Time–Frequency Cepstral Features and Heteroscedastic Linear Discriminant Analysis for Language Recognition

Wei-Qiang Zhang, *Member, IEEE*, Liang He, Yan Deng, Jia Liu, *Member, IEEE*, and Michael T. Johnson, *Senior Member, IEEE*

***Abstract*—The shifted delta cepstrum (SDC) is a widely used feature extraction for language recognition (LRE). With a high context width due to incorporation of multiple frames, SDC outperforms traditional delta and acceleration feature vectors. However, it also introduces correlation into the concatenated feature vector, which increases redundancy and may degrade the performance of backend classifiers. In this paper, we first propose a time–frequency cepstral (TFC) feature vector, which is obtained by performing a temporal discrete cosine transform (DCT) on the cepstrum matrix and selecting the transformed elements in a zigzag scan order. Beyond this, we increase discriminability through a heteroscedastic linear discriminant analysis (HLDA) on the full cepstrum matrix. By utilizing block diagonal matrix constraints, the large HLDA problem is then reduced to several smaller HLDA problems, creating a block diagonal HLDA (BDHLDA) algorithm which has much lower computational complexity. The BDHLDA method is finally extended to the GMM domain, using the simpler TFC features during re-estimation to provide significantly improved computation speed. Experiments on NIST 2003 and 2007 LRE evaluation corpora show that TFC is more effective than SDC, and that the GMM-based BDHLDA results in lower equal error rate (EER) and minimum average cost (Cavg) than either TFC or SDC approaches.**

***Index Terms*—Language recognition (LRE), time–frequency cepstrum (TFC), block diagonal heteroscedastic linear discriminant analysis (BDHLDA).**

I. Introduction

LANGUAGE recognition (LRE) is a growing area within the field of speech signal processing. It has many applications, such as multilingual speech recognition, speech translation, multilanguage call centers, information security, and forensics [1]–[3]. Generally speaking, LRE acts as a front-end for *human–human* or *machine–human* systems—determining the type of language being spoken and routing the speech to specific back-ends.

Language recognition can be subdivided into two tasks: language identification and language verification. Language identification is a closed-set n-class classification problem which tries to determine *which language is/was being spoken*. On the other hand, language verification is an open-set two-class detection problem which tries to determine *whether the target language is/was being spoken*. In addition to these two cases, there is another typical task, open-set language recognition, which tries to judge which of a set of languages if any is/was spoken. Different tasks have different applications, but the underlying algorithms are similar.

Many methods have been developed for LRE. The most successful ones can be divided into two categories: acoustic model methods and phonotactic methods. In the acoustic model approach, the acoustic feature vectors of the speech are modeled directly by such methods as Gaussian mixture models (GMMs) [4], support vector machines (SVMs) [5], or SVMs with GMM super vectors (SVM GSVs) [6]. This method usually uses spectral (or cepstral) feature vectors, so it is also referred to as the *spectrum* method. In the phonotactic approach, the speech is first decoded into a token string or lattice, and then language models such as phoneme N-gram models [7]–[9], binary trees (BTs) [10], or vector space models (VSMs) [11] are applied. This method utilizes the intermediate results of decoders (or tokenizers), so it is also referred to as the *token* method.

No matter what approach is used, feature extraction is the first and possibly most important step for LRE. Feature vectors can be divided into two categories: basic feature vectors and derived feature vectors. Basic feature vectors are extracted from the speech signal directly while derived ones are further transformed from the basic feature sequences. Basic and derived feature vectors are often used together to achieve better performance.

In the acoustic model approach, Mel-frequency cepstral coefficients (MFCCs) [3], perceptual linear prediction (PLP) [12], and linear prediction cepstral coefficients (LPCCs) [13] are widely used basic feature vectors. Derived feature vectors have historically consisted of the first and second derivatives. However, in recent years, due to Torres-Carrasquillo's original contributions and Matejka's extensional work [4], [14],

Manuscript received October 07, 2009; revised March 24, 2010; accepted March 24, 2010. Date of publication April 08, 2010; date of current version October 27, 2010. This work was supported in part by the National Natural Science Foundation of China and Microsoft Research Asia under Grant 60776800, in part by the National Natural Science Foundation of China and Research Grants Council under Grant 60931160443 and in part by the National High Technology Development Program of China under Grants 2006AA010101, 2007AA04Z223, 2008AA02Z414, and 2008AA040201. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gokhan Tur.

W.-Q. Zhang, L. He, Y. Deng, and J. Liu are with the Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: wqzhang@tsinghua.edu.cn; heliang06@mails.tsinghua.edu.cn; y-deng05@mails.tsinghua.edu.cn; liuj@tsinghua.edu.cn).

M. T. Johnson is with the Electrical and Computer Engineering Department, Marquette University, Milwaukee, WI 53233 USA (e-mail: mike.johnson@mu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2010.2047680

the shifted delta cepstrum (SDC) has been proposed and has rapidly become the most prevalent derived feature vector in the acoustic model approach.

The SDC feature, which has much broader context than traditional feature vectors, captures additional discriminative information between languages and improves system performance. However, the SDC introduces correlation into the new feature vector, which may not be as effective for backend classifier modeling, such as the commonly used diagonal GMM.

Alternatives to this approach include feature transformation methods [15], which have received a lot of attention from speech signal processing community. Linear transformation algorithms, such as linear discriminant analysis (LDA) and heteroscedastic linear discriminant analysis (HLDA), have been successfully applied in language recognition and other speech recognition tasks [16]–[19].

Usually, the feature transformation is used on the entire final feature vector. For example in [18], the HLDA is performed on SDC concatenated with MFCC. In fact, it is possible to show that derived feature vectors can be expressed as a linear transformation of concatenated basic vectors. Feature derivation can thus be considered as a form of feature transformation through proper definition of transformation matrices.

This paper will first focus on the derived and then the transformed feature vectors for an acoustic model approach to LRE. Aimed at improving the performance of SDC features, we first propose a time–frequency cepstral feature vector, which extracts information from the continuous basic feature vector by utilizing the temporal discrete cosine transform (DCT). After that, we desire a HLDA method on the full feature vector directly, but this creates tremendous computational complexity. To address this, we introduce block diagonal matrix constraints and reduce the large HLDA problem to several smaller HLDA problems.

The rest of this paper is organized as follows. A simple review of commonly used derived feature vectors is provided in Section II. Section III presents the time–frequency cesptrum and Section IV proposes block diagonal HLDA. Section V demonstrates the effectiveness of each technology through detailed experiments. Finally, conclusions are given in Section VI.

II. Commonly Used Derived Feature Vectors

In LRE, the derived feature vectors are usually calculated from basic feature vectors and then appended to them to form a new feature vector. As discussed previously, these commonly used derived feature vectors include the differential cepstrum and SDC. We will introduce those briefly in this section.

A. Delta and Acceleration Cepstrum

Letting $\boldsymbol{c}_i$ represents the ith frame basic cepstrum vector, the first-order derivative cepstrum (usually referred as the delta cepstrum) can be expressed as

$$\boldsymbol{d}_i = \frac{\sum_{\tau=1}^{T} \tau(\boldsymbol{c}_{i+\tau} - \boldsymbol{c}_{i-\tau})}{2\sum_{\tau=1}^{T} \tau^2} \tag{1}$$

where τ is the frame delay and T is the window parameter for controlling context width.

The second-order derivative cepstrum (usually referred as the acceleration or delta-delta cepstrum) is defined in a similar way, except that the input is the delta cepstrum $\boldsymbol{d}_i$ and the output is the acceleration cepstrum $\boldsymbol{a}_i$. The new concatenated feature vector from basic, delta and acceleration cepstra is

$$\boldsymbol{y}_i^{\mathrm{C-D-A}} = \begin{bmatrix} \boldsymbol{c}_i \\ \boldsymbol{d}_i \\ \boldsymbol{a}_i \end{bmatrix}. \tag{2}$$

Usually, the context width parameter is set to $T = 2$ and the dimension of the basic feature vector is set to 13, which includes either the zeroth DCT coefficient C0 or the log-energy. Thus the total dimension is 39.

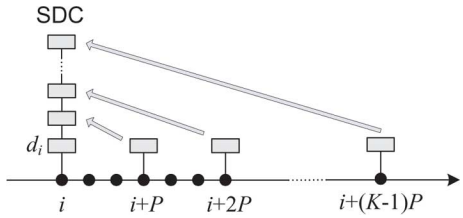


Fig. 1. Diagram of the shifted delta cepstrum (SDC).

B. Shifted Delta Cepstrum

In the SDC, a *simpler* form of the delta cepstrum is used instead of (1), defined as

$$\boldsymbol{d}_i = \boldsymbol{c}_{i+\tau} - \boldsymbol{c}_{i-\tau}. \tag{3}$$

The SDC is a stack of K-frames of this simple delta cepstrum, expressed as

$$\boldsymbol{\delta}_i = \begin{bmatrix} \boldsymbol{d}_i \\ \boldsymbol{d}_{i+P} \\ \vdots \\ \boldsymbol{d}_{i+(K-1)P} \end{bmatrix} \tag{4}$$

where K is the number of frames being stacked and P is the amount of frame shift. An illustration of the SDC is shown in Fig. 1.

Matejka *et al.* found that the performance of the SDC can be further improved if it is appended to the basic feature vector [14]. The new feature vector is

$$\boldsymbol{y}_i^{\mathrm{C-SDC}} = \begin{bmatrix} \boldsymbol{c}_i \\ \boldsymbol{\delta}_i \end{bmatrix}. \tag{5}$$

Empirically, researchers have found that when $N = 7$ (including zeroth DCT coefficient C0), $\tau = 1$, $P = 3$, and $K = 7$, the SDC gives quite good performance [14]. In this case, the total dimension is $7 + 7 \times 7 = 56$.

III. Time–Frequency Cepstrum

From the previous section, we can see that the SDC is essentially a downsampling of a sequence of simple delta ceptrum frames $[\boldsymbol{d}_i, \boldsymbol{d}_{i+1}, \ldots, \boldsymbol{d}_{i+(K-1)P}]$ without any anti-aliasing filtering. The total context is much broader than a single delta and acceleration cepstrum. Compared with direct concatenation, downsampling reduces the dimensionality. While this straightforward process is easy to implement, it also has two drawbacks: 1) there is no evidence to indicate whether the maximum information content of the simple delta cepstrum sequence is lower than the Nyquist frequency, $1/(2P)$ of the

frame rate, so the P-fold downsampling may cause the loss of significant useful information; and 2) although the concatenated simple delta cepstra are separated by P frames, they still have some temporal correlation, which will introduce correlation into the new feature vector.

In fact, we can see that the information content of the SDC feature vector comes from an equivalent single linear transform of a cepstrum matrix

$$\boldsymbol{X}_i = [\,\boldsymbol{c}_i \quad \boldsymbol{c}_{i+1} \quad \cdots \quad \boldsymbol{c}_{i+(M-1)}\,] \tag{6}$$

where M is the context width. This suggests the possibility that we can utilize the cepstrum matrix in a more optimal way instead of calculating and decimating the delta cepstrum.

The problem is how to extract more context information from the cepstrum matrix and yet remove the correlation between elements. This is similar to compression tasks in the image processing field, for which the 2-D DCT is often used, which can be thought of as a combined vertical and horizontal DCT. If the original image is $\boldsymbol{I}_o$, then the transformed image $\boldsymbol{I}_t$ can be obtained by

$$\boldsymbol{I}_t = \boldsymbol{C}_v \boldsymbol{I}_o \boldsymbol{C}_h^{\mathrm{T}} \tag{7}$$

where $\boldsymbol{C}_v$ and $\boldsymbol{C}_h$ are the vertical and horizontal transform matrix, respectively, and the superscript T denotes matrix transpose. The 2-D DCT is an effective method to de-correlate and reduce dimensionality in image processing.

In our case, however, the basic cepstral feature vectors have already been de-correlated. So to implement a 2-D DCT we need only perform a DCT in the temporal (horizontal) direction. Letting $\boldsymbol{C}$ denote a DCT transform matrix, the cepstrum matrix $\boldsymbol{X}_i$ can be de-correlated with

$$\boldsymbol{Y}_i = \boldsymbol{X}_i \boldsymbol{C}^{\mathrm{T}}. \tag{8}$$

After this operation, most of the variability in $\boldsymbol{X}_i$ will be concentrated in the coefficients in the upper left part of $\boldsymbol{Y}_i$, which corresponds to the *low-frequency* components of the 2-D DCT. To give a simple demonstration, the variance of each element of $\boldsymbol{Y}_i (M = 18, N = 10)$ was computed using the CallFriend corpus. The normalized variances (normalized by the maximum elements) are plotted in Fig. 2, and supports this assumption.

These components can be extracted to form a new feature vector by scanning the matrix in zigzag order as shown in Fig. 3. In this vector, the lower the index, the lower the frequency. The vector can then be truncated to D dimensions to form the TFC feature vector:

$$\boldsymbol{y}_i^{\mathrm{TFC}} = \mathrm{zig}_D(\boldsymbol{Y}_i) \tag{9}$$

where $\mathrm{zig}_D(\cdot)$ denotes rearrangement of the elements of the matrix in zigzag scan order, truncated to dimension D. The overall process for TFC feature extraction is shown in Fig. 4.

In order to give an intuitive example, the correlation coefficient matrix of the MFCC-SDC (56 dimensions) and the TFC (55 dimensions) were computed on the CallFriend corpus. The results are shown in Fig. 5. A clear correlation pattern can be seen in the off-diagonal elements of the SDC features, whereas the TFC features are much less correlated. The mean squares of the offdiagonal elements of the correlation coefficients matrices were also computed. The values for the MFCC-SDC and TFC

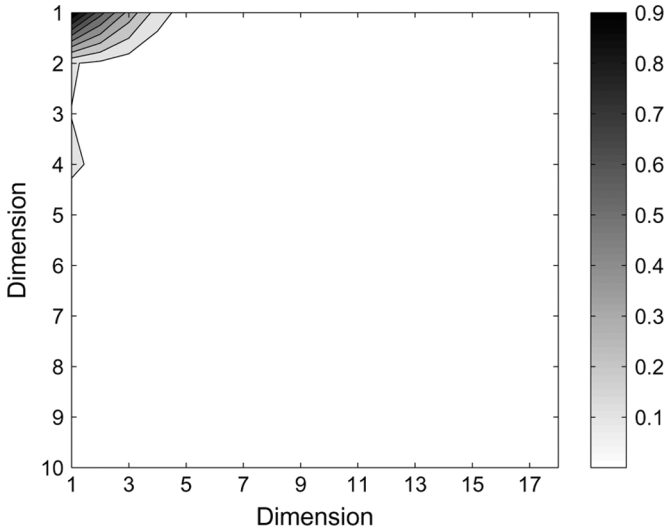


Fig. 2. Normalized variances of each element of the cepstrum matrix after a horizontal DCT.

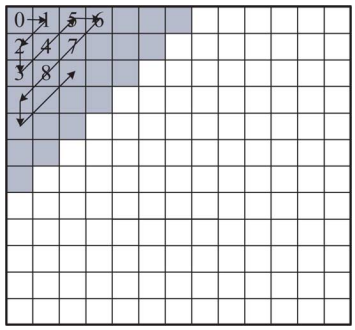


Fig. 3. Illustration of zigzag scan.

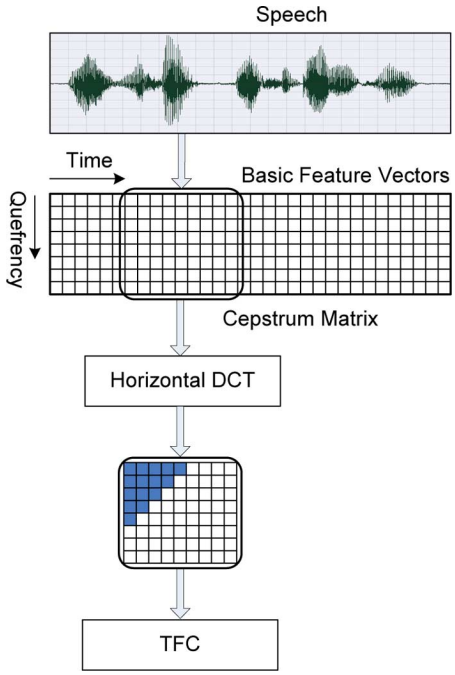


Fig. 4. Procedures of TFC feature extraction.

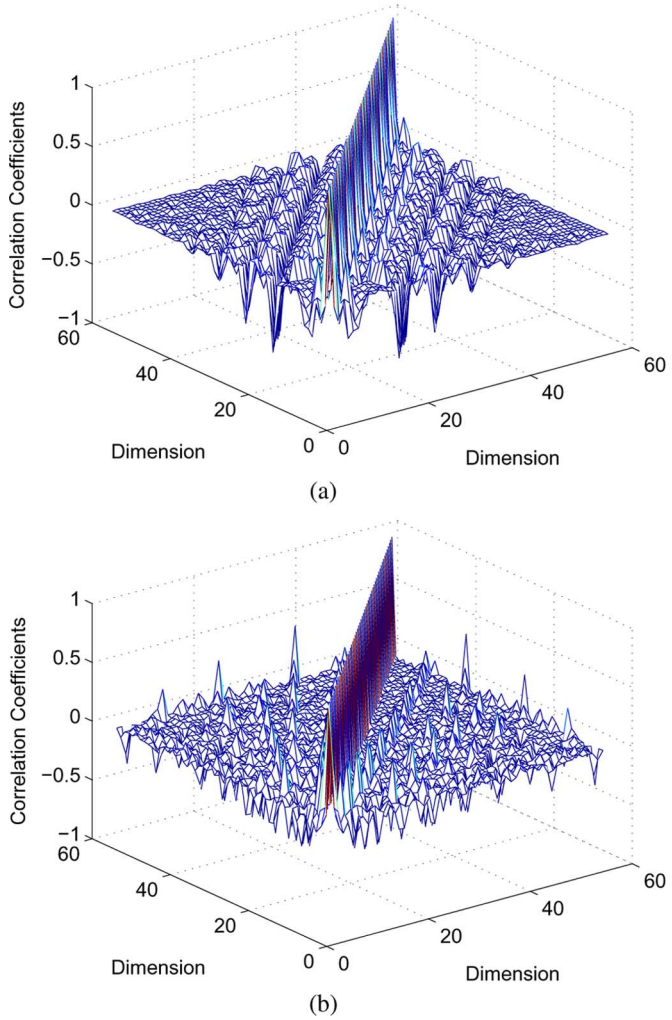


Fig. 5. Correlation coefficients matrix of MFCC-SDC and TFC feature vectors. (a) MFCC-SDC (56 dimensions). (b) TFC (55 dimensions).

were 0.0090 and 0.0057, respectively, which also indicates that the TFC features are less correlated than the SDC features.

Vaseghi *et al.* have proposed a cepstral-time matrix (CTM) feature vector [20], which is similar to the TFC feature proposed here. For extracting the CTM, a 2-D DCT is first performed on successive frames of sub-band energies to generate a *cepstrum-time matrix*, and then a low-rank sub-matrix is selected as elements of the feature vector.

Although our proposed TFC is similar to the CTM feature vector, there is one major difference between them. The TFC feature vector selects the elements using a zigzag scan order in the upper-left triangular area of the cepstrum-time matrix, while the CTM approach selects the entire upper-left rectangular area of the matrix. Due to the energy compaction properties of the DCT, the TFC structure concentrates the signal information into fewer coefficients than the CTM.

Castaldo *et al.* have performed detailed experiments on CTM feature vectors for language recognition [21]. The results show that the performance of the CTM is similar, or even slightly worse than, that of the SDC. But through the experiments in Section V, we will see that the TFC outperforms the SDC with similar configurations.

Note that we select an isosceles triangular area in TFC. There are other possible configurations, such as a nonsymmetric triangle or trapezoid, according to the variance pattern of Fig. 2. While this may lead to further improvements, here we focus on the isosceles case, which facilitates a zigzag scan.

IV. Block Diagonal HLDA

Although the TFC feature vector de-correlates each dimension, it is not optimal with respect to discriminability. Heteroscedastic linear discriminant analysis (HLDA) is an attractive tool to solve this problem, which has been successfully used in the speech processing community for feature extraction and dimensionality reduction [16]–[19]. HLDA, which is a generalization of LDA without a homoscedasticity assumption, projects the features into a low-dimensional subspace while preserving discriminative information. HLDA addresses two problems: 1) diagonalization, focusing on transforms that allow us to model all classes well with diagonal covariance Gaussians; and 2) dimensionality reduction, focusing on transforms that allow us to discard non-discriminative information. Thus, we may be able to gain additional performance improvement by replacing the DCT with HLDA.

A. Problem Statement

Suppose

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & \cdots & x_{1M} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NM} \end{bmatrix} \tag{10}$$

where the subscript i of $\boldsymbol{X}_i$ [see (6)] has been omitted for simplicity. Let $\boldsymbol{x}^{(n)}$ denote the column vector which is the transpose of nth row of $\boldsymbol{X}$

$$\boldsymbol{x}^{(n)} = \begin{bmatrix} x_{n1} & \cdots & x_{nM} \end{bmatrix}^{\mathrm{T}}. \tag{11}$$

We can obtain a supervector by concatenating each column vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{x}^{(N)} \end{bmatrix} \tag{12}$$

which is the operation of stacking the rows of $\boldsymbol{X}$ to a column vector.

For HLDA, we seek a matrix $\boldsymbol{A}$, which transfers D-dimensional $\boldsymbol{x}$ to a new vector $\boldsymbol{y}$

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{A}_U \boldsymbol{x} \\ \boldsymbol{A}_{D-U} \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}_U \\ \boldsymbol{y}_{D-U} \end{bmatrix} \tag{13}$$

where $\boldsymbol{A}_U$ consists of the first U rows of $\boldsymbol{A}$, $\boldsymbol{A}_{D-U}$ consists of the remaining $D-U$ rows, $\boldsymbol{y}_U$ are the useful dimensions, and $\boldsymbol{y}_{D-U}$ are the nondiscriminatory dimensions in the transformed space.

Let $\boldsymbol{x}_i$ denote a training sample and $g(i)$ indicate its class label. In the HLDA framework, the classes are modeled as full-covariance Gaussians [16] with two constraints: 1) all covariance matrices have the same orientation in the sense that they can be all diagonalized by the same linear transformation; 2) in the diagonalized space, means, and variances for some of the dimensions are shared across all classes/Gaussians. These are the non-discriminatory dimensions that are not effective to discriminate between classes. The probability density function of such a model is

$$p(\boldsymbol{x}_i) = \mathcal{N}\left(\boldsymbol{x}_i; \boldsymbol{B}\boldsymbol{\mu}_{g(i)}, \boldsymbol{B}\boldsymbol{\Sigma}_{g(i)}\boldsymbol{B}^{\mathrm{T}}\right) \tag{14}$$

where $\mathcal{N}(\cdot)$ denotes the normal distribution, and $\boldsymbol{B}\boldsymbol{\mu}_{g(i)}$ and $\boldsymbol{B}\boldsymbol{\Sigma}_{g(i)}\boldsymbol{B}^{\mathrm{T}}$ are the mean vector and covariance matrix of class $g(i)$. $\boldsymbol{\Sigma}_{g(i)}$ is assumed to be diagonal and $D - U$ of its diagonal coefficients are shared across all classes. The transformation matrix $\boldsymbol{B}$ is also shared by all classes. HLDA then represents the joint estimation of all these parameters $\boldsymbol{B}, \boldsymbol{\mu}_{g(i)}$ and $\boldsymbol{\Sigma}_{g(i)}$ in a maximum-likelihood (ML) sense. This can be equivalently calculated by defining the transformation

$$\boldsymbol{A} = \boldsymbol{B}^{-1}. \tag{15}$$

Suppose there are J classes, the jth (within-class) covariance matrix is $\boldsymbol{W}_j$ and the total (global) covariance matrix is $\boldsymbol{T}$. The objective function is defined as the loglikelihood of all the training samples, simplified as [16]

$$\mathcal{L}(\boldsymbol{A}; \{\boldsymbol{x}_i\}) = \sum_{j=1}^{J} \frac{I_j}{2} \log \times \left(\frac{|\boldsymbol{A}|^2}{|\mathrm{diag}(\boldsymbol{A}_U \boldsymbol{W}_j \boldsymbol{A}_U^{\mathrm{T}})||\mathrm{diag}(\boldsymbol{A}_{D-U} \boldsymbol{T} \boldsymbol{A}_{D-U}^{\mathrm{T}})|} \right) \tag{16}$$

where $\mathrm{diag}(\cdot)$ denotes the diagonal elements and $|\cdot|$ denotes the determinant of a matrix. The HLDA solution maximizes the objective function

$$\hat{\boldsymbol{A}} = \arg\max_{\boldsymbol{A}} \mathcal{L}(\boldsymbol{A}; \{\boldsymbol{x}_i\}). \tag{17}$$

In our case, $\boldsymbol{x}$ is a D-dimensional vector (where $D = M \times N$), which is typically a high-dimensional space, up to several hundred dimensions. Applying HLDA on $\boldsymbol{x}$ directly is computationally infeasible. In order to solve this problem, we will introduce some constraint conditions and decouple the larger HLDA problem into several smaller ones.

B. Block Diagonal Conditions

1) Transformation Matrix: If we constrain the general transformation to the horizontal direction, $\boldsymbol{A}$ will degenerate to a block diagonal matrix, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}^{(1)} & & \\ & \ddots & \\ & & \boldsymbol{A}^{(N)} \end{bmatrix} \tag{18}$$

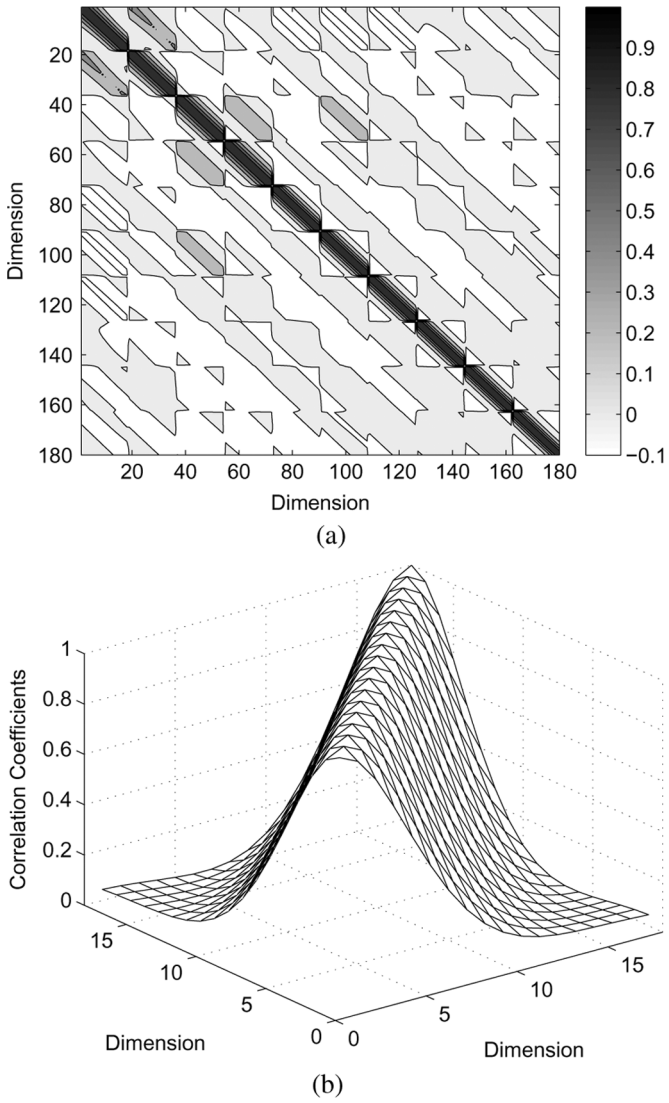


Fig. 6. Correlation coefficients matrix of the supervector $\boldsymbol{x}$ over the full data set. (a) Overview of all dimensions. (b) Partial enlargement of first 18 dimensions.

where each $\boldsymbol{A}^{(n)}$ is a $M \times M$ sub-matrix. In this case, $\boldsymbol{y}$ can be decomposed as

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}^{(1)} \\ \vdots \\ \boldsymbol{y}^{(N)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}^{(1)}\boldsymbol{x}^{(1)} \\ \vdots \\ \boldsymbol{A}^{(N)}\boldsymbol{x}^{(N)} \end{bmatrix}. \tag{19}$$

2) Covariance Matrix: For the covariance matrix, we assume that there is no correlation between different cepstral coefficients, which we believe to be sufficiently removed by DCT as a last step in extracting MFCCs. Thus, we assume only temporal correlation of individual cepstral coefficients. This will lead the total (global) covariance matrix $\boldsymbol{T}$ and each class covariance matrix $\boldsymbol{W}_j$ to have a block diagonal structure:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}^{(1)} & & \\ & \ddots & \\ & & \boldsymbol{T}^{(N)} \end{bmatrix} \tag{20}$$

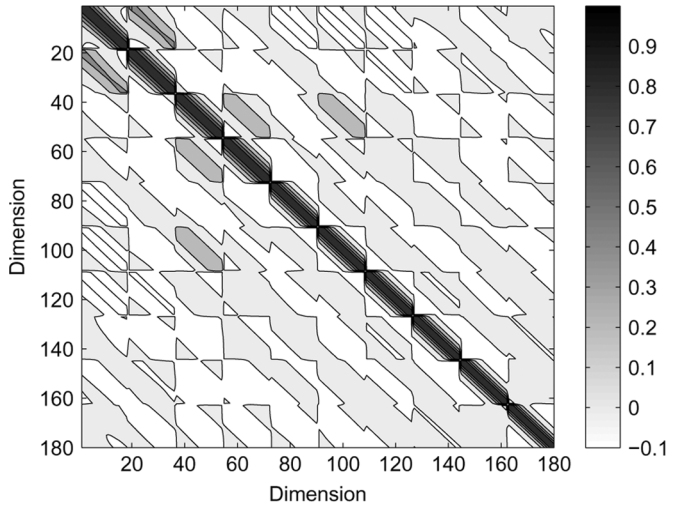


Fig. 7. Correlation coefficients matrix of the supervector $\boldsymbol{x}$ of the English data subset.

$$\boldsymbol{W}_j = \begin{bmatrix} \boldsymbol{W}_j^{(1)} & & \\ & \ddots & \\ & & \boldsymbol{W}_j^{(N)} \end{bmatrix}, \quad \forall j. \tag{21}$$

To illustrate this, we set $M = 18$ and $N = 10$ and compute the correlation coefficients matrix of $\boldsymbol{x}$ using the CallFriend corpus. The results obtained from the full data and English subset are shown in Figs. 6 and 7. (Other individual classes show similar patterns to Fig. 7, so they are not plotted here.) We can see that the covariance matrices have a nearly block diagonal form, which supports our assumption.

C. Decoupling the Objective Function

We can simplify the HLDA problem by utilizing the block diagonal conditions discussed above. Since we assume only temporal correlation of individual cepstral coefficients and no correlation across different coefficients, we only need to decorrelate the individual sub-vectors corresponding to temporal trajectories of different cepstral coefficients. Therefore, we need to estimate the individual transformations to the temporal sub-vectors of a particular cepstral coefficient matrix. We can show that using (18), (20), and (20), (16) can be decoupled as shown in (22) at the bottom of the page, where $U^{(n)}$ is the number of useful dimensions for nth HLDA problem and $\sum_{n=1}^{N} U^{(n)} = U$.

Given this block diagonal structure of covariance matrices, the large problem is decoupled into several smaller problems. So we refer to this algorithm as block diagonal HLDA (BDHLDA). Using this strategy, the solution of the whole problem becomes

$$\hat{\boldsymbol{A}} = \begin{bmatrix} \hat{\boldsymbol{A}}^{(1)} & & \\ & \ddots & \\ & & \hat{\boldsymbol{A}}^{(N)} \end{bmatrix} \tag{23}$$

where $\hat{\boldsymbol{A}}^{(n)}$ is the solution of nth smaller problem

$$\hat{\boldsymbol{A}}^{(n)} = \arg\max_{\boldsymbol{A}^{(n)}} \mathcal{L}^{(n)}\left(\boldsymbol{A}^{(n)}, \left\{\boldsymbol{x}_i^{(n)}\right\}\right). \tag{24}$$

D. Algorithm Complexity

Suppose

$$\operatorname{diag}\left(\boldsymbol{A}_{U^{(n)}}^{(n)} \boldsymbol{W}_j^{(n)} \left(\boldsymbol{A}_{U^{(n)}}^{(n)}\right)^{\mathrm{T}}\right) = \begin{bmatrix} \left(\sigma_j^{(n)}\right)_1^2 & & \\ & \ddots & \\ & & \left(\sigma_j^{(n)}\right)_M^2 \end{bmatrix} \tag{25}$$

$$\operatorname{diag}\left(\boldsymbol{A}_{M-U^{(n)}}^{(n)} \boldsymbol{T}^{(n)} \left(\boldsymbol{A}_{M-U^{(n)}}^{(n)}\right)^{\mathrm{T}}\right) = \begin{bmatrix} \left(\sigma^{(n)}\right)_1^2 & & \\ & \ddots & \\ & & \left(\sigma^{(n)}\right)_M^2 \end{bmatrix}. \tag{26}$$

Through some straightforward derivations, we obtain [22]

$$\left(\boldsymbol{a}_m^{(n)}\right)' = \boldsymbol{c}_m^{(n)} \left(\boldsymbol{G}_m^{(n)}\right)^{-1} \sqrt{\frac{1}{\boldsymbol{c}_m^{(n)} \boldsymbol{G}_m^{(n)} \left(\boldsymbol{c}_m^{(n)}\right)^{\mathrm{T}}}} \tag{27}$$

where $(\boldsymbol{a}_m^{(n)})'$ is the mth row of the transformation matrix $\boldsymbol{A}^{(n)}$. $\boldsymbol{c}_m^{(n)}$ is the mth row of the cofactor matrix

$$\begin{aligned} &\mathcal{L}(\boldsymbol{A}; \{\boldsymbol{x}_i\}) \\ &= \sum_{j=1}^{J} \frac{I_j}{2} \sum_{n=1}^{N} \log\left(\frac{\left|\boldsymbol{A}^{(n)}\right|^2}{\left|\operatorname{diag}\left(\boldsymbol{A}_{U^{(n)}}^{(n)} \boldsymbol{W}_j^{(n)} \left(\boldsymbol{A}_{U^{(n)}}^{(n)}\right)^{\mathrm{T}}\right)\right| \left|\operatorname{diag}\left(\boldsymbol{A}_{M-U^{(n)}}^{(n)} \boldsymbol{T}^{(n)} \left(\boldsymbol{A}_{M-U^{(n)}}^{(n)}\right)^{\mathrm{T}}\right)\right|} \right) \\ &= \sum_{n=1}^{N} \sum_{j=1}^{J} \frac{I_j}{2} \log\left(\frac{\left|\boldsymbol{A}^{(n)}\right|^2}{\left|\operatorname{diag}\left(\boldsymbol{A}_{U^{(n)}}^{(n)} \boldsymbol{W}_j^{(n)} \left(\boldsymbol{A}_{U^{(n)}}^{(n)}\right)^{\mathrm{T}}\right)\right| \left|\operatorname{diag}\left(\boldsymbol{A}_{M-U^{(n)}}^{(n)} \boldsymbol{T}^{(n)} \left(\boldsymbol{A}_{M-U^{(n)}}^{(n)}\right)^{\mathrm{T}}\right)\right|} \right) \\ &= \sum_{n=1}^{N} \mathcal{L}^{(n)}\left(\boldsymbol{A}^{(n)}, \{\boldsymbol{x}_i^{(n)}\}\right) \end{aligned} \tag{22}$$

$\boldsymbol{C}^{(n)} = |\boldsymbol{A}^{(n)}|(\boldsymbol{A}^{(\boldsymbol{n})})^{-1}$ for the current estimate of $A^{(n)}$. $\boldsymbol{G}_m^{(n)}$ is

$$\boldsymbol{G}_m^{(n)} = \begin{cases} \sum_{j=1}^{J} \frac{I_j}{\left(\sigma_j^{(n)}\right)_m^2} \boldsymbol{W}_j^{(n)} & 1 \le m \le U^{(n)} \\ \frac{I}{\left(\sigma^{(n)}\right)_m^2} \boldsymbol{T}^{(n)} & U^{(n)} < m \le M. \end{cases} \tag{28}$$

Estimation of matrix $A^{(n)}$ is an iterative procedure, where we iteratively reestimate rows of $A^{(n)}$ until convergence.

In the BDHLDA or HLDA algorithm, the iteration computation load is not that significant. The real bottleneck for computation lies in calculating the statistics, especially the covariance matrices, from the training samples. In BDHLDA, $\boldsymbol{T}^{(n)}$ and $\boldsymbol{W}_j^{(n)}$ are both $M \times M$ matrices. If the number of the training sample is I, the computational complexity will be $O(M^2 I)$. The BDHLDA consists of N HLDA problems, so the total computational complexity is $O(NM^2 I)$. If we do not use the constraint conditions, the computational complexity of the large-scale HLDA algorithm will be $O((NM)^2 I)$. Thus, the computational complexity of BDHLDA is thus just $1/N$ of that of HLDA.

E. GMM-Based BDHLDA

In the previous section, we discussed the BDHLDA algorithm, which assumes that classes are Gaussian distributed. In LRE, a natural method is to treat each language as a class. However, with this approach, the classes would have largely non-Gaussian distributions and would thus be unlikely to give good performance. An effective solution is to use a GMM to model the data. Burget *et al.* uses this method to improve the performance of the HLDA algorithm [19]. This strategy can also be used in the BDHLDA algorithm; however, due to the high-dimensionality of the original features, Burget's method cannot be applied directly.

Before giving the GMM-based BDHLDA, let us first review the GMM-based HLDA algorithm. For each class (in our case, one class denotes one language), we can train a GMM using the original feature vectors. Each Gaussian component will give a fine partition for the feature space; thus, we obtain many subclasses corresponding to the components. GMM gives a *soft* partition, i.e., each training sample belongs to several subclasses with certain occupation probability, so we need to calculate the statistics according to this probability.

Suppose the jth class is a modeled as a GMM with parameters $\lambda_j = \{w_{jg}, \boldsymbol{\mu}_{jg}, \boldsymbol{\Sigma}_{jg}\}_{g=1}^{G}$, with probability density function (pdf) for one frame of original feature vector $\boldsymbol{x}_i$

$$p(\boldsymbol{x}_i|\lambda_j) = \sum_{g=1}^{G} w_{jg} \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{jg}, \boldsymbol{\Sigma}_{jg}). \tag{29}$$

We can obtain the gth component's occupation (posterior probability) as

$$\gamma_{jg}(\boldsymbol{x}_i) = \frac{w_{jg}\mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}_{jg}, \boldsymbol{\Sigma}_{jg})}{p(\boldsymbol{x}_i|\lambda_j)}. \tag{30}$$

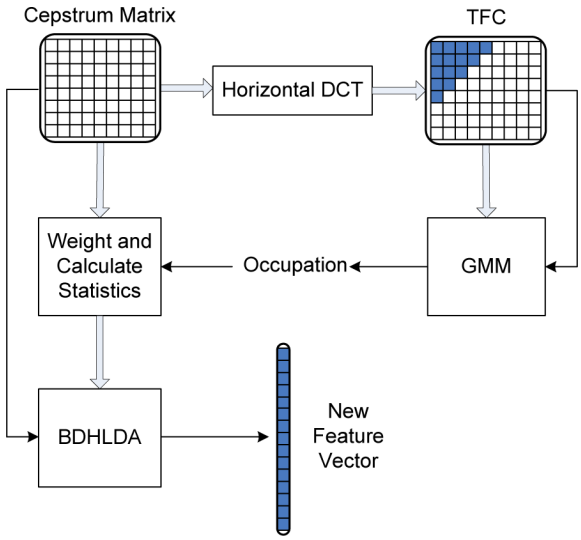


Fig. 8. GMM-based BDHLDA.

Based on this occupation, the statistics are

$$I_{jg} = \sum_{g(i)=j} \gamma_{jg}(\boldsymbol{x}_i) \tag{31}$$

$$\boldsymbol{m}_{jg} = \frac{1}{I_{jg}} \sum_{g(i)=j} \gamma_{jg}(\boldsymbol{x}_i)\boldsymbol{x}_i \tag{32}$$

$$\boldsymbol{W}_{jg} = \frac{1}{I_{jg}} \sum_{g(i)=j} \gamma_{jg}(\boldsymbol{x}_i)(\boldsymbol{x}_i - \boldsymbol{m}_{jg})(\boldsymbol{x}_i - \boldsymbol{m}_{jg})^{\mathrm{T}}. \tag{33}$$

In this way, the feature space from J classes can be broken into $L \times G$ subclasses. This is the GMM-based HLDA algorithm. We can see that this algorithm gives a finer partition of the feature space and also better satisfies the distribution assumption.

For the BDHLDA algorithm, however, the dimension of the original feature vector is very high. The computation load for training the GMM is tremendous, and thus calculating $\gamma_{jg}(\boldsymbol{x}_i)$ is not feasible. The BDHLDA statistics can be collected in a one pass retraining fashion, with mixture component occupation probabilities computed using TFC features and the corresponding models.

As illustrated in Fig. 8, we can summarize the GMM-based BDHLDA method as follows.

1) Construct the feature matrix $\boldsymbol{X}_i$ by using the basic feature vectors, and then perform a horizontal DCT to obtain the TFC feature vectors $\boldsymbol{y}_i^{\mathrm{TFC}}$.
2) Using the TFC features, train a GMM for each language.
3) Calculate the occupation likelihood $\gamma_{jg}(\boldsymbol{y}_i^{\mathrm{TFC}})$ for each TFC feature vector.
4) Using $\gamma_{jg}(\boldsymbol{y}_i^{\mathrm{TFC}})$ as the weight, calculate the statistics of $\boldsymbol{x}_i^{(n)}$ for each n.
5) Using these statistics, solve each HLDA sub-problem, and then obtain the solution for each. When solving the HLDA, set the useful number of dimensions to $U^{(n)}$ for the nth problem.

6) Using the transform matrix of each HLDA, transform $\boldsymbol{x}_i^{(n)}$ and get

$$\boldsymbol{y}_i^{(n)} = \boldsymbol{A}^{(n)} \boldsymbol{x}_i^{(n)}. \tag{34}$$

7) Let $(\boldsymbol{y}_i^{(n)})_{U^{(n)}}$ denote the first $U^{(n)}$ dimensions of $\boldsymbol{y}_i^{(n)}$. Concatenate $\{(\boldsymbol{y}_i^{(n)})_{U^{(n)}}, n = 1, 2, \ldots, N\}$ to get the new feature vector

$$\boldsymbol{y}_i^{\text{BDHLDA}} = \begin{bmatrix} \left(\boldsymbol{y}_i^{(1)}\right)_{U^{(1)}} \\ \left(\boldsymbol{y}_i^{(2)}\right)_{U^{(2)}} \\ \vdots \\ \left(\boldsymbol{y}_i^{(N)}\right)_{U^{(N)}} \end{bmatrix}. \tag{35}$$

V. Experiments

A. Experimental Setup

The TFC and BDHLDA feature vectors are evaluated on NIST LRE data. We perform our experiments on 2003 LRE data (LRE03) and 2007 LRE data (LRE07). LRE03 has a simple channel condition and matched training data, which provides a relatively pure test condition, so we use it to make initial performance comparison and parameter optimization. LRE07 has several miscellaneous data sources and is more challenging than LRE03, so we use it to give further validation in our experiments.

In addition, in order to speed up the training process for small scale testing, we use only every 20th feature vector for training to give a reduced training set. We will label this as *1/20 training* in contrast with *full training* which corresponds to using all the feature vectors for training in our experiments.

1) Experimental Data: For LRE03, the training data comes from the CallFriend corpus, which consists of Arabic, English (southern and non-southern English), Farsi, French, German, Hindi, Japanese, Korean, Mandarin (mainland and Taiwan Mandarin), Spanish (Caribbean and non-Caribbean Spanish), Tamil, and Vietnamese telephone speech. Each language/dialect contains 60 half-hour conversations.

For LRE07, the training data comes from CallFriend, CallHome, OGI, OHSU, and LRE07Train corpus. The target languages include Arabic, Bengali, Chinese (Cantonese, Mandarin, Wu, and Min), English (American and Indian English), Farsi, German, Hindustani (Hindi and Urdu), Japanese, Korean, Russian, Spanish (Caribbean and non-Caribbean Spanish), Tamil, Thai, and Vietnamese.

2) Experimental Setup: The evaluation is performed in the framework of NIST LRE [23]. The detection task is done for each language and the closed-set pooled equal error rate (EER) and minimum average cost (Cavg) [23] are used as performance measures. We use diagonal GMM as the classifiers to validate the performance of the proposed feature vectors. Each language is modeled as a GMM, with 256 mixture components in preliminary experiments and with 512 mixture components in large scale experiments. The GMMs are first trained via maximum likelihood (ML) criteria with eight iterations, and then trained via maximum mutual information (MMI) criteria [14] with 20 iterations.

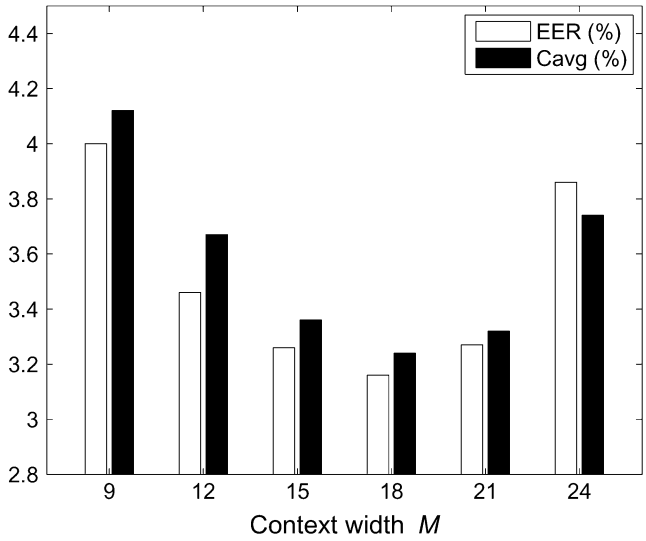


Fig. 9. TFC feature dimension $D = 36$, LRE03, 1/20 training, 30-s test.

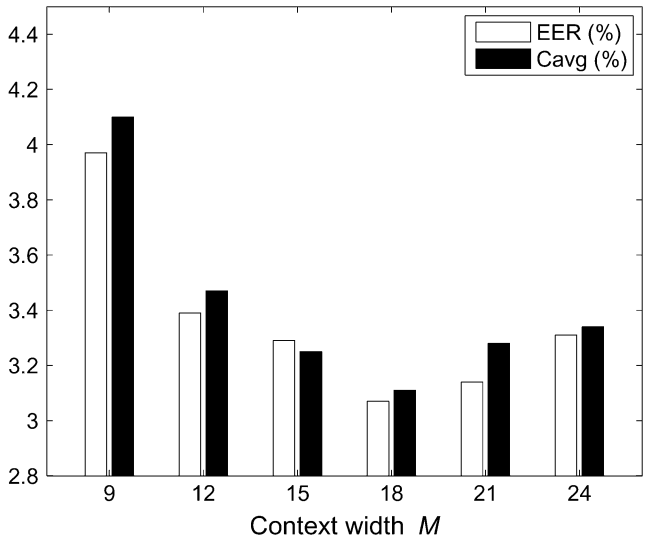


Fig. 10. TFC feature dimension $D = 45$, LRE03, 1/20 training, 30-s test.

B. TFC Feature Vector

In the TFC feature vector, the context width M and feature dimension D are the control parameters. With a zigzag scan (see Fig. 3), we create a triangular area of elements. We first fix the dimension D as 36, 45, and 55, corresponding to increasing triangles, and vary the context width M from 9 to 24 with step size 3.

The results on LRE03 30 s duration segments with $1/20$ training are illustrated in Figs. 9–11. From the results, we can see that across feature dimensions, $M = 18$ always gives the best performance. With SDC feature vectors, the context width for the optimized parameters $(N, \tau, P, K) = (7, 1, 3, 7)$ is 21. The optimized TFC and SDC have a similar context width, which reveals that the discriminative information for different languages may primarily lie in broad temporal segments.

Next, we fix the context width $M = 18$ and vary the feature dimensions from 36 to 78 (which corresponds to varying the right side length of the triangular area from 8 to 12). The results are shown in Fig. 12. From this figure, we can observe that when $D = 55$, the TFC feature vector obtains best performance. This

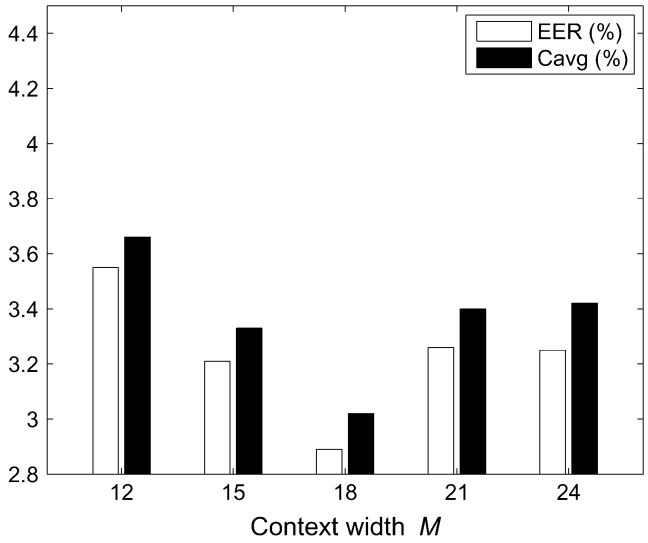


Fig. 11. TFC feature dimension $D = 55$, LRE03, 1/20 training, 30-s test.

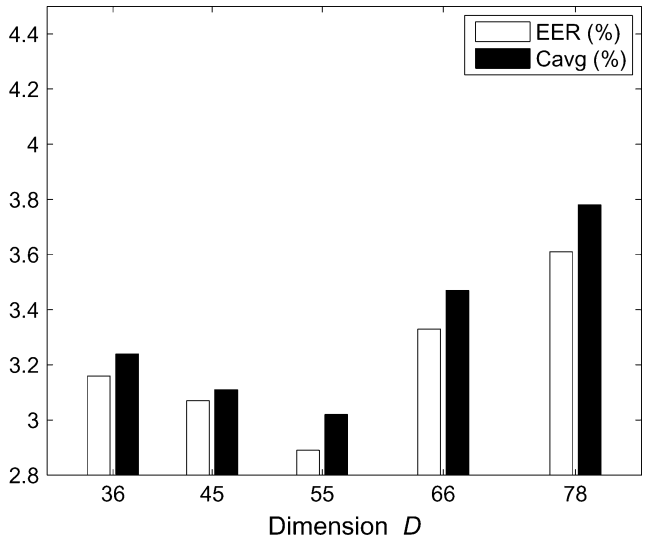


Fig. 12. TFC feature context width $M = 18$, LRE03, 1/20 training, 30-s test.

value is also very similar to the SDC feature with dimension 56.

Through the above experiments, we obtained the optimized parameters for the TFC feature vector as $(M, D) = (18, 55)$. In the following experiments, we will use these values as the default configuration.

After optimizing the parameters, we compare the TFC with other common derived feature vectors. Features tested include MFCC concatenated with delta and acceleration (labeled as MFCC-D-A), MFCC concatenated with SDC (labeled as MFCC-SDC), CTM (using the parameters labeled as DCT 7-6-21 in [21], which achieve the best result for LRE03 30 s test.) and TFC feature vectors. All the parameters and results are listed in Table I. From the results, we can see that the MFCC-D-A, which uses a relatively short context and lower dimensionality, results in higher EER and Cavg. The MFCC-SDC, CTM, and TFC have broader context width and higher dimension, their EERs and Cavgs are lower than that of MFCC-D-A. The TFC feature vector has similar parameters to MFCC-SDC and CTM, but outperforms them.

TABLE I
COMPARISON OF DIFFERENTIAL CEPSTRUM, SDC, CTM, AND TFC FEATURE VECTORS, LRE03, 1/20 TRAINING, 30-s TEST

Feature vector	Context width	Dimension	EER (%)	Cavg (%)
MFCC-D-A	9	39	6.06	6.44
MFCC-SDC	21	56	3.63	3.68
CTM	21	49	3.52	3.58
TFC	18	55	2.89	3.02

TABLE II
COMPARISON OF TFC, BDHLDA, AND GMM-BASED BDHLDA FEATURE VECTORS, LRE03, 1/20 TRAINING, 30-s TEST

Feature vector	Context width	Dimension	EER (%)	Cavg (%)
TFC	18	55	2.89	3.02
BDHLDA	18	55	2.85	2.98
GBDHLDA	18	55	2.68	2.87

C. BDHLDA

We compare BDHLDA, GMM-based BDHLDA (labeled as GBDHLDA), and TFC feature vectors on the LRE03 task. For the TFC features, the optimized parameters are $M = 18$ and $D = 55$ (so that the cepstrum matrix has $M = 18$ and $N = 10$, with resulting dimension $MN = 180$). For BDHLDA, we treat each language as a class and use the DCT matrix to initialize the transform matrix. In order to give a fair comparison, we set $M = 18$ and $N = 10$. For nth $(n = 1, 2, \ldots, 10)$ small problem of BDHLDA, the number of useful dimensions $U^{(n)}$ is set as $11 - n$.

The results are shown in Table II. We can see that the BDHLDA is slightly better than TFC, suggesting that the BDHLDA de-correlates better than the horizontal DCT. On the other hand, the GMM-based BDHLDA gives additional improvement compared with BDHLDA because its characteristics better fit the true data distribution. Although GMM-based BDHLDA has a higher computational load than BDHLDA, it does give some additional performance gain.

D. Large Scale Experiments

In this section, we test the MFCC-SDC, TFC, and GMM-based BDHLDA using the full training set. We increase GMM mixture components to 512 and perform the evaluation on LRE03 and LRE07 and test on 30-s, 10-s, and 3-s durations. For the GMM-based BDHLDA, we use the equalized HLDA (EHLDA) [24] to balance the training data of each language. Note that we only adjust the weight between languages, while retaining the proportion of Gaussian components within each language. The detection error trade-off (DET) curves are showed in Figs. 13 and 14, and the EERs and Cavgs are listed in Tables III and IV. We also provide the results obtained by ML-trained GMMs for comparison. From the results, we can see the consistent performance improvement due to changing from SDC to TFC and to BDHLDA, especially

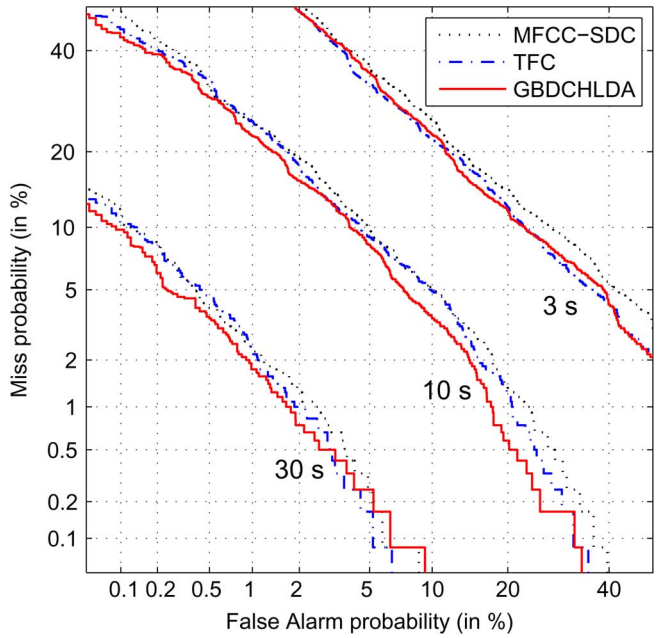


Fig. 13. DET curves of SDC, TFC, and GMM-based BDHLDA feature vectors, LRE03, full training.

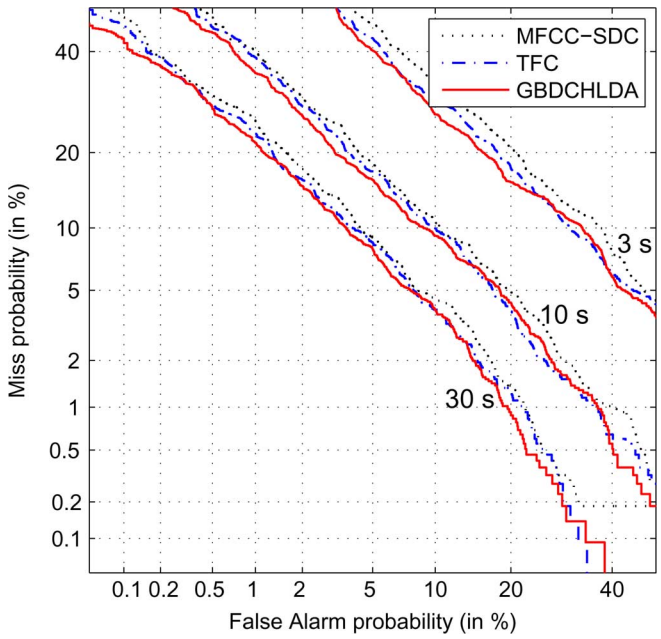


Fig. 14. DET curves of SDC, TFC, and GMM-based BDHLDA feature vectors, LRE07, full training.

for the 30-s duration segments. For LRE03, the Cavg decreased from 1.54% to 1.36% and further to 1.31%, and for LRE07, the Cavg decreased from 7.06% to 6.85% and further to 6.56%.

VI. Conclusion

In this paper, we have proposed two approaches to improve the extensively used SDC feature vector for language recognition. To do this, we have developed theoretically founded methods for capturing information from the time–cepstrum matrix. These methods include the TFC feature vector, based on a horizontal DCT of the cepstrum matrix for de-correlation, coupled with feature selection using zigzag scan order for maximal information content. This initial idea is then extended from a feature de-correlation focus to a feature discriminability focus by developing a BDHLDA algorithm, which is essentially an HLDA on the entire cepstrum matrix with block diagonal matrix constraints to give lower computational complexity. The BDHLDA approach is finally extended to work in the GMM model domain, using the TFC features internally to provide computationally efficient implementation. Experiments on NIST 2003 and 2007 LRE evaluation corpus show that the TFC is more effective than the SDC and that the final GMM-based BDHLDA is more effective than either SDC or TFC features.

TABLE III
Comparison of SDC, TFC, and GMM-Based BDHLDA Feature Vectors, LRE03, Full Training

Feature vector	Training method	EER (%)			Cavg (%)		
		30 s	10 s	3 s	30 s	10 s	3 s
MFCC-SDC	ML	6.73	15.28	22.02	6.79	15.19	22.68
TFC	ML	6.14	14.22	21.51	6.49	14.74	21.43
GBDHLDA	ML	5.58	14.19	20.14	5.62	14.45	20.44
MFCC-SDC	MMI	1.62	7.01	16.54	1.54	7.33	16.89
TFC	MMI	1.44	6.90	15.99	1.36	7.12	15.79
GBDHLDA	MMI	1.32	6.27	14.98	1.31	6.46	15.33

TABLE IV
Comparison of SDC, TFC, and GMM-Based BDHLDA Feature Vectors, LRE07, Full Training

Feature vector	Training method	EER (%)			Cavg (%)		
		30 s	10 s	3 s	30 s	10 s	3 s
MFCC-SDC	ML	13.74	17.39	25.55	13.27	16.96	25.41
TFC	ML	13.11	17.09	24.50	12.67	16.26	24.53
GBDHLDA	ML	12.27	16.62	24.24	12.25	16.18	24.38
MFCC-SDC	MMI	6.84	10.30	20.35	7.06	10.49	20.46
TFC	MMI	6.53	9.97	18.69	6.85	10.03	18.73
GBDHLDA	MMI	6.11	9.61	17.90	6.56	9.95	18.18

References

[1] Y. K. Muthusamy, E. Barnard, and R. A. Cole, "Reviewing automatic language identification," *IEEE Signal Process. Mag.*, vol. 11, no. 4, pp. 33–41, Oct. 1994.

[2] M. A. Zissman and K. M. Berkling, "Automatic language identification," *Speech Commun.*, vol. 35, no. 1–2, pp. 115–124, Aug. 2001.

[3] M. A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech Audio Process.*, vol. 4, no. 1, pp. 31–34, Jan. 1996.

[4] P. A. Torres-Carrasquillo, "Language identification using Gaussian mixture models," Ph.D. dissertation, Michigan State Univ., East Lansing, 2002.

[5] W. Zhang, B. Li, D. Qu, and B. Wang, "Automatic language identification using support vector machines," in *Proc. ICSP*, Guilin, China, Nov. 2006.

[6] B. Campbell *et al.*, "2007 NIST LRE MIT lincoln laboratory site presentation," in *Proc. 2007 NIST Lang. Recognition Eval. Workshop*, Orlando, FL, Dec. 2007.

[7] T. J. Hazen and V. W. Zue, "Automatic language identification using a segment-based approach," in *Proc. Eurospeech*, Berlin, Sep. 1993, vol. 2, pp. 1303–1306.

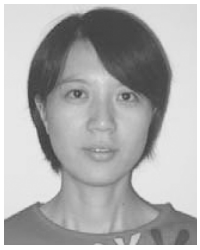
[8] M. A. Zissman and E. Singer, "Automatic language identification of telephone speech messages using phoneme recognition and N-gram modeling," in *Proc. ICASSP*, Adelaide, Australia, Apr. 1994, vol. 1, pp. 305–308.
[9] J.-L. Gauvain, A. Messaoudi, and H. Schwenk, "Language recognition using phone lattices," in *Proc. Interspeech*, Jeju Island, Korea, Oct. 2004, pp. 25–28.
[10] J. Navratil, "Spoken language recognition—A step toward multilinguality in speech processing," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 6, pp. 678–685, Sep. 2001.
[11] H. Li, B. Ma, and C.-H. Lee, "A vector space modeling approach to spoken language identification," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 1, pp. 271–284, Jan. 2007.
[12] B. Yin, E. Ambikairajah, and F. Chen, "Combining cepstral and prosodic features in language identification," in *Proc. ICPR*, Hong Kong, Aug. 2006, vol. 4, pp. 254–257.
[13] F. J. Goodman, A. F. Martin, and R. E. Wohlford, "Improved automatic language identification in noisy speech," in *Proc. ICASSP*, Glasgow, U.K., May 1989, vol. 1, pp. 528–531.
[14] P. Matejka *et al.*, "Brno university of technology system for NIST 2005 language recognition evaluation," in *Proc. IEEE Odyssey*, San Juan, Puerto Rico, Jun. 2006.
[15] R. O. Duda and P. B. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[16] N. Kumar and A. G. Andreou, "Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition," *Speech Commun.*, vol. 26, no. 4, pp. 283–297, Dec. 1998.
[17] G. Garau and S. Renals, "Combining spectral representations for large-vocabulary continuous speech recognition," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 3, pp. 508–518, Mar. 2008.
[18] L. Burget, P. Matejka, and J. Cernocky, "Discriminative training techniques for acoustic language identification," in *Proc. ICASSP*, Toulouse, France, May 2006, vol. 1, pp. 209–212.
[19] L. Burget *et al.*, "Analysis of feature extraction and channel compensation in a GMM speaker recognition system," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 7, pp. 1979–1986, Sep. 2007.
[20] S. V. Vaseghi, P. N. Conner, and B. P. Milner, "Speech modelling using cepstral-time feature matrices in hidden Markov models," in *Proc. IEE-I*, Oct. 1993, vol. 140, no. 5, pp. 317–320.
[21] F. Castaldo *et al.*, "Language identification using acoustic models and speaker compensated cepstral-time matrices," in *Proc. ICASSP*, Honolulu, HI, Apr. 2007, vol. 4, pp. 1013–1016.
[22] M. J. F. Gales, "Semi-tied covariance matrices for hidden Markov models," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 3, pp. 272–281, May 1999.
[23] "NIST Language Recognition Evaluation," [Online]. Available: http://www. nist.gov/speech/tests/lang/index.htm
[24] W.-Q. Zhang and J. Liu, "An equalized heteroscedastic linear discriminant analysis algorithm," *IEEE Signal Process. Lett.*, vol. 15, pp. 585–588, 2008.

Wei-Qiang Zhang (S'04–M'09) was born in Hebei, China, in 1979. He received the B.S. degree in applied physics from University of Petroleum, Shandong, China, in 2002, the M.S. degree in communication and information systems from Beijing Institute of Technology, Beijing, China, in 2005, and the Ph.D. degree in information and communication engineering from Tsinghua University, Beijing, in 2009.

He is a Research Assistant at the Department of Electronic Engineering, Tsinghua University. His research interests are in the area of speech and signal processing, primarily in parameter estimation, higher order statistics, time–frequency analysis, speaker recognition, and language recognition.

Liang He received the B.S. degree in communication engineering from Civil Aviation University of China, Tianjin, China, in 2004, the M.S. degree in information science and electronic engineering from Zhejiang University, Hangzhou, China, in 2006. He is currently pursuing the Ph.D. degree in the department of electronic engineering, Tsinghua University, Beijing, China.

His research interests are in the area of speech signal processing and information theory, primarily in speaker recognition, language recognition, diversity, and cooperation in wireless communication and modulation/emitter recognition.



Yan Deng received the B.S. degree in communication engineering from National University of Defense Technology, Changsha, China, in 2005. She is currently pursuing the Ph.D. degree in the Department of Electronic Engineering, Tsinghua University, Beijing, China.

Her research focuses upon language recognition and speaker recognition.

Jia Liu (M'97) received the B.S., M.S., and Ph.D. degrees in communication and electronic systems from Tsinghua University, Beijing, China, in 1983, 1986, and 1990, respectively.

He worked at the Remote Sensing Satellite Ground Station, Chinese Academy of Sciences, after the Ph.D. degree and worked as a Royal Society Visiting Scientist at the Cambridge University Engineering Department, Cambridge, U.K., from 1992 to 1994. He is now a Professor in the Department of Electronic Engineering, Tsinghua University. His research fields include speech recognition, speaker recognition, language recognition, expressive speech synthesis, speech coding, and spoken language understanding.

Michael T. Johnson (S'93–M'93–SM'02) received the B.S. degree in computer science engineering and the B.S. degree in engineering, with electrical concentration, from LeTourneau University, Longview, TX, in 1989 and 1990, respectively, the M.S.E.E. degree from the University of Texas, San Antonio, in 1994, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 2000.

He was a Design Engineer and Engineering Manager from 1990 to 1996, and is currently an Associate Professor in the Department of Electrical and Computer Engineering at Marquette University, Milwaukee, WI. His primary research area is speech and signal processing, with interests in machine learning, statistical pattern recognition, and nonlinear signal processing.