be assigned to a cluster.

To do a long term test, we used a 5.1 speaker system. We distributed an audio book such that it was played from one speaker at a time. The speaker was switched in a random pattern to simulate multiple people taking turns speaking. This setup, by design, provides an accurate ground truth about which speaker was active at which time. Like this, a total of 60 additional minutes of annotated data was obtained. All the experiments were recorded with five smartphones (Samsung Galaxy S3, Samsung Galaxy Nexus, Samsung Nexus S, HTC One M7).

A. Segment Length

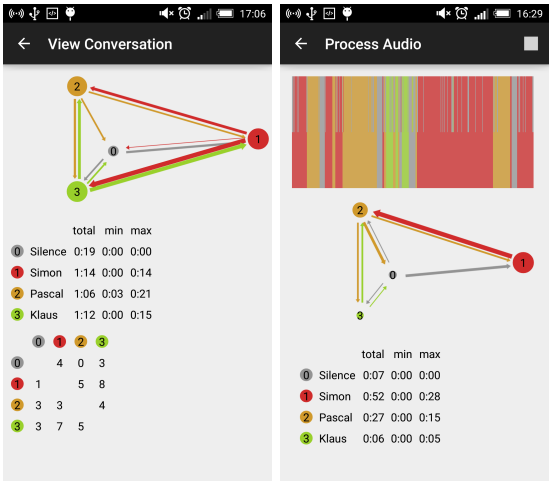
The length of the audio segment has a large impact on the reliability of the observed TDoAs $d_k(i)$. With smaller segment lengths, fewer TDoAs are correctly estimated. As a result, incorrect observations are removed in the filtering step and some are classified as noise. This leads to a poor clustering result with many time segments not assigned to any speaker. However, the processing power limits the length of the segments, larger segments require more processing power to compute correlation functions. Additionally, the segments should capture only one active speaker and also detect short pauses. For the remainder of the evaluation, a segment length of 8192 was used. This allowed us to run *Disca* on our test devices in real time.

Figure 4 shows for the different fragment lengths, which segments could be assigned to a cluster.

B. Distributed Speaker Diarisation performance

Naively comparing the ground truth annotation to the output of our diarisation algorithm leads to roughly 93% of time segments being correctly classified.

In the real discussion experiments, performance is slightly worse at 90%. Manual inspection showed that many misclassifications occur when the speaker changes. More explicitly, time segments that are within 0.2 seconds of a speaker change annotated in the ground truth are only in 58% of cases correct. Ignoring these time segments, 94% of the remaining time



(a) Overview of a previously recorded conversation.

(b) On-line visualization of the speaker activity.

Fig. 5: Selection of saved conversations and overview of those. Transitions between speakers are shown in the transition diagram and the number of transitions are displayed. For all speakers their time contributing to the conversation is listed.

segments to be correctly classified. Note that 8% of time segments lie within 0.2 seconds of a speaker change.

When annotating the recordings, it is in many cases unclear at which time exactly a speaker changes happen. In most cases there is a slight pause between the speakers and it is unclear to which speaker the pause should be assigned. In other cases, one speaker interrupts another creating a slight overlap. In the rarest cases, speakers change without either an audible pause or overlap.

In the audio book experiment, the performance is slightly higher at 96% of the segments being correctly identified. Neglecting errors within 0.2 seconds of a speaker leads to 98% of the segments being correctly classified. Note that, again, 8% of time segments lie within 0.2 seconds of a speaker change. The improved performance is mostly due to the more controlled sequence of speakers without long pauses or overlaps. Also, the ground truth data is not subjective and free of annotation errors due to the experimental setup.

To estimate the performance when less smartphones are contributing to the system, we randomly selected three of the available five recordings. The results were within 1% of the previously discussed results with five recordings.

V. Android Application

The implemented Android application is able to detect the active speakers in real time. When starting the detection, the participating persons can be selected. Speakers selected on one phone are automatically matched to the cluster which is closest to that phone. The number of speakers is not tied to the number of phones participating. Additional speakers are assigned a color which at any time can be matched with a name manually.

While detecting the active speakers, the application shows at the top of the screen the sequence of the last speakers. The upper half of the bar in Figure 5b shows the result from the clustering step and the lower half the detected speakers after filtering with the Markov model. The transition diagram is updated periodically and shows how often the transition between the different speakers and silence occurred. The area of the circles corresponds to the total time the person has spoken. When the detection is stopped, the results are saved. Figure 5a shows a the statistics available for a completed discussion. In addition to the information shown while processing, the number of transitions is also shown in text form.

VI. Conclusion

We have shown how a set of off-the-shelf smartphones can be used to distinguish active speakers in a conversation. We show that speaker diarization can be performed using multiple phones and software albeit the practical limitations of inaccurate clocks. The resulting system could also be used to perform beamforming to boost the audio quality for the active speaker.

References

[1] J. Romano, N.C. and J. Nunamaker, J.F., "Meeting analysis: findings from research and practice," in *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, 2001.

[2] S. Nakagawa, K. Asakawa, and L. Wang, "Speaker recognition by combining mfcc and phase information," *spectrum*, 2007.

[3] X. A. Miro, *Robust speaker diarization for meetings*. Universitat Politècnica de Catalunya, 2007.

[4] H. Lu, A. B. Brush, B. Priyantha, A. K. Karlson, and J. Liu, "Speakersense: energy efficient unobtrusive speaker identification on mobile phones," in *PerCom*, 2011.

[5] C. Xu, S. Li, G. Liu, Y. Zhang, E. Miluzzo, Y.-F. Chen, J. Li, and B. Firner, "Crowd++: unsupervised speaker count with smartphones," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013.

[6] M. S. Brandstein and H. F. Silverman, "A practical methodology for speech source localization with microphone arrays," *Computer Speech & Language*, 1997.

[7] X. Anguera, C. Wooters, B. Peskin, and M. Aguiló, "Robust speaker segmentation for meetings: The icsi-sri spring 2005 diarization system," in *Machine Learning for Multimodal Interaction*. Springer, 2005.

[8] X. Anguera, C. Wooters, and J. Hernando, "Acoustic beamforming for speaker diarization of meetings," *Audio, Speech, and Language Processing, IEEE Transactions on*, 2007.

[9] S. Sur, T. Wei, and X. Zhang, "Autodirective audio capturing through a synchronized smartphone array," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014.

[10] M. Parviainen, P. Pertila, and M. Hamalainen, "Self-localization of wireless acoustic sensors in meeting rooms," in *Hands-free Speech Communication and Microphone Arrays (HSCMA), 2014 4th Joint Workshop on*, 2014.

[11] M. Brandstein and H. Silverman, "A robust method for speech signal time-delay estimation in reverberant rooms," in *Acoustics, Speech, and Signal Processing, 1997 IEEE International Conference on*, 1997.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, 1996.

[13] A. Matyja and K. Siminski, "Comparison of algorithms for clustering incomplete data," *Foundations of Computing and Decision Sciences*, no. 2, 2014.