Multimedia Systems (2005)
Digital Object Identifier (DOI) 10.1007/s00530-004-0160-5

Unsupervised speaker segmentation and tracking in real-time audio content analysis*

Lie Lu, Hong-Jiang Zhang

Microsoft Research Asia, 5F Beijing Sigma Center, No. 49 Zhichun Road, Hai Dian District, Beijing, 100080, China
(e-mail: {llu, hjzhang}@microsoft.com)

Published online: 12 January 2005 –

Abstract. This paper addresses the problem of real-time speaker segmentation and speaker tracking in audio content analysis in which no prior knowledge of the number of speakers and the identities of speakers is available. Speaker segmentation is to detect the speaker change boundaries in a speech stream. It is performed by a two-step algorithm, which includes potential change detection and refinement. Speaker tracking is then performed based on the results of speaker segmentation by identifying the speaker of each segment. In our approach, incremental speaker model updating and segmental clustering is proposed, which makes the unsupervised speaker segmentation and tracking feasible in real-time processing. A Bayesian fusion method is also proposed to fuse multiple audio features to obtain a more reliable result, and different noise levels are utilized to compensate for background mismatch. Experiments show that the proposed algorithm can recall 89% of speaker change boundaries with 15% false alarms, and 76% of speakers can be unsupervised identified with 20% false alarms. Compared with previous works, the algorithm also has low computation complexity and can perform in 15% of real time with a very limited delay in analysis.

Keywords: Audio content analysis – Audio indexing – Speaker segmentation – Speaker change detection – Speaker tracking

1 Introduction

Speaker recognition, including both speaker identification and verification [1,7], has been widely researched in recent years. In these existing speaker recognition systems, it is supposed that the input speech belongs to one of the known speakers. However, in many applications, such as in a real-time conversation or news broadcasting, the speech stream is continuous and there is no information about the beginning and end of the speech segment of a speaker. Therefore, if we need to index speech streams based on speaker or to perform video content analysis based on audio track, it is necessary to find speaker change points first in such applications before the speaker can be identified. This procedure is called speaker segmentation, or speaker change detection. Furthermore, speaker tracking, which clusters a speech stream by speaker identities, can be performed based on the results of speaker segmentation. Speaker tracking is also essential in many applications, such as conference and meeting indexing [6,18], audio/video retrieval or browsing [24,25], speaker adaptation for speech recognition [12,21], and video content analysis [30].

Unlike the speaker identification or verification problem defined in most previous studies, we assume that there is no prior knowledge about the number and identities of speakers in the speaker segmentation and tracking process. If the speakers are registered a priori, traditional speaker identification algorithms can be employed for speaker segmentation and tracking, as in the work of [2]. However, in many cases, such as a continuous speech stream from live news broadcasting or a meeting, the prior knowledge of speaker identities and number of speakers is often not available or difficult to acquire. Even in well-structured news broadcasting, we cannot assume that the anchorpersons are always the same. Therefore, it is desirable to perform unsupervised speaker change detection and tracking algorithm in audio content analysis.

Several papers have reported work on unsupervised speaker identification or tracking using different algorithms in different applications. Sugiyama [3] studied a simpler case in which the number of speakers to be clustered was assumed known, and vector quantization (VQ) and the hidden Markov model (HMM) were used in the implementation. The algorithm proposed by Wilcox [4] was also based on HMM segmentation, where an agglomerative clustering method was used when the prior knowledge of speakers was unavailable. Another system [5,8] was proposed to separate controller speech and pilot speech with a Gaussian mixture model (GMM), in addition to speech and noise detection that were also considered in the framework. Speaker discrimination from telephone speeches was studied in [6] using HMM segmentation. However, in this system, the number of speakers was limited to two. Mori [12] addressed the problem of detecting speaker changes and speaker clustering without prior available knowledge. The speaker grouping information was

* Part of the work presented in this paper was published in the 10th ACM International Conference on Multimedia, 1–6 December 2002

used in speaker adaptation for speech recognition. Chen [13] also presented an approach to detecting changes in speaker identity, environmental conditions, and channel conditions using Bayesian information criteria. A segmentation accuracy of about 80% was reported. Couvreur [16] presented a first approach to building an automatic system for broadcasting news-speaker-based segmentations. The system was developed in the framework of the THISL project based on a chop-recluster method. Sonmez [17] and Bonastre [18] also reported their work on speaker tracking and detection systems.

Previous efforts to tackle the problem of unsupervised speaker segmentation and tracking consist of clustering audio segments into homogeneous clusters according to speaker identity, background conditions, or channel conditions. Most methods are based on VQ models, GMMs, or HMMs. A disadvantage of these models is that iterative operations are unavoidable in speaker modeling, which makes these algorithms very time consuming. Such approaches are more suitable in offline processing applications without prior knowledge of speakers. If they are used in applications in which real-time processing is required, offline supervised training of speaker models have to be processed in advance with prior knowledge. Here, "real-time" or "online" means that the results of speaker segmentation and tracking can be instantly obtained with the parsing of audio streams, and the audio stream parsing is performed once and only once. In our system, the intent is to segment and track speakers online or in real time, as well as *without* any prior knowledge such as speaker count or labeled training data. Thus, the approaches mentioned above cannot be applied in applications like ours.

In this paper, we present an effective algorithm on real-time speaker change detection and speaker tracking in a more general unsupervised case, where both speaker identity and speaker number are assumed to be unknown.

1.1 Issues and solutions

To perform unsupervised speaker segmentation and speaker tracking for real-time audio content analysis, a number of issues need to be addressed. In this section, we discuss these issues and present our solutions.

1. Because there is no prior knowledge of speakers in our applications, it is impossible to obtain labeled speaker data and thus an accurate speaker model a priori. In our implementation, the speaker data are obtained from speech stream gradually, and the speaker model is established incrementally by online updating. In such cases, the initial speaker model cannot be estimated accurately with limited available data. That means we cannot obtain an accurate GMM model with full components. Therefore, in our approach, GMM with one mixture component is initially used to estimate the initial speaker model when the available data are limited. Then, both the number of mixture components and the parameters of each component of GMM are incrementally updated, while the available data increase. As the process accumulates more speaker data, the speaker model will become more accurate.

2. Because of the real-time processing requirement, it is not affordable to use complex audio features and complex clustering methods. For example, when training a GMM model, a traditional expectation-maximization (EM) algorithm is not feasible since its time cost depends largely on the iterations and training samples so that it usually could not meet the real-time requirement. Although online-EM may be adopted in updating an existed GMM model in real time, it can only update the parameter of each component, but not the number of mixture components, which is necessary in our applications, as mentioned above. Therefore, it is not suitable to employ online-EM in such a system. In our implementation, incremental quasi-GMM, which utilizes segmental clustering, is proposed to solve this problem. This method is less time consuming (15% of real time as experiments indicate) and makes it possible to incrementally update both the number of mixture components and the parameters of each component of the speaker models. Although it is not as accurate as traditional EM algorithm, it is capable of capturing the main components of speaker models.

3. Also with the real-time requirement, the speaker identity is expected to be estimated with little delay after the speaker changes are detected. This is challenging since there is little data to model the new speaker when a speaker change occurs. Since it is difficult to obtain an accurate model for new speakers, we perform the speaker tracking in several different positions. We estimate the new speaker identity when speaker change is detected and rectify the previous estimate result in the later positions, where more speaker data are available.

4. Various audio features can be used for speaker segmentation and tracking. However, it is difficult to select the best feature. In our implementation, various features such as mel-frequency cepstral coefficient (MFCC), line spectrum pairs (LSP), and pitch are employed, since different features can complement each other in different contexts. A Bayesian fusion framework is used on these features to get a more robust result.

5. The environment in an audio context is so complex that channel or environment/background mismatch remains a challenging problem. In our application, traditional cepstral mean subtraction (CMS) is employed to compensate for channel mismatch. However, it is not sufficient to compensate for background mismatch, especially for additive noise. To compensate for environment mismatch more effectively, the noise level is detected in each speaker segment and used in the final decision.

1.2 System overview

The flow diagram of our proposed real-time unsupervised speaker segmentation and tracking algorithm is illustrated in Fig. 1. It is designed to meet the requirements of unsupervised real-time processing and to address the issues mentioned in the previous subsection. It is mainly composed of four modules: front-end processing module, segmentation module, clustering and speaker model updating module, and speaker tracking module. It is assumed that the input audio stream is speech only; nonspeech segments of the input audio have been filtered. In our system, nonspeech audio segment filtering is performed by an audio segmentation and classification algorithm similar to the one presented in [15,23,31].

In the front-end process, the input speech stream is divided into 3 s subsegments with 2.5 s overlapping. These subseg-

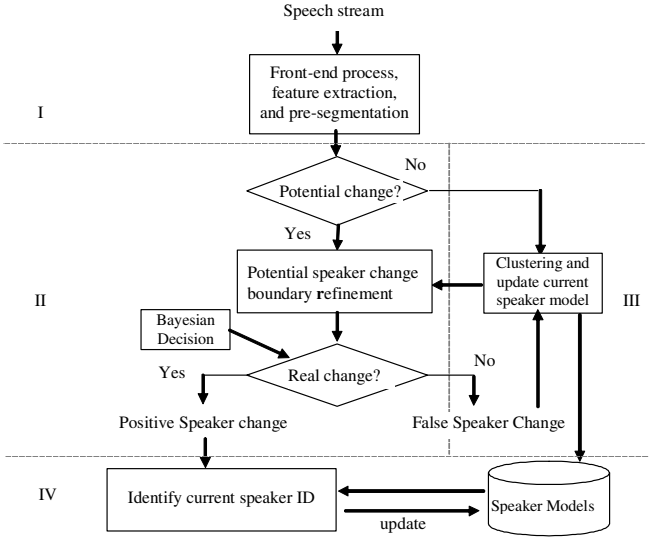

Fig. 1. A simple flow diagram for speaker change detection and speaker tracking, composed of four main modules: I. Frontend process. II. Speaker segmentation. III. Clustering and speaker model updating. IV. Speaker tracking

ments are used as the basic unit and preprocessed by removing silence and unvoiced frames.

Then, speaker change detection is performed by a "coarse to refine" process, where potential speaker change is first detected and then confirmed. This process gives a coarse estimation of when the available data are limited and then does a confirmation when available data increase, which also ensures that more accurate segmentation results can be obtained under the requirements of being both unsupervised and real-time processing.

If no potential change is detected between two subsegments, or a potential change is confirmed as a false alarm, the current speaker model is updated by incorporating data of the current subsegment. In our approach, incremental quasi-GMM with segmental clustering is utilized for speaker model updating.

Finally, when a speaker change is confirmed, the algorithm searches the speaker model database to identify the newly appeared speaker. The process is like a classification process with rejection. If the speaker could not be found in the database, it is registered as a new speaker and added to the database; otherwise, the detected speaker model in the database is updated by new speaker data.

In this way, the speaker model and speaker database are gradually updated and increased. Such a design meets the requirement of being both unsupervised and real-time processing.

The rest of the paper is organized as follows. Section 2 presents the selected features and distance metrics, it also presents the method of feature fusion. A detailed description on segmentation and clustering scheme is presented in Sect. 3. Section 4 describes the speaker tracking scheme. In Sect. 5, experiments and the evaluations of the proposed algorithms are given.

2 Feature selection, distance measure, and feature fusion

In this section, we discuss the problem of feature selection and distance measurement. In previous research, many features were proposed for speaker recognition systems. The most widely used features are LPC [3,7], MFCC [20], and LSP [1]. Other features are also used in some studies [22]. Meanwhile, CMS or RASTA processing [7,9,10] is performed on the feature vectors to remove the channel convolutional effects.

Different features show different performances in different speaker recognition systems. It is difficult to determine which feature is the best for all purposes. However, different features can complement each other in different contexts. With this in mind, we fuse various features to improve the performance of our system.

Our approach to feature fusion is based on feature discrimination power analysis. We first measure the dissimilarity of different feature sets between two speaker models. The dissimilarities are then Bayesian fused to get a more reliable result.

2.1 Acoustic feature selection

Line spectrum pairs (LSP) and mel-frequency cepstrum coefficient (MFCC) are commonly used and have proven their effectiveness in speaker recognition [1,20]. Therefore, both are used in our speaker segmentation and tracking system. In general, MFCC and LSP have a similar overall performance on speaker recognition. But they perform differently in some specials cases or contexts. In other words, these two features can complement each other to improve the performance of speaker segmentation and tracking.

Besides these two features, pitch information is also useful. Pitch information is a fast and effective discriminator between male and female speakers. Thus, it is also considered in our system and used as an assistant feature. These three features, LSP, MFCC, and pitch, will ultimately be Bayesian fused to get a more reliable result in our algorithm.

Moreover, in an audio stream, the environment and channel are variable and very complex to estimate. They affect the performance dramatically. Compensating for the effect of the channel or environment mismatch remains a difficult issue in speaker recognition research. Cepstral mean subtraction (CMS) is used in our algorithm. However, CMS alone is not sufficient, as proved by many researches. Noise level, which represents average background energy, is also estimated in our system to help compensate for these effects. In this paper, we also treat the music background as noise.

In general, speech is always composed of alternating voiced sound and unvoiced sound at the syllable rate. For the unvoiced sound, the energy is so small that it is always masked or contaminated by background noise. At the same time, the pause between voiced speech segments is also filled with background sound. Thus the energy of unvoiced/pause sound is similar to background sound energy. Based on these facts, an algorithm following [28] is used for adaptive background noise level detection. In our implementation, we estimate a noise level every 6 s to track the variation of background.

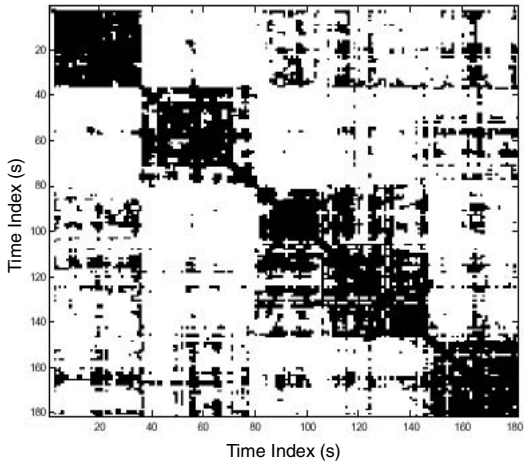

Fig. 2. A self-similarity matrix using LSP divergence shape

2.2 Dissimilarity measure

Kullback-Leibler (K-L) divergence [1] is used to measure the dissimilarity between each two speech subsegments, assuming that each subsegment is modeled as a Gaussian, as follows:

$$D = \frac{1}{2} tr[(C_i - C_j)(C_j^{-1} - C_i^{-1})] + \frac{1}{2} tr[(C_j^{-1} + C_i^{-1})(u_i - u_j)(u_i - u_j)^T], \quad (1)$$

where C_i and C_j are the estimated covariance matrixes of the ith speech subsegment and jth subsegment, respectively, and u_i and u_j are the corresponding estimated mean vectors.

The divergence is composed of two parts. The first part is determined by the covariance of two subsegments, and the second is determined by the covariance and mean. Since the mean is easily biased due to different environmental conditions, we will not consider the second part. Thus only the first part is used to represent the dissimilarity, based on [1]. It is also similar to the *CMS* method of compensating for the convolutional effects of channel. The final distance is called divergence shape, defined by

$$D = \frac{1}{2} tr[(C_i - C_j)(C_j^{-1} - C_i^{-1})]. \quad (2)$$

In general, if two speech clips are spoken by the same speaker, the dissimilarity between them would be small; otherwise, the dissimilarity would be large. Thus a simple criterion is: if the dissimilarity between two speech segments is larger than a given threshold, these two segments could be considered as being spoken by different speakers.

To show the effectiveness of the selected feature and distance measure, Fig. 2 illustrates a self-similarity matrix for a 180 s-long speech segment. The dissimilarity between any two subsegments is calculated. One threshold is used to transform the distance to a binary value (0, 1) based on the above simple criterion. Value 0 is represented by black pixels, while value 1 is represented by white pixels. The figure is clearly symmetric, and four different speakers exist in this speech segment. The self-similarity matrix obtained by using MFCC also shows similar characteristics.

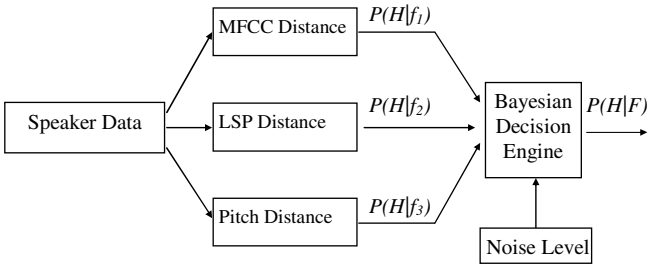

Fig. 3. A Bayesian fusion model

2.3 Feature fusion

Our approach to feature fusion is based on feature discrimination power analysis. We first measure the distance of different feature sets between two speaker models. The distance of each feature set gives the probability of the hypotheses and is then fused by a Bayesian decision engine to get a more reliable final decision, as illustrated in Fig. 3.

Since the features are extracted to represent different facets of a speaker, these features f_i $(i = 1, \ldots, N)$ are assumed independent of each other for simplicity in our implementation (although it is possible for them to be more or less correlated). Based on Bayesian inference [26,27], the fusion can be given:

$$P(H|F) = P(H)^{1-N} \prod_{i=1}^{N} P(H|f_i) \ldots = P(H) \prod_{i=1}^{N} P(f_i|H) \Big/ \prod_{i=1}^{N} P(f_i). \quad (3)$$

Thus, a basic hypothesis test can be employed to get the final decision:

$$\lambda = \frac{P(H_0|F)}{P(H_1|F)} \begin{cases} \geq \lambda_0 & accept\ H_0 \\ < \lambda_0 & accept\ H_1 \end{cases}, \quad (4)$$

where H_0 andH_1 are two opposite assumptions, λ is the likelihood ratio, and λ_0 is a threshold. In the experiments, various λ_0 are tested to obtain the relation curve between precision and false alarms.

The prior probability for each hypothesis can be estimated from the training data. Meanwhile, each $P(f_i \mid H)$ can also be easily acquired from the feature distance distribution at each hypothesis. For example, in speaker segmentation, the LSP distance distribution at speaker change boundary or non-boundary is illustrated in Fig. 4. In the implementation, the distribution is modeled as a single Gaussian for each speech segment:

$$P(f_i|H) = A_i \exp\{-\frac{(f_i - \mu_i)^2}{\sigma_i^2}\}, \quad (5)$$

where μ_i is the mean, σ_i is the variance, and A_i is used for normalization.

In general, the distance distribution is affected by the background noise. It is usually noticed that the distance is relatively larger in a noisy environment. That is, μ_i and σ_i depend on noise level. In our implementation, all speech data are divided into two noise level groups, which represent clear speech and

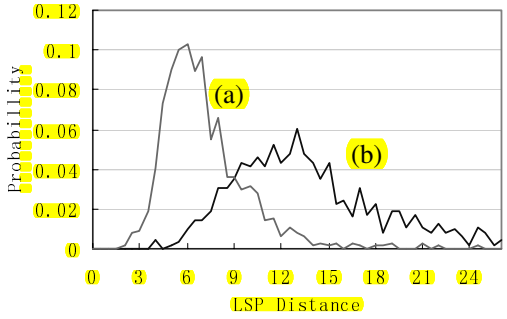

Fig. 4. LSP distance distribution at **a** nonspeaker boundary and **b** speaker boundary

noisy speech, assuming they have the same prior probability in the training data. In the fusion model, noise level is used to select the distribution parameters. This method processes the speaker data in similar environments and can help compensate for the effect of environment mismatch.

3 Speaker segmentation

The purpose of speaker segmentation is to detect the speaker change points in the speech stream in real time, as shown in Fig. 5. It is a "coarse to refine" process, where potential speaker changes are first detected when the available data are limited and then confirmed when available data increase. In the coarse step, an initial Gaussian model is estimated for each subsegment, and then the *LSP/MFCC/pitch* divergence distance between every consecutive two models is examined. A potential speaker change boundary is detected if the distance is above a given threshold. Otherwise, the data of the current subsegment are incorporated into previous subsegments to estimate a more accurate model of the current speaker. If a potential speaker change boundary is detected, Bayesian fusion, mentioned above, is used to confirm if it is really a speaker change boundary.

3.1 Potential speaker boundary detection

At this "coarse" step, an initial speaker model is estimated from each subsegment once it comes, and the dissimilarity between each two neighboring models is calculated at each time slot, as shown in Fig. 5a. In our implementation, only the LSP divergence distance is used to detect potential speaker change, since it is accurate enough and can recall more than 95% of true boundaries, while adding MFCC and pitch has little benefit.

A potential speaker change boundary is detected between the ith and the $(i+1)$th subsegment if the following conditions are satisfied:

$$\begin{aligned} &D(i,i+1) > D(i+1,i+2)\,, \\ &D(i,i+1) > D(i-1,i)\,, \\ &D(i,i+1) > Th_i\,, \end{aligned} \tag{6}$$

where $D(i,j)$ is the distance between the ith subsegment and the jth subsegment and Th_i is a threshold.

The first two conditions guarantee that a local peak exists, and the last condition prevents very low peaks from being detected. Reasonable results can be achieved by using this simple criterion. However, the threshold is difficult to set in advance. The threshold setting is affected by many factors, such as insufficient estimate data and different environmental conditions. For example, the distance increases if the speech is in a noisy environment. Accordingly, the threshold should increase in a noisy environment. To obtain the optimal result, an automatic threshold setting method is proposed as follows:

$$Th_i = \alpha \cdot \frac{1}{N} \sum_{n=0}^{N} D(i-n-1, i-n)\,, \tag{7}$$

where N is the number of previous distances used for predicting the threshold and α is a coefficient used as an amplifier. That is, the threshold is automatically set according to the previous N successive distances, and thus it catches and adapts to the context variations. In this step, false alarms are preferred to missing cases, since the false alarms can be rectified in the refinement processing, while missing boundaries cannot restored. In order to avoid too many missing cases, α is set as a small number. We have tested many selections of α (from 0 to 2) in our experiments, and we choose 1.2 in our algorithm to recall more than 95% of true speaker boundaries with relatively fewer false alarms, as Fig. 8 shows. The threshold determined in this way works well in different conditions. However, false detections (about 60%) still exist due to the insufficient data in estimating the speaker model from only one short speech subsegment.

To solve this problem, we should use as much data as possible to update the speaker model. A more accurate refinement method is proposed to refine the above results.

3.2 Incremental speaker model updating

In order to collect more data to estimate a speaker model more accurately, we utilize the results of potential speaker boundary detection. If no potential speaker boundary is detected, the system infers that the current coming subsegment is from the same speaker as the previous one. Thus, we update the current speaker model using these available new data, as illustrated in Fig. 5b. In this step, the speaker model is comprised of three features: LSP, MFCC, and pitch. Supposing each feature satisfies Gaussian distribution, the speaker model for the ith subsegment can be represented as $\{G_F(u_i, C_i)\}$, where F represents a different feature. In the remainder of the paper, we will omit the subscript F for simplicity and express the speaker model as $G(u, C)$, since each feature can be processed in the same way.

In our approach, GMM-32 is found sufficient to model a speaker, as indicated in some other works [1,18]. (More generally, GMM-s represents a Gaussian mixture model with s mixture components.) The model is established progressively in the following manner. Initially, there is no sufficient speaker data to accurately estimate a GMM-32 model; thus, GMM-1 is estimated. When more speaker data are available, the model gradually grows up to GMM-2, GMM-3, ..., and finally GMM-32. When the GMM-32 is reached, the updating of the speaker model is terminated.

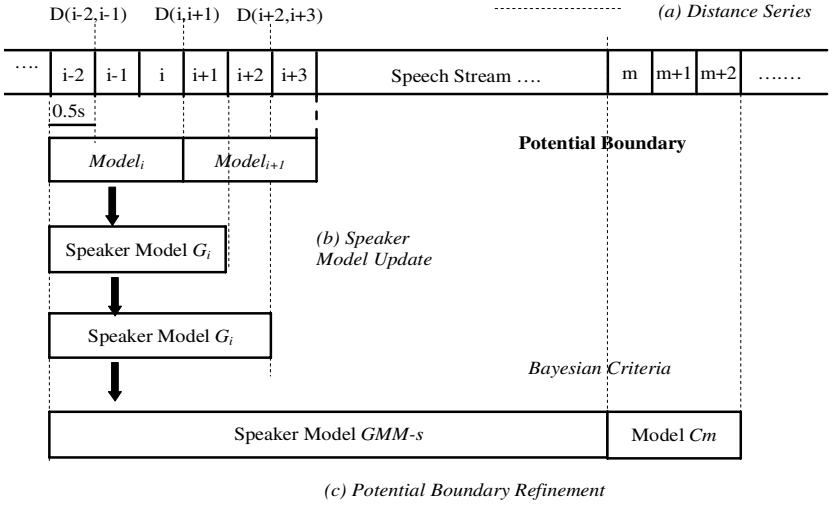

Fig. 5. A step-by-step illustration of the speaker change detection algorithm

A conventional EM algorithm is usually used to estimate the GMMs. However, the EM algorithm employs recursive process, so that it is usually time consuming and does not meet the real-time requirement. An online-EM algorithm may update a GMM in real time, but it cannot update the number of mixture components. Therefore, we introduce an alternative method, incremental quasi-GMM with segmental clustering, to update the number of mixture components. Although it may neglect low weighted components in a GMM, it is still capable of capturing the most important components in a GMM. Furthermore, the real-time requirement is met due to the computational simplicity of the approach. Through our empirical experiments it could achieve reasonable accuracy.

Suppose that the current speaker model G_i has been obtained from the previous $(M-1)$ subsegments and is represented by $G(u, C)$; the Mth speech subsegment, whose model is represented by $G(u_m, C_m)$, is also from the same speaker, without a potential speaker change point. Thus, the speaker model G_i can be updated using the feature data of the Mth subsegment by the following method:

$$\mu' = \frac{N}{N+N_m}\mu + \frac{N_m}{N+N_m}\mu_m\,, \tag{8}$$

$$C' = \frac{N}{N+N_m}C + \frac{N_m}{N+N_m}C_m + \frac{N\cdot N_m}{(N+N_m)^2}(\mu-\mu_m)(\mu-\mu_m)^T\,, \tag{9}$$

$$N' = N + N_m\,, \tag{10}$$

where N and N_m are the number of feature vectors used to estimate G_i and $G(u_m, C_m)$, respectively.

The third part of Eq. 9 is determined by means that are easily biased by environmental conditions. Thus, in practice we ignore the mean part of Eq. 9 to compensate for the effect of different environmental conditions or transmission channels. Then Eq. 9 is simplified as

$$C' = \frac{N}{N+N_m}C + \frac{N_m}{N+N_m}C_m\,. \tag{11}$$

The above procedure is looped per subsegment on timeline till the dissimilarity between the speaker models before and after updating is small enough or a potential speaker change point is met. The dissimilarity is also measured by the divergence shape distance. When the dissimilarity is sufficiently small, it is assumed that the current Gaussian model is estimated accurately, i.e, it is not necessary to continue updating G_i. The next Gaussian model, G_{i+1}, is initiated with current subsegment and updated with the new available data using the same method.

For one speaker, several Gaussian models will be estimated by the above method, which is called segmental clustering since each Gaussian component is obtained from one speech segment. Combining these Gaussian models would form a quasi-Gaussian mixture model. The weight of each Gaussian model is set by its corresponding number of training data, as $w_i = N_i/N$, where N_i is the number of feature vectors used to estimate G_i, and $N = \sum_{i=1}^{S} N_i$ is the total number of feature vectors.

3.3 *Speaker change boundary refinement*

Often there are false positives in the potential speaker change boundary obtained with the algorithms described above. To remove false boundaries, a refinement algorithm is applied to update the boundary hypotheses, based on the dissimilarity between the current subsegment and the speaker model obtained from previous subsegments before the current potential boundary, as illustrated in Fig. 5c.

Suppose at the potential speaker boundary the model of the previous speaker is GMM-s, in which each Gaussian model is $G(u_i, C_i)$ (i = 1,... s) and the model of the current segment is $G(u_m, C_m)$; then the distance between them is roughly estimated as the weighted sum of the distance of $G(u_m, C_m)$ and each $G(u_i, C_i)$:

$$D = \sum_{i=1}^{S} w_i \cdot D(C_i, C_m)\,. \tag{12}$$

The LSP distance, MFCC distance, and pitch distance are calculated, respectively, according to the above method. Then,

these features are fused and given as input to a Bayesian decision engine, as shown in Fig. 3. The basic hypothesis test used to refine the potential speaker change point is:

H_0: *It is a true speaker change boundary*
H_1: *It is not a true speaker change boundary*

The optimum test to decide between these two hypotheses is a likelihood ratio test given by

$$\lambda = \frac{P(H_0|F)}{P(H_1|F)} \begin{cases} \geq \lambda_0 & accept\ H_0 \\ < \lambda_0 & accept\ H_1 \end{cases}. \tag{13}$$

If a potential candidate is not a real speaker change boundary, the current speaker data are still used to update the speaker model based on the method described in Sect. 3.2.

4 Speaker tracking

When a speaker change boundary is confirmed, the next step is to identify the new speaker. The new speaker may or may not have registered in the speaker model database, which means this step is like a speaker identification and rejection process.

To identify the new speaker, the current subsegment is compared with all the existing speaker models in the database to find which model is most similar to the current subsegment. The most similar model is the most possible speaker of the current subsegment.

The dissimilarity between the verifying speaker model and the current subsegment is set as the weighted distance sum of k nearest Gaussian components in the speaker model, but not the weighted sum of all components as given in Eq. 12:

$$D' = \sum_{i \in N(k)} w_i \cdot D(C_i, C_m), \tag{14.1}$$

and $N(k)$ is a set of k nearest Gaussian components to C_m in the verifying speaker model, i.e.,

$$N(k) = \{i | D(C_i, C_m) < D_{kNN}\}, \tag{14.2}$$

where D_{kNN} is the $(k+1)$th smallest one in the distance series $D(C_i, C_m)), i = 1, \ldots, s$, assuming there are s components in the verifying speaker model.

We design the distance in this way since the data of a speaker model may be from different environments or channels and the weighted sum of all components may incorporate data from various channels, thus introducing noises and reducing the identification performance. In implementation, the number k is adaptively chosen based on the noise level (see Fig. 11 below). That is, only the components with a noise level similar to that of the current subsegment are used in the distance computation. It reduces the effect introduced by the environment mismatch. By contrast, in the module of speaker change boundary refinement, the speaker model is estimated from a continuous speech segment between two contiguous speaker change points. We can assume that there is no environment/channel change during this segment so that the weighted sum of all components can be used.

Then, for each speaker model, we will have a hypothesis test:

H_{i0}: *The current subsegment is speaker-i*
H_{i1}: *The current subsegment is not speaker-i*

The likelihood ratio for speaker-i can be given by

$$\lambda_i = \frac{P(H_{i0}|F)}{P(H_{i1}|F)}, \tag{15}$$

where the probability $P(H_i|F)$ can be estimated from distribution of the distance between intraspeakers or interspeakers, similar to the mapping method mentioned in Sect. 2.3.

The most possible speaker models could be found according to the biggest likelihood ratio. If the largest likelihood ratio is larger than a threshold, the speaker of the current segment is identified and the speaker model is updated with the new data; otherwise, the current segment is considered from a new speaker, which is then registered in the database. In this way, we can determine the identity of the current speaker.

Suppose up to now there are K speakers registered in the speaker model database; the concrete expression to identify the speaker of the current segment is as follows:

$$ID = \begin{cases} \arg\max_i \lambda_i & if\ \max \lambda_i \geq \lambda_0 \\ K+1 & if\ \max \lambda_i < \lambda_0 \end{cases}, \tag{16}$$

where $1 \leq i \leq K$ and $K+1$ represents a new speaker identity. It is similar to a classification process with rejection. Suppose that each speaker appears with equal probability; λ_0 can be set at 1. The threshold could also be set experimentally according to application context.

In fact, it is difficult to identify the speaker of the current subsegment immediately after the speaker change point is detected. This is due to the fact that our first segment is only 3 s long, which is not sufficient to properly classify a speaker. In our implementation, we try three ways of making identification decisions, corresponding to three different amounts of data used in making a decision.

1. Use only the data of the current subsegment. This allows one to perform identification immediately after the boundary is detected.
2. Perform identification until the next potential speaker change point is detected. In other words, in making an identification, we use all previous subsegments until the last speaker change is confirmed.
3. Perform identification until the next speaker change boundary is confirmed. That is, the information used in the next refinement is also used in making the decision, compared to case 2.

The least data are available in case 1, and the most data are available in case 3. We first estimate the new speaker identity by 1 and then rectify the previous estimation in later decisions by 2 and 3, where more speaker data are available. The experiments presented in the next section prove that case 3 achieves the highest performance. However, case 3 also introduces the longest delay in decision making; thus it may not be desirable in some applications with a strict real-time requirement.

5 Experiments

In this section, we present the evaluation results of the proposed real-time speaker change detection and speaker tracking

algorithms. The previous approaches are suitable for applications that either allow offline processing without any prior knowledge or real-time processing with supervised offline training, where the speaker models can be trained offline a priori with prior knowledge. However, these approaches cannot be applied to applications such as the one we address that require unsupervised training with real-time processing. Therefore, it is hard to compare the proposed algorithm quantitatively with existing approaches. In general, our algorithm is suitable for various applications on real-time audio content analysis. In this section, we evaluate our algorithm on a broadcasting news analysis.

In our experimentation, the input speech stream is downsampled into a uniform format: 8 KHz, 16 bits, monochannel, and then divided into 3 s subsegments with 2.5 s overlapping. That is, the basic processing unit is 3 s, and the temporal resolution of the segmentation is 0.5 s.

5.1 Database information

The evaluations of the proposed speaker change detection and speaker tracking algorithms were performed on the Hub-4 1997 English Broadcast News Speech Database. The database is composed of about 97 h of news broadcasting from different radio stations such as CNN, ABC, CRI, and C-SPAN. In our application, we only used the news broadcasting from *ABC World News Tonight* and *CNN Headline News*, which is a total of about 10 h. Half of our data was selected randomly for training and the other half was for testing. In testing the database, each speech file is about 30 min long, and there are about 30 speakers and about 60–80 speaker changes in each file.

Though this database was originally designed for spoken document retrieval, it is also suitable for our intended application: speaker segmentation and tracking for news broadcasting. The ground truth is obtained from its accompanying transcripts.

As we mentioned in previous sections, we use 3 s subsegments as our basic identification unit. This unit size was determined from the statistics of our experiments. This size is critical since if it is too short, there will be insufficient data to estimate an accurate speaker model; otherwise, if it is too long, two speakers' speech may intervene, resulting in inaccurate speaker model estimation, and it will also introduce more delay.

Figure 6 shows a histogram for the length of speaker segment in the training database. It is seen that about 5% of speaker segments are less than or equal to 2 s, and 10% are less than 3 s. We tested the performance with a subsegment of 2 s or 3 s; it was observed that the performance decreased dramatically when the subsegment was 2 s. Hence, we selected 3 s as a subsegment unit size. It implies that for those speaker segments that are less than 3 s the segmentation and tracking results are not so reliable.

5.2 Speaker segmentation

Recall and precision is used to evaluate the performance of the proposed speaker segmentation algorithm. As mentioned above, since the smallest unit used to cluster a speaker model is 3 s, our algorithm is not very good at detecting the speaker change point if the speaker segment is very short. The recall for short segments (<3 s) is only 37.6%. For the long segments (>3 s), the performance is much higher. The following reported results are all based on the long segments.

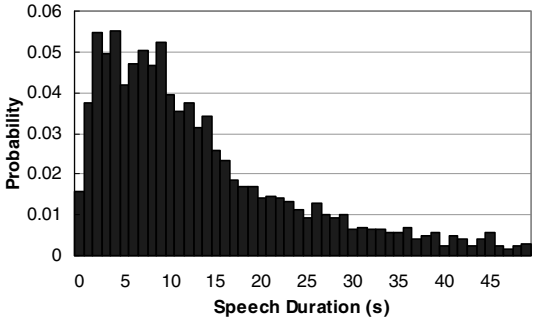

Fig. 6. The histogram for the length of speaker segment

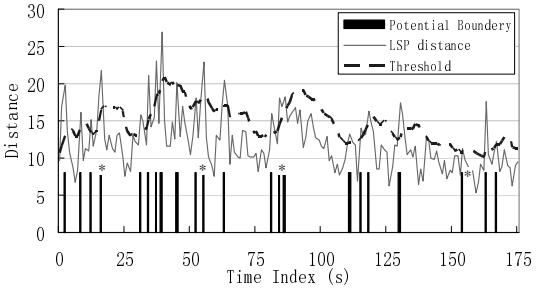

Fig. 7. Example of speaker change detection algorithm

Figure 7 shows an intuitive example of speaker change detection on 176 s-long speech. The speech clip contains four speaker change boundaries, which are 17 s, 52 s, 86 s, 154 s, respectively. Figure 7 shows the initial LSP distance between every two speech subsegments, the adaptive threshold, and the potential speaker change boundaries. The real boundaries are also marked with an asterisk.

To determine the adaptive threshold (Eq. 7), we should choose the amplifier coefficient α. Figure 8 shows the recall-false curve of potential speaker change detection with various selections of α marked beside a solid square, where false = 1 – precision. It can be seen that both recall and false decrease with an increase in α. In order to recall more than 95% of true speaker boundaries with relatively fewer false alarms, α is set at 1.2 in our implementation. This means that LSP is sufficient for potential speaker change detection since it has achieved a 95% recall rate. We also tested the performance adding MFCC and pitch, but little improvement was obtained. Therefore, only LSP is used in the potential speaker change detection for simplicity.

However, many false detections (about 60%) are still present in the potential boundaries. Bayesian fusion decision is performed on these potential speaker change points to remove the false ones, when more data are available. Figure 9 shows the corresponding recall–false curve achieved by the refinement process, with a few different thresholds in using Bayesian decision. Figure 9 also compares the performance among different feature fusions. It can be seen that the performances of individual MFCC and LSP are similar, while pitch

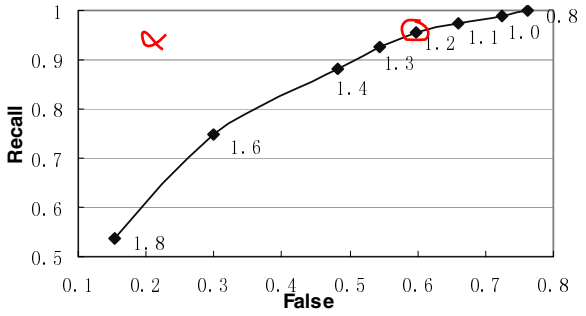

Fig. 8. Recall–false curve of potential segmentation with different selections of amplifier coefficient

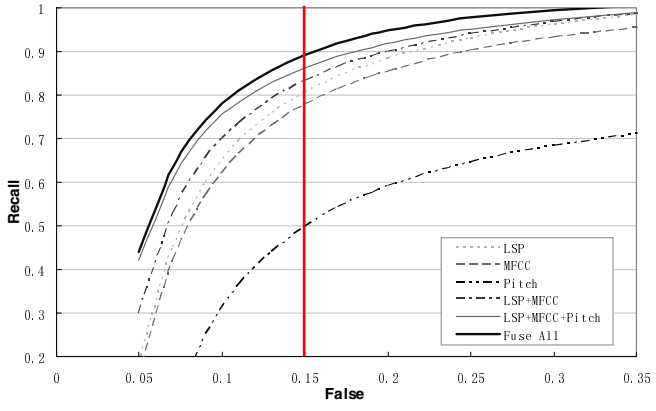

Fig. 9. False–recall curve for speaker segmentation

is least. After the fusion of LSP and MFCC, the recall is improved by about 3% and 6% over individual LSP and MFCC, respectively, when the false alarm rate is about 15%. With the same false alarm rate, the fusion of all three features can further improve the recall by about 3%. Actually, this means that the effect after pitch is added to fusion. The "fuse all" curve represents the final performance after considering noise modeling. It is seen that when the false alarm is 15%, the overall recall is improved up to 89%. Figure 9 also shows the trade-off between recall and false alarm and provides a reference to select different recall and false alarm rates for different applications.

Because there is no ground truth on speech/nonspeech information in the test database, an audio-type classifier was used to segment them. Unfortunately, there still exist some misclassifications. The detection results are strongly affected by the short nonspeech segments misclassified as speech segments, especially those sounds intervening in speech, such as a burst of wind, laughter, or applause. Many (>50%) false alarms are caused for this reason.

Since the algorithm is based on a resolution of 0.5 s and a context of 3–6 s, the detected speaker change point may shift from the true one. In the above experiment, we take it as a true detection if the shift is less than 3 s. Another experiment was carried out to obtain the statistics of the shift information. Figure 10 illustrates a shift histogram for the detected speaker change boundaries.

Figure 10 shows that almost 70% of detected boundaries are less than 1 s away from the true boundary, and 86.6% are

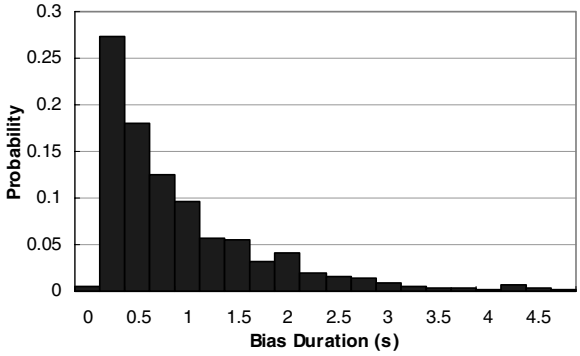

Fig. 10. Histogram of shift duration from true speaker change boundary

less than 2 s away. Only 7.6% are beyond the 3 s range. This also proves that our algorithm has good resolution in speaker boundary detection. It is very easy to increase the resolution from 0.5 s to 0.1 s, or even higher if we increase the overlapping of the shifting window. However, the computation complexity will increase linearly.

5.3 Speaker tracking

In our approach, the tracking problem is considered as a retrieval problem. For a special speaker, he/she has a ground truth speech set and a detected speech set; thus recall and precision can be used to evaluate the tracking results. In our implementation, the average recall and the average precision are used to evaluate the tracking performance. Here, average recall and precision are calculated from a weighed sum of recall and precision of each speaker in the test speech file.

Suppose there are N speakers in a test speech segment, and the recall and precision of the ith speaker is R_i and P_i, respectively; then:

$$\begin{aligned} R_{avr} &= \sum_{i=1}^{N} \alpha_i R_i \,, \\ P_{avr} &= \sum_{i=1}^{N} \alpha_i P_i \,, \end{aligned} \tag{17}$$

where the weight of the ith speaker is denoted by α_i, which can be calculated from:

$$\alpha_i = L_i / \sum_{i=1}^{N} L_i \,, \tag{18}$$

where L_i is the speech length of the ith speaker.

In the experiments, we first compared the speaker tracking performance with different schemes of k in Eq. 13, including $k = 1$, $k = \text{all}$, and our adaptive k scheme. Figure 11 shows the corresponding recall–false curve, where $F_{avr} = 1 - P_{avr}$. As shown in the figure, the performance is worst when k is chosen as 1, and our adaptive k has the best performance, which is 2–5% better than selecting all components in distance calculations.

In the above result, speaker identification is performed immediately after a speaker turn is detected. We also compared

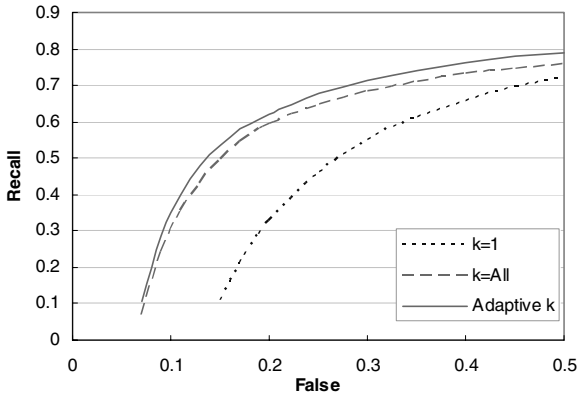

Fig. 11. Average false-recall curve with different k selection scheme

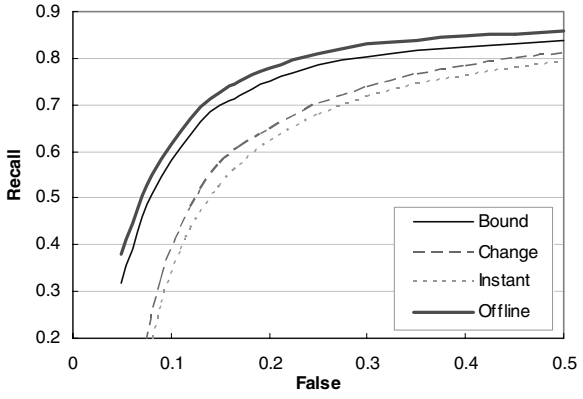

Fig. 12. Average false-recall curve for speaker tracking in four cases: (1) Instant: once at a real speaker change boundary. (2) Change: at the next potential speaker change. (3) Bound: at the next real speaker change boundary. (4) Offline

three positions where the speaker is identified. The first one is to perform speaker recognition instantly once a speaker change is found (*Instant*). The second one is to perform speaker recognition after the next potential change is found (*Change*). The third one is to perform speaker recognition when the next real speaker boundary is detected (*Bound*). Figure 12 illustrates the average false alarm vs. average recall curve in these three cases.

Figure 12 shows that, from the first case to the third case, when the available data increase, the corresponding performance also improves. The performance of *Bound* (the third case) is the best among these three cases, since it has the most data to model the current speaker for speaker tracking. When the false alarm rate is 0.1, the recall of *Bound* is 20% higher than that of the other two; when the false alarm rate is 0.2, the recall of *Bound* is 12% higher. The performance of *Change* (the second case) is only slightly better than that of *Instance* (the first case). It indicates that the data used in the second case are not yet sufficient to obtain an accurate speaker model. In our experiments, there are about 30 speakers in each testing speech file. The higher the speaker number is, the more confusion there will be in speaker identification.

In the experiments, we also implemented an offline speaker tracking system by storing all raw features in memory and using EM-based GMM-32 to update the speaker model instead of

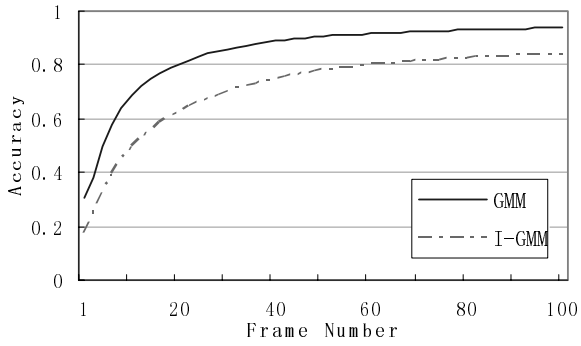

Fig. 13. Performance comparisons between EM-based GMM and incremental quasi-GMM in a speaker recognition system

our incremental quasi-GMM. The corresponding false–recall curve is also shown in Fig. 12 with a bold line. The figure shows that offline tracking yields only a slight (about 2–4%) performance improvement. This is because, initially, GMM-32 could not be fully estimated due to the small data size. It introduces some errors in speaker identification, and the error is further diffused in later speaker tracking. Thus, the overall performance is only slightly better than our real-time algorithm, while the speed is much slower.

In order to further clearly compare the performance between EM-based GMM and our incremental quasi-GMM, we implemented a speaker recognition system. This system contains 25 speakers, with each person having between 120 and 200 s of data for training and testing. Thus, the speaker models can be fully trained with enough data, and other influent factors, such as error diffusion, are excluded. Figure 13 illustrates the performance comparison using the curve of accuracy vs. testing length, which is measured in 25-ms frames. It can be seen that there is about 8–9% performance decrease when using incremental GMM instead of EM-based GMM. This means that the incremental quasi-GMM can roughly catch the main components in speaker modeling; when the testing length is longer, the accuracy between them could be closer.

Although in speaker recognition systems there is an 8–9% accuracy decrease, in speaker tracking, incremental GMM has only a 3% performance decrease due to other effects. This shows that it is appropriate to use incremental GMM in our speaker tracking system to meet the real-time requirement.

5.4 Computation efficiency

We have also tested the time complexity of our algorithm. With a Pentium III 864-MHz PC running Windows XP, the segmentation and tracking process can be completed in about 15% of the time-length of an audio clip. The LSP/MFCC correlation analysis is the most time-consuming part of our algorithm. After using an optimized function to compute correlation analysis, the time performance has been increased dramatically. Our scheme can totally meet the real-time processing requirement in multimedia applications.

6 Discussion and conclusion

In this paper, we have presented a novel approach to real-time unsupervised speaker segmentation and speaker tracking. A two-step speaker change detection algorithm is proposed that includes potential speaker change detection and refinement. Speaker tracking is based on the results of speaker change. A Bayesian fusion method is used to fuse different features, which include MFCC, LSP, pitch, and noise level, to obtain a more reliable result. The algorithm achieves 89% recall with a 15% false alarm rate on speaker segmentation and 76% recall with a 20% false alarm rate on unsupervised speaker tracking. Due to the proposed incremental GMM, the algorithm is computationally efficient and can perform in 15% of real time. Compared with EM-based GMM, incremental GMM has only 2–3% performance drops on speaker tracking, while it performs more than eight times faster. Although this system is designed for news broadcast processing, the same algorithms could be used in other audio applications.

There is still room for improvement in the proposed approach. In particular, our future research will be focused on addressing the following issues.

First, in order to process in real time, the available data to train speaker model are always limited. Estimating an accurate model from limited training data is still a challenge. Also, in news broadcasting, the environment and context are so complex that the segmentation result is often affected. In the experiments, we have found that if there is a burst of laughter between speeches, it is easily detected as a speaker change boundary. Therefore, another future focus will be on addressing this issue. Furthermore, environmental and channel variations also affect speaker tracking results. It has also been found that the same speaker in different environments sometimes is detected as different speakers. This indicates that our compensation for the mismatch effect of environment or channel is still insufficient.

References

1. Campbell JP (1997) JR. Speaker recognition: a tutorial. Proc IEEE 85(9):1437–1462
2. Brummer JNL (1994) Speaker recognition over HF radio after automatic speaker segmentation. In: Proc. IEEE South African symposium on communications and signal processing, COMSIG-94 pp 171–176
3. Sugiyama M, Murakami J, Watanabe H (1993) Speech segmentation and clustering based on speaker features. In: Proc. IEEE international conference on acoustics, speech, and signal processing
4. Wilcox L, Chen F, Kumber D, Balasubramanian V (1994) Segmentation of speech using speaker identification. In: Proc. IEEE international conference on acoustics, speech, and signal processing
5. Siu MH, Yu G, Gish H (1992) An unsupervised, sequential learning algorithm for the segmentation of speech waveform with multiple speakers. In: Proc. IEEE international conference on acoustics, speech, and signal processing, pp 189–192
6. Cohen A, Lapidus V (1996) Unsupervised speaker segmentation in telephone conversations. In: Proc. 19th convention of electrical and electronics engineers, Israel, pp 102—105
7. Gish H, Schmidt M (1994) Text-independent speaker identification. IEEE Signal Process Mag 11(4):18–32
8. Gish H, Siu MH, Rohlicek R (1991) Segregation of speakers for speech recognition and speaker identification. In: Proc. IEEE international conference on acoustics, speech, and signal processing, pp 873–876
9. Hermansky H, Morgan N (1994) RASTA processing of speech. IEEE Trans Speech Audio Process 2(4):578–589
10. Mammone RJ, Zhang XY, Ramachandran RP (1996) Robust speaker recognition: a feature-based approach. IEEE Signal Process Mag 13(5):58–71
11. Murthy HA, Beaufays F, Heck LP, Weintraub M (1999) Robust text-independent speaker identification over telephone channels. IEEE Trans Speech Audio Process 7(5):554–568
12. Mori K, Nakagawa S (2002) Speaker change detection and speaker clustering using VQ distortion for broadcast news speech recognition. In: Proc IEEE international conference on acoustics, speech, and signal processing
13. Chen S, Gopalakrishnan PS (1998) Speaker, environment and channel change detection and clustering via the Bayesian information criterion. In: Proc. DARPA workshopt on broadcast news transcription and understanding
14. Schwarz G (1978) Estimating the dimensions of a model. Ann Stat 6:461–464
15. Lu L, Jiang H, Zhang H.J (2001) A robust audio classification and segmentation method. In: Proc 9th ACM Multimedia, pp 203–211
16. Couvreur L, Boite JM (1999) Speaker tracking in broadcast audio material in the framework of the THISL project. In: Proc. ESCA ETRW workshop on accessing information in spoken audio, pp 84–89
17. Sonmez K, Heck L, Weintraub M (1999) Speaker tracking and detection with multiple speakers. In: Proc Eurospeech '1999, Budapest, 5:2219–2222
18. Bonastre JF, Delacourt P, Fredouille C, Merlin T, Wellekens C (2000) A speaker tracking system based on speaker turn detection for NIST evaluation. In: Proc IEEE international conference on acoustics, speech, and signal processing, pp 1177–1180
19. Fredouille C, Bonastre JF, Merlin T (1999) Segmental normalization for robust speaker verification. In: Workshop on robust method for speech recognition in adverse conditions, pp 103–106
20. Reynolds DA, Quatieri TF, Dunn RB (2000) Speaker verification using adapted Gaussian mixture models. Dig Signal Process 10:19–41
21. Padmanabhan M, Bahl LR, Nahamoo D, Picheny MA (1998) Speaker clustering and transformation for speaker adaptation in speech recognition systems. IEEE Trans Speech Audio Process 6(1):71–77
22. Berg BL, Beex AA (1999) Investigating speaker features from very short speech records. In: Proc. IEEE international symposium on circuits and systems (ISCAS'99), 3:102–105
23. Lu, L, Li SZ, Zhang H-J (2001) Content-based audio segmentation using support vector machines. In: Proc ICME01, pp 956–959
24. Roy D, Malamud C (1997) Speaker identification based text to audio alignment for an audio retrieval system. In: Proc IEEE international conference on acoustics, speech, and signal processing, pp 1099–1102
25. Kimber DG, Wilcox LD, Chen FR, Moran TP (1995) Speaker segmentation for browsing recorded audio. In: ACM CHI'95 Mosaic of Creativity, pp 212–213
26. Wang L, Chan KL (2000) Bayesian fusion: an approach for image retrieval using multiple features. In: Proc. international conference on image and vision computing, Hamilton, New Zealand
27. Abidi MA, Gonzalez RC (1992) Data fusion in robotics and machine intelligence. Academic, Boston 1992

28. Wang D, Lu L, Zhang H-J (2003) Speech segmentation without speech recognition. In: Proc. IEEE international conference on acoustics, speech and signal processing, 1:468–471
29. Lu L, Zhang H-J (2002) Speaker change detection and tracking in real-time news broadcasting analysis. In: Proc. 10th ACM international conference on multimedia, pp 602–610
30. Patel NV, Sethi IK (1997) Video classification using speaker identification. In: Proc. IS&T/SPIE conference on storage and retrieval for image and video databases, San Jose, CA, 5:218–225
31. Li D, Sethi IK, Dimitrova N, McGee T (2001) Classification of general audio data for content-based retrieval. Pattern Recog Lett 22(5):533–544