Tree-Based Covariance Modeling of Hidden Markov Models

Ye Tian, Jian-Lai Zhou, Hui Lin, and Hui Jiang, *Member, IEEE*

***Abstract*—In this paper, we present a tree-based, full covariance hidden Markov modeling technique for automatic speech recognition applications. A multilayered tree is built first to organize all covariance matrices into a hierarchical structure. Kullback–Leibler divergence is used in the tree-building to measure inter-Gaussian distortion and successive splitting is used to construct the multilayer covariance tree. To cope with the data sparseness problem in estimating a full covariance matrix, we interpolate the diagonal covariance matrix of a leaf-node at the bottom of the tree with the full covariance of its parent and ancestors along the path up to the root node. The interpolation coefficients are estimated in the maximum likelihood sense via the EM algorithm. The interpolation is performed in three different parametric forms: 1) inverse covariance matrix, 2) covariance matrix, and 3) off-diagonal terms of the full covariance matrix. The proposed algorithm is tested in three different databases: 1) the DARPA Resource Management (RM), 2) the Switchboard, and 3) a Chinese dictation. In all three databases, we show that the proposed tree-based full covariance modeling consistently performs better than the baseline diagonal covariance modeling. The algorithm outperforms other covariance modeling techniques, including: 1) the semi-tied covariance modeling (STC), 2) heteroscedastic linear discriminant analysis (HLDA), 3) mixtures of inverse covariance (MIC), and 4) direct full covariance modeling.**

***Index Terms*—Automatic speech recognition, covariance modeling, Gaussian mixture models, tree modeling.**

I. Introduction

GAUSSIAN mixture continuous density hidden Markov model (CDHMM) has become the predominant modeling technique for large-vocabulary continuous speech recognition (LVCSR) in the last decade or two. In CDHMM, typically a large set of Gaussian kernels which are parameterized by the corresponding means and covariance matrices need to be estimated from the available training data. Normally, two choices are taken for the covariance matrix in its parameterization: full or diagonal covariance matrix. The former choice, by all means, is more correct than the latter one because it does not assume null correlations between different feature components as in the diagonal covariance [1]. However, due to the sparseness of training data, most off-diagonal elements in the full covariance matrices cannot be estimated reliably. If full covariance matrices must be used, we may have to greatly limit the total number of Gaussian components in the system. But when a smaller number of Gaussian components is used, the resultant model may not be able to characterize the true data distributions at an acceptable level of precision. On the other hand, the obvious incorrect assumption of uncorrelated components in the diagonal covariance modeling hampers the recognition performance. Obviously, either approach has its own limitation.

To improve the LVCSR performance, different approaches have been proposed to cope with the above-mentioned problems. Essentially, they can be put into two categories: 1) to de-correlate the features, or 2) to model the full covariance in a more data efficient manner. In the first approach, linear transformations has been extensively used to transform the original speech features into new coordinates which are less correlated than the original ones. For example, discrete cosine transformation (DCT) [2], linear discriminant analysis (LDA) [3]–[5], Karhunen–Loeve transform (KLT) [3], heteroscedastic discriminant analysis (HDA) [6], or more recently, maximum likelihood linear transform (MLLT) [7], heteroscedastic linear discriminant analysis (HLDA) [8], [9], etc. Different criterions have been adopted in finding the corresponding linear transformation, including the following:

1) principal component analysis for DCT or KLT;
2) class discrimination maximization for LDA and HDA;
3) maximum likelihood criterion for MLLT and HLDA.

Also, since a single linear transformation is in general not adequate to decorrelate the feature components in every HMM state, multisubspace transformations are used in the semitied covariance (STC) modeling [12] and multiple HLDA (MHLDA) [13] to alleviate this problem.

In the second category, where more data efficient covariance modeling techniques are derived, typical schemes include the block-diagonal covariance models [10], sparse inverse covariance matrices [11], mixtures of inverse covariance (MIC) [14], extended MLLT (EMLLT) [15], and the SPAM models (models with a Subspace constraint on the Precisions And Means) [16]. In MIC, SPAM, and EMLLT, the full inverse covariance, or the precision matrix, is represented as a linear combination of a set of prototype precision matrices. Both the prototype precision matrices and corresponding weights are estimated in the maximum likelihood sense.

The above taxonomy is artificial, and many algorithms actually combine both the decorrelation and full covariance modeling in their modeling procedure.

Manuscript received January 31, 2005; revised September 28, 2005. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Paris Smaragdis.

Y. Tian is with the Speech and Natural Language Division, Microsoft Corporation, Redmond, WA 98052 USA (e-mail: ytian@microsoft.com).

J.-L. Zhou is with the Microsoft Research Asia, Beijing 100080, China (e-mail:jlzhou@microsoft.com).

H. Lin is with the Department of Electronics Engineering, Tsinghua University, Beijing 100084, China (e-mail: linhui99@mails.tsinghua.edu.cn).

H. Jiang is with the Department of Computer Science, York University, Toronto, Ontario M3J 1P3, Canada (e-mail: hj@cs.yorku.edu).

Digital Object Identifier 10.1109/TSA.2005.863210

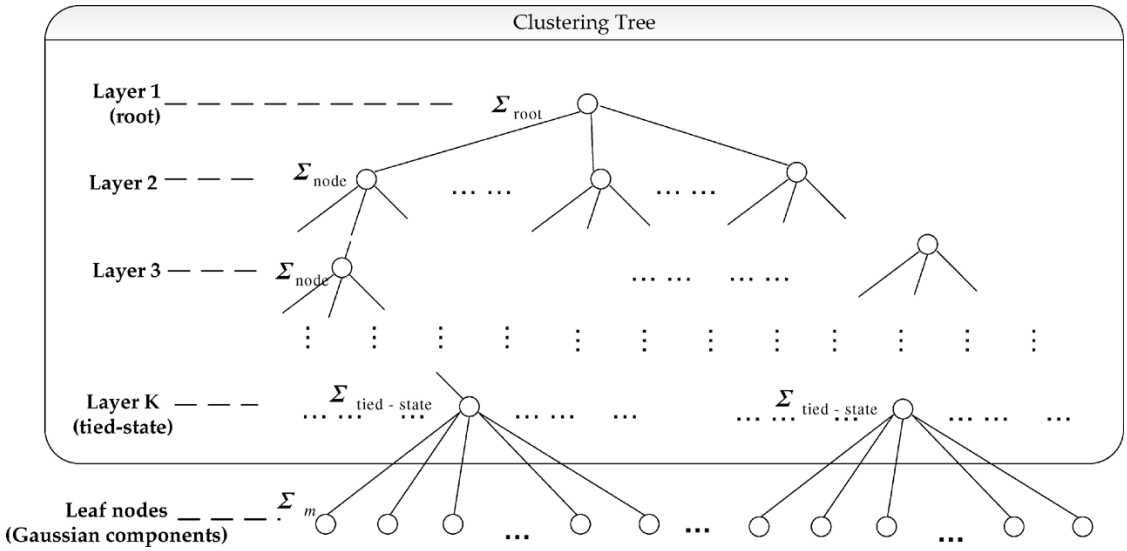


Fig. 1. Tree generated from the covariance matrix clustering.

In this paper, along the line of the linear combination of covariance matrices, we propose a tree-based covariance modeling (TCM), where a tree structure is used to organize all covariance matrices in a CDHMM hierarchically, and different amount data sharing in estimation is built naturally into the tree. The bottom layer, or the leaf nodes, of the tree corresponds to all Gaussians used in the CDHMM, and the covariance matrix of each Gaussian is obtained by linearly combining the diagonal covariance matrix at the leaf node and all the full covariance matrices of its parent and ancestors, all the way up to the root node. The full covariance matrices in the upper layer of the tree are estimated from the training data since more data are available than the leaf-node. The linear interpolation coefficients are also estimated from the data in the maximum likelihood sense.

The TCM provides a flexible framework and the linear interpolation of matrices can be performed in several different ways. The interpolation can be done for either covariance or precision matrices. Alternatively, because the diagonal elements can be reliably estimated, the interpolation can be also conducted only to compensate the off-diagonal elements of the covariance matrix, which represents correlation between different components feature. This approach is called tree-based off-diagonal compensation (TOC), which is experimentally found to yield the best performance. We have evaluated the proposed TCM strategy on the DAPRA resource management database [24], the switchboard database [29], and a Chinese dictation database [26]. Experimental results show that the TOC method can achieve significant error reduction over the best diagonal covariance models, and it also yields better performance than other existing covariance modeling methods, such as STC, HLDA, MIC, and direct full covariance modeling.

The remainder of this paper is organized as follows. In Section II, we present the key steps of our tree-based covariance modeling. In Section III, we present how to build a hierarchical tree-based structure in the covariance space. In Section IV, we present reestimation of the full covariance matrices in the leaf nodes. In Section V, we present the algorithm on estimating the linear interpolation weights. In Section VI, the recognition experimental setups and results are given. Finally, we conclude the paper with a summary of results and discussions.

II. Brief Sketch of the Proposed Method

First a brief sketch of our tree-based covariance modeling is summarized in the following five steps, and detail procedure of each step is elaborated in the next few sections.

1) Train a baseline Gaussian mixture state-tied triphone HMMs with diagonal covariance matrices. We will keep the mixture weights and mean vectors of the baseline model set unchanged but modify the covariance matrices in following tree-building steps.
2) Build a tree of full covariance starting from the root by using all tied-states as base elements in tree-building. After the tree is built, for each tied-state node we attach all Gaussian components associated with it as its children, as shown in Fig. 1.
3) Estimate a full covariance matrix for each non leaf node in the tree from data belonging to all its children.
4) Each Gaussian covariance at a leaf node is linearly interpolated into a full covariance matrix in the maximum likelihood sense. The tree-based prototypes for linear interpolation are the estimated covariance matrices of the nodes along the upward path from the leaf node to the root node.
5) Replace the original diagonal covariance matrices in the model set with the newly interpolated full covariance matrices and then use them for recognition.

If necessary, we can also run the above TCM procedure iteratively. Based on the CDHMM set derived in step 5), we can recollect statistics over the entire training corpus, reestimate the initial CDHMMs, rebuild the hierarchical tree and reestimate the interpolation weights based on the newly constructed tree.

III. Tree-Based Covariance Structure

In this section, we explain how to build a tree-based covariance structure for a set of state-tied triphone CDHMMs. The tree-based structure of sharing HMM parameters is not new

in speech recognition and has been used for the purpose of HMM adaptation for years [18]–[20]. However, most of previous works are focused on building a tree structure to organize the mean vectors in HMMs. In contrast, in this paper, we are interested in how to build a similar tree structure but merely based upon the covariance matrices.

In Fig. 1, we illustrate the tree-based covariance structure. Each leaf node stands for a Gaussian component in the CDHMM set. The layer just above the Gaussian components is the tied-state layer. We use tied-states as base elements in tree-building since the training data for each Gaussian component may not be enough for a reliable estimation of a full covariance matrix. Standard data-driven clustering approach is applied on the tied-states to build a hierarchical tree from the root node. First, we build root node and assign all tied-states to the root. Then, we recursively split each node into different child nodes. After the tree is built, for each tied-state node at the bottom layer we attach all Gaussian components as another layer.

A. Initial Estimation of Gaussians

The HMM state output probability density function (pdf) parameter set $\boldsymbol{\theta}$ is defined as $\boldsymbol{\theta} = \{\boldsymbol{\theta}_i | i = 1, 2, \cdots, N\}$, where $\boldsymbol{\theta}_i$ denotes the ith tied-state output pdf, and it is modeled as a Gaussian mixture model. It is parameterized as

$$f(\boldsymbol{o}|\boldsymbol{\theta}_i) = \sum_{m=1}^{M} \omega_{im} \mathbb{N}(\boldsymbol{o}, \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im})$$

where $\boldsymbol{o}$ is an observation of speech feature vector, ω_{im} is the mixture weight of mth Gaussian component, and $\boldsymbol{\mu}_{im}$ and $\boldsymbol{\Sigma}_{im}$ are the corresponding mean vector and the covariance matrix of the mth Gaussian component, respectively. The pdf of the mth Gaussian component is

$$\mathbb{N}(\boldsymbol{o}, \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}) = \sqrt{\frac{\left|\boldsymbol{\Sigma}_{im}^{-1}\right|}{(2\pi)^D}} \times \exp\left(-\frac{(\boldsymbol{o}-\boldsymbol{\mu}_{im})^{\mathrm{T}} \boldsymbol{\Sigma}_{im}^{-1} (\boldsymbol{o}-\boldsymbol{\mu}_{im})}{2}\right) \quad (1)$$

where D is the dimension of the feature space.

Initially, all of the triphone CDHMM parameters are estimated via the conventional Baum–Welch algorithm with a decision-tree-based, state-tying strategy. The standard estimation formula of the Gaussian mean is

$$\boldsymbol{\mu}_{im} = \frac{\sum_{\tau} \gamma_{im}(\tau) \boldsymbol{o}(\tau)}{\sum_{\tau} \gamma_{im}(\tau)} \quad (2)$$

where $\boldsymbol{o}(\tau)$ is the τth observation vector and $\gamma_{im}(\tau)$, the probability that $\boldsymbol{o}(\tau)$ belongs to the mth Gaussian component in the ith state. Similarly, the corresponding full covariance matrix, $\boldsymbol{\Sigma}_{im}$, is estimated as

$$\boldsymbol{\Sigma}_{im} = \frac{\sum_{\tau} \gamma_{im}(\tau) \left(\boldsymbol{o}(\tau) - \boldsymbol{\mu}_{im}\right) \left(\boldsymbol{o}(\tau) - \boldsymbol{\mu}_{im}\right)^{\mathrm{T}}}{\sum_{\tau} \gamma_{im}(\tau)} \quad (3)$$

where T denotes the transpose of a vector.

However, due to the data sparseness, most of the full covariance matrices of Gaussian components cannot be reliably estimated. We use tied-states as base elements in tree-building, and a Gaussian pdf is used to represent each tied-state. The mean vector $\boldsymbol{\mu}_{\mathrm{tied-state},i}$ and the full covariance matrix $\boldsymbol{\Sigma}_{\mathrm{tied-state},i}$ of the ith tied-state are estimated with the observation vectors associated with the state, i.e.,

$$\boldsymbol{\mu}_{\mathrm{tied-state},i} = \frac{\sum_{\tau} \gamma_i(\tau) \boldsymbol{o}(\tau)}{\sum_{\tau} \gamma_i(\tau)} \quad (4)$$

$$\boldsymbol{\Sigma}_{\mathrm{tied-state},i} = \frac{\sum_{\tau} \gamma_i(\tau) \left(\boldsymbol{o}(\tau) - \boldsymbol{\mu}_{\mathrm{tied-state},i}\right) \left(\boldsymbol{o}(\tau) - \boldsymbol{\mu}_{\mathrm{tied-state},i}\right)^{\mathrm{T}}}{\sum_{\tau} \gamma_i(\tau)} \quad (5)$$

where $\gamma_i(\tau)$ is the probability that $\boldsymbol{o}(\tau)$ belongs to the ith state. The weight of the ith tied-state node is computed as

$$\gamma_{\mathrm{tied-state},i} = \sum_{\tau} \gamma_i(\tau). \quad (6)$$

B. Hierarchal Tree Building

Here we use the standard top-down, data-driven clustering approach on all tied-states in the CDHMM set. Two key components in the clustering algorithm, namely, the distortion measure between any two Gaussians pdf's and the corresponding centroid of a cluster are given as follows.

1) Distortion Measure: We need to first define a distortion measure $d(m,n)$ between any two given Gaussian pdf's, $g_m(\boldsymbol{o}) = \mathbb{N}(\boldsymbol{o}, \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$ and $g_n(\boldsymbol{o}) = \mathbb{N}(\boldsymbol{o}, \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$. Here, the distortion measure between them is calculated as the symmetric Kullback–Leibler divergences between $g_m(\cdot)$ to $g_n(\cdot)$.

$$\begin{aligned} d(m,n) &= \int g_m(\boldsymbol{o}) \log \frac{g_m(\boldsymbol{o})}{g_n(\boldsymbol{o})} d\boldsymbol{o} + \int g_n(\boldsymbol{o}) \log \frac{g_n(\boldsymbol{o})}{g_m(\boldsymbol{o})} d\boldsymbol{o} \\ &= \frac{1}{2} \log \frac{|\boldsymbol{\Sigma}_m|}{|\boldsymbol{\Sigma}_n|} + \frac{1}{2} Tr\left(\boldsymbol{\Sigma}_m^{-1} \boldsymbol{\Sigma}_n\right) \\ &\quad + \frac{1}{2} (\boldsymbol{\mu}_n - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (\boldsymbol{\mu}_n - \boldsymbol{\mu}_m) - \frac{D}{2} \\ &\quad + \frac{1}{2} \log \frac{|\boldsymbol{\Sigma}_n|}{|\boldsymbol{\Sigma}_m|} + \frac{1}{2} Tr\left(\boldsymbol{\Sigma}_n^{-1} \boldsymbol{\Sigma}_m\right) \\ &\quad + \frac{1}{2} (\boldsymbol{\mu}_m - \boldsymbol{\mu}_n)^T \boldsymbol{\Sigma}_n^{-1} (\boldsymbol{\mu}_m - \boldsymbol{\mu}_n) - \frac{D}{2} \end{aligned}$$

TABLE I
OVERVIEW OF CLUSTERING ALGORITHM

set root node index k=1 assign all the tied-states to root node $G(1)=\{\Sigma_{\text{tied-state},i}; i=1....N\}$ calculate the covariance of root node using (8) add root node to tree structure Γ
call function *split*($G(k)$, Γ) to generate the whole tree structure Γ
for i=1 **to** N expand to Gaussian component level and add it to tree structure Γ as another layer **end for**

Since only the covariance clustering is of our primary interest, we ignore the distance associated with the means. By removing parts relevant to the means and constants, the distortion measure between two equal mean Gaussians is then

$$d^*(m,n)=Tr\left(\boldsymbol{\Sigma}_m^{-1}\boldsymbol{\Sigma}_n+\boldsymbol{\Sigma}_n^{-1}\boldsymbol{\Sigma}_m\right). \tag{7}$$

2) New Centroid Gaussian: At each node in a tree structure, if a collection of Gaussians $\{g_1(\boldsymbol{o}),\ldots,g_N(\boldsymbol{o})\}$ is clustered into a group associated with a node k, the centroid is also represented as a Gaussian pdf $\mathbb{N}(\boldsymbol{o},\boldsymbol{\mu}_{\text{node},k},\boldsymbol{\Sigma}_{\text{node},k})$. The mean $\boldsymbol{\mu}_{\text{node},k}$ and covariance $\boldsymbol{\Sigma}_{\text{node},k}$ are estimated by

$$\boldsymbol{\mu}_{\text{node},k}=\frac{\sum_{i=1}^{N}\gamma_{\text{tied-state},i}\boldsymbol{\mu}_{\text{tied-state},i}}{\sum_i\gamma_{\text{tied-state},i}}$$

and the equation at the bottom of the page, where $\gamma_{\text{tied-state},i}$ is the weight of the ith tied-state node defined by (6). $\boldsymbol{W}_{\text{node},k}$, and $\boldsymbol{B}_{\text{node},k}$ are the within-class and between-class covariance, respectively. Again, since only the covariance is of our primary interest, the distortion between the Gaussian means, or the between-class covariance, $\boldsymbol{B}_{\text{node},k}$, is ignored. The new centroid covariance matrix for the cluster is then

$$\boldsymbol{\Sigma}_{\text{node},k}=\boldsymbol{W}_{\text{node},k}=\frac{\sum_{i=1}^{N}\gamma_{\text{tied-state},i}\boldsymbol{\Sigma}_{\text{tied-state},i}}{\sum_i\gamma_{\text{tied-state},i}}. \tag{8}$$

The weight of the new node k is calculated by

$$\gamma_{\text{node},k}=\sum_{i=1}^{N}\gamma_{\text{tied-state},i}.$$

3) Tree Building Algorithm: The clustering algorithm shown in Table I is used to construct a tree structure. The meanings of the parameters are as follows:

1) T_{N}: the number of children nodes of each node in the tree;
2) T_γ: the threshold for splitting;
3) Γ: the tree structure, contains the parent-children relationship of all nodes, the weight and covariance matrix of each node;
4) $G\{\}$: the set of tied-states for splitting;
5) $C\{\}$: the set of current centroids.

First, we build the root node and assign all tied-states to the root. Second, we call recursion function $\text{Split}()$ to build the tree structure and calculate the covariance of each node. Third, for each tied-state node at the bottom layer we attach Gaussian components as another layer.

The overview of function $\text{Split}()$ is shown in Table II. We first choose the tied-states that have largest distance as initial centroids, and repeat the k-mean procedure until the total distance converges. After assigning the tied-states to different centroids, we call $\text{Split}()$ for each centroid to do clustering with new tied-state subset.

In this paper, no attempt has been made to optimize the two thresholds values of T_{N} and T_γ, though various schemes, such as BIC [22] can be used to determine their optimal values.

IV. TREE-BASED COVARIANCE MODELING

After the tree is built, for those nodes in the lower level of the tree, we may have to use a more constrained covariance matrix, like a diagonal or block diagonal matrix, due to the data sparseness problem. However, for the nodes in the upper level of the tree, we will be able to reliably estimate the full covariance matrices since a large amount of training data usually is available. In this tree, each intermediate node represents a cluster of Gaussians of its children and is parameterized by a single full covariance matrix calculated in (8) as a by-product from clustering. For Gaussian components in leaf nodes, full covariance matrices estimated (3) are not reliable in most cases due to data sparseness. However, if we follow a path in the tree upward from the leaf to the root node, we will have a sequence of full covariance matrices which can be used to improve the covariance estimation of the leaf node. Following the idea of linear combination in [14], for each Gaussian in leaf node we can linearly interpolate all covariance (or inverse covariance) matrices along the upward path to root to estimate the full covariance (or inverse covariance) matrix for current node. This approach is named as *Tree-based Covariance Modeling* (TCM). In contrast to MIC and EMLLT in [14] and [15] that use global prototypes which are shared among all the Gaussian components, we use different prototypes for different Gaussian components in the TCM.

For the Gaussian component in the mth leaf node, we denote the set of all intermediate nodes along the upward path from this node to the root as

$$\Psi(m)=\{m\text{'s parent and all its ancestors}\}. \tag{9}$$

$$\begin{aligned}\boldsymbol{\Sigma}_{\text{node},k}&=\boldsymbol{W}_{\text{node},k}+\boldsymbol{B}_{\text{node},k}\\&=\frac{\sum_{i=1}^{N}\gamma_{\text{tied-state},i}\boldsymbol{\Sigma}_{\text{tied-state},i}}{\sum_i\gamma_{\text{tied-state},i}}+\frac{\sum_{i=1}^{N}\gamma_{\text{tied-state},i}(\boldsymbol{\mu}_{\text{tied-state},i}-\boldsymbol{\mu}_{\text{node},k})(\boldsymbol{\mu}_{\text{tied-state},i}-\boldsymbol{\mu}_{\text{node},k})^{\text{T}}}{\sum_i\gamma_{\text{tied-state},i}}\end{aligned}$$

TABLE II
OVERVIEW OF RECURSIVE FUNCTION SPLIT () FOR CLUSTERING

FUNCTION Split(tied-state set $G\{\}$,tree structure Γ)
begin function
if number$\{G\{\}\} \le T_N$ or $\gamma_{\text{node},k} <= T_\gamma$
exit function;
end if
select T_N tied-states which are farthest apart as the centroids:
for i=1 **to** number$\{G\{\}\}$
for j=1 **to** number$\{G\{\}\}$
distance[i][j] = $d^*[G\{i\}, G\{j\}]$
end for
end for
set the children centroid $C\{\} = \{\}$
while (number$\{C\{\}\} < T_N$)
select maximum value from distance[i][j] and defined as distance[i_0][j_0]
if $i_0 \notin C\{\}$ and $j_0 \notin C\{\}$ **then**
$C\{\} = C\{\} + \{i_0, j_0\}$
end if
remove distance[i_0][j_0] from distance list
end while
Do
for m=1 **to** number$\{G\{\}\}$
assign the m-th tied-state to one of the above centroids
$I(m) = \arg\min_{1 \le i \le T_N} d^*[m, C\{i\}]$,
total distance += $d^*[m, C\{I(m)\}]$.
end for
update new centroid covariance for $C\{\}$ by (8)
while (total distance converges);
for i=1 **to** number$\{C\{\}\}$
add the centeriod $C\{i\}$ to tree structure Γ
set $G'\{\} = \{\}$
for m=1 **to** number$\{G\{\}\}$
if $I(m) = i$ **then** $G'\{\} = G'\{\} + G\{i\}$
end for
call *split*($G'\{\}, \Gamma$);
end for
end function

TCM provides a flexible framework for full covariance modeling. The linear interpolation can be performed in several slightly different ways, which results in following three different covariance modeling schemes.

A. Tree-Based Mixture of Inverse Covariance Modeling (TMIC)

As demonstrated in [14], the interpolation can be done in terms of the inverse covariance matrix, also known as the precision matrix. The interpolation of inverse covariance matrices results in faster calculation in Gaussian likelihood function since the matrix inversion can be avoided. The estimated inverse full covariance $\hat{\mathbf{\Sigma}}_m^{-1}$ for the Gaussian component m is defined as

$$\hat{\mathbf{\Sigma}}_m^{-1} = \lambda_{m,0}\text{diag}(\mathbf{\Sigma}_m)^{-1} + \sum_{k \in \Psi(m)} \lambda_{m,k}\mathbf{\Sigma}_{\text{node},k}^{-1} \qquad (10)$$

where $\text{diag}(\mathbf{\Sigma}_m)$ denotes the diagonal matrix of $\mathbf{\Sigma}_m$, $\lambda_{m,0}$, and $\{\lambda_{m,k}\}$ denote all the combination weights for the Gaussian m, and $\Psi(m)$ is defined in (9), and $\{\mathbf{\Sigma}_{\text{node},k}\}$ is the full covariance matrices of node k and is calculated in (8). This method is called tree-based mixture of inverse covariance modeling (TMIC) in this paper.

B. Tree-Based Mixture of Covariance Modeling (TMC)

Alternatively, the linear matrix interpolation can be directly done in the domain of covariance matrix. In this case, a full covariance matrix $\hat{\mathbf{\Sigma}}_m$ is calculated for the mth Gaussian component as

$$\hat{\mathbf{\Sigma}}_m = \lambda_{m,0}\text{diag}(\mathbf{\Sigma}_m) + \sum_{k \in \Psi(m)} \lambda_{m,k}\mathbf{\Sigma}_{\text{node},k} \qquad (11)$$

where $\lambda_{m,0}$ and $\{\lambda_{m,k}\}$ denote another set of combination weights. This method is named as tree-based mixture of covariance modeling (TMC) hereafter.

C. Tree-Based Off-Diagonal Compensation (TOC)

In a leaf node, the covariance matrix consists of two parts of information: the diagonal or variance information and the off-diagonal components representing the correlation among different feature dimensions. Generally speaking, the diagonal components can be reliably estimated from a relatively small amount of data, but the off-diagonal or the correlation information cannot be reliably estimated without a large amount of data. This observation motivates us to keep the diagonal elements intact as initially estimated and use the tree-based interpolation to compensate only the correlation information of all off-diagonals components. We shall name this method tree-based off-diagonal compensation (TOC).

In TOC, for each Gaussian component in a leaf node, the off-diagonal components of covariance matrices of all the nodes along the upward path from this leaf node to the root are used to compensate the off-diagonal components of covariance matrix based on the linear combination strategy. In TOC, the full covariance $\hat{\mathbf{\Sigma}}_m$ of Gaussian component m is estimated as

$$\hat{\mathbf{\Sigma}}_m = \text{diag}(\mathbf{\Sigma}_m) + \sum_{k \in \Psi(m)} \lambda_{m,k}\left[\mathbf{\Sigma}_{\text{node},k} - \text{diag}(\mathbf{\Sigma}_{\text{node},k})\right] \qquad (12)$$

where the diagonal elements $\text{diag}(\mathbf{\Sigma}_m)$ are kept intact in the interpolation, and the off-diagonal terms are interpolated by $\{\lambda_{m,k}\}$ in TOC.

D. Tree-Based Inverse Covariance Off-Diagonal Compensation (TIOC)

Because interpolation in inverse covariance space has a computational advantage during decoding, we can also extend the off-diagonal compensation to the inverse covariance space. We keep the diagonal of inverse covariance unchanged and compensate only the off-diagonal of inverse covariance. We name this method tree-based inverse covariance off-diagonal compensation (TIOC).

In TIOC, for each Gaussian component in a leaf node, the off-diagonal components of inverse covariance matrices of all the nodes along the upward path from this leaf node to the root are used to compensate the off-diagonal components of inverse covariance matrix based on the linear combination strategy. The full inverse covariance $\hat{\boldsymbol{\Sigma}}_m^{-1}$ of Gaussian component m is estimated as

$$\hat{\boldsymbol{\Sigma}}_m^{-1} = \mathrm{diag}\left(\boldsymbol{\Sigma}_m^{-1}\right) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1} - \mathrm{diag}\left(\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}\right)\right] \tag{13}$$

where the diagonal elements $\mathrm{diag}(\boldsymbol{\Sigma}_m^{-1})$ are kept intact in the interpolation and the off-diagonal terms are interpolated by $\{\lambda_{m,k}\}$ in TIOC. Note that $\boldsymbol{\Sigma}_m$ may be singular, in this case we use $[\mathrm{diag}(\boldsymbol{\Sigma}_m)]^{-1}$ instead of $\mathrm{diag}(\boldsymbol{\Sigma}_m^{-1})$ in TIOC.

V. Maximum Likelihood Estimation of Interpolation Weights

No matter which TCM modeling scheme is used, we need to estimate the interpolation weights for all Gaussian components to derive the full covariance matrices. In this paper, we adopt the maximum likelihood estimation (MLE) method for this purpose, as in [14]. For simplicity, we assume all covariance matrices attached in tree nodes are kept unchanged once the tree is built. We further assume that all mean vectors of the CDHMM set are initially estimated as in (2) and remain unchanged. Under these assumptions, the likelihood function of the CDHMM covariance matrices can be cast as a function of all unknown interpolation weights. All unknown interpolation weights are estimated to maximize the likelihood function of the whole training set. However, due to the hidden parameter problem in the underlying CDHMMs, this maximum likelihood estimation problem cannot be easily solved in a direct way. Thus, we adopt to use the generalized expectation–maximization (GEM) algorithm to derive maximum likelihood estimation of the unknown weights iteratively. Let's denote the interpolation weights for the mth Gaussian as $\boldsymbol{\lambda}_m = \{\lambda_{m,k} | k \in \Psi(m)\}$ and all interpolation weights for the whole model set as $\boldsymbol{\Lambda} = \{\boldsymbol{\lambda}_m | m \text{ is Gaussian in the model set}\}$.

A. E-Step

As in [14], the auxiliary function can be written as

$$Q(\boldsymbol{\Lambda};\hat{\boldsymbol{\Lambda}}) = \sum_{m=1}^{M} \gamma_m \left[\log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right| - \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right)\right] \tag{14}$$

where $\boldsymbol{\Lambda}$ and $\hat{\boldsymbol{\Lambda}}$ are the old and new estimated model, respectively, M the total number of Gaussian components in the CDHMM set, γ_m is the Gaussian occupation, $\boldsymbol{\Sigma}_m$ is the initial full covariance matrix estimated in (3), and $\hat{\boldsymbol{\Sigma}}_m$ is the new full covariance matrix to be estimated for the mth component by using one of the above tree-based covariance modeling schemes, i.e., (10) for TMIC, (11) for TMC, or (12) for TOC. Obviously, the new covariance matrix $\hat{\boldsymbol{\Sigma}}_m$ is a function of the interpolation weights of the mth Gaussian component, i.e., $\boldsymbol{\lambda}_m$. Thus, the auxiliary function $Q(\cdot)$ is a function of all interpolation weights $\boldsymbol{\Lambda}$.

For the mth Gaussian component, the interpolation weights $\boldsymbol{\lambda}_m$ are independent from those of other Gaussian components. Hence, the optimization problem of the whole model in (14) can be equivalently decomposed into M small optimization problems

$$\begin{aligned} Q(\boldsymbol{\Lambda};\hat{\boldsymbol{\Lambda}}) &= \sum_{m=1}^{M} \gamma_m \cdot \left[\log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right| - \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right)\right] \\ &= \sum_{m=1}^{M} \gamma_m \cdot Q_m(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m) \end{aligned}$$

where

$$Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m) = \log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right| - \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right). \tag{15}$$

Therefore, for $m = 1, 2 \ldots, M$, we optimize the subauxiliary function $Q(\boldsymbol{\lambda}_m)$ to derive the interpolation weights $\boldsymbol{\lambda}_m$ for Gaussian m.

B. M-Step

The unknown interpolation weights are updated to maximize the auxiliary function. In other words, for $m = 1, 2, \ldots, \mathrm{M}$, λ_m is updated as

$$\begin{aligned} \boldsymbol{\lambda}_m' &= \arg\max_{\hat{\boldsymbol{\lambda}}_m} Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m) \\ &= \arg\max_{\hat{\boldsymbol{\lambda}}_m} \left[\log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right| - \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\cdot\boldsymbol{\Sigma}_m\right)\right]. \end{aligned}$$

The maximization is conducted subject to the constraint that the interpolated full matrix $\hat{\boldsymbol{\Sigma}}_m$ must be a valid covariance matrix in a Gaussian distribution, i.e., positive definite. Obviously, the above constrained maximization problem cannot easily solved in a closed-form but an iterative solution. In this paper, we use the *quasi-Newton* method [23] to optimize $\boldsymbol{\lambda}_m$. The quasi-Newton method gradually builds up an approximate Hessian matrix by using gradient information from some or all of the previous iterations. In the nth step, the weights λ_m are updated based on the gradient of the subauxiliary function $Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m^{(n)})$

$$\hat{\boldsymbol{\lambda}}_m^{(n+1)} = \hat{\boldsymbol{\lambda}}_m^{(n)} - \left(\boldsymbol{H}^{(n+1)}\right)^{-1} \cdot \frac{\partial Q\left(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m^{(n)}\right)}{\partial\lambda_m}$$

where $\boldsymbol{H}^{(n+1)}$ is the Hessian matrix calculated by

$$\boldsymbol{H}^{(n+1)} = \boldsymbol{H}^{(n)} + \frac{\boldsymbol{g}^{(n)}\boldsymbol{g}^{(n)\mathrm{T}}}{\boldsymbol{g}^{(n)\mathrm{T}}\boldsymbol{s}^{(n)}} - \frac{\boldsymbol{s}^{(n)\mathrm{T}}\boldsymbol{H}^{(n)\mathrm{T}}\boldsymbol{H}^{(n)}\boldsymbol{s}^{(n)}}{\boldsymbol{s}^{(n)\mathrm{T}}\boldsymbol{H}^{(n)}\boldsymbol{s}^{(n)}}$$

where

$$\boldsymbol{s}^{(n)} = (\partial Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m^{(n+1)})/\partial\boldsymbol{\lambda}_m) - (\partial Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m^{(n)})/\partial\boldsymbol{\lambda}_m)$$

and $\boldsymbol{g}^{(n)} = \hat{\boldsymbol{\lambda}}_m^{(n+1)} - \hat{\boldsymbol{\lambda}}_m^{(n)}$.

The gradient of the objective function $Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m^{(n)})$ of three covariance modeling scheme is given in Appendix I. We can guarantee that the subauxiliary function $Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m)$ is monotonically nondecreasing from one iteration to the next. There-

fore, the above optimization process, known as generalized EM (GEM) algorithm, will eventually converge to a local maximum of $Q(\boldsymbol{\lambda}_m; \hat{\boldsymbol{\lambda}}_m)$.

C. Constraint of Interpolation Weights

Note that unlike the case of mixtures of densities, the weights are not constrained to sum to one or even to be positive. A valid covariance matrix must be symmetric and positive definite. To make it invertible, all eigen values need to be positive. Thus, we constrain the minimum eigen value of the interpolated matrix to be positive in the optimization process, i.e.,

$$\min\left\{\text{Eigenvalues}(\hat{\boldsymbol{\Sigma}}_m)\right\} > 0. \tag{16}$$

VI. Experiments

The initial investigation of the proposed algorithm is evaluated on medium-to-large vocabulary speech recognition tasks. Our algorithms are compared with standard HLDA, STC and MIC in three different database: 1) the DARPA Resource Management, 2) the Switchboard minitrain, and 3) a Chinese dictation.

A. Resource Management Database

1) Experiment Setup: In this part of experiments, a standard medium-size vocabulary (994 words) speech recognition task in the DARPA Resource Management (RM) [24] database, is tested. Speaker-independent HMMs were trained using the NIST/RM SI-109 training set consisting of 3990 utterances from 109 native American talkers (31 females and 78 males), each providing 30 or 40 utterances. Among these, 11 files on the CD-ROM have a long series of identical time-domain samples were removed. The speech signal is sampled at 16 kHz and the analysis frames are 25 ms wide and shifted every 10-ms overlap. A 39-dimensional feature vector, including 12 mel-scale frequency cepstral coefficients (MFCC), log energy, and their first- and second-order time differences, was extracted.

We build our training system with the hidden Markov model toolkit (HTK) [25]. The training begins with a flat start single Gaussian mixture component monophone system. The total number of monophone is 47. After four iterations of Baum–Welch training, the monophone models are cloned to produce a single mixture component triphone system. The word-internal triphone models are used. These initial triphone models are trained with two iterations of embedded training after which a decision-tree clustering is applied to produce a tied state triphone system. The baseline HMM system is produced by standard iterative mixture splitting using four iterations of embedded training after each mixture increase. Finally, a mixture of six Gaussian mixture components with diagonal covariance matrices is trained for each tied-state as the baseline model. This gave a total of about 1600 tied states and 9600 distinct Gaussian densities.

A total of 1199 sentences {feb89, oct89, feb91, sep92} (one sentence corrupted with long series of identical time-domain values is removed) is used for evaluation. The word error rate (WER) of the baseline system is 4.09%.

TABLE III
Comparison of WER Between Nontree-Based Prototypes and Tree-Based Prototypes on RM Database

Covariance estimation	global prototypes	Tree-based prototypes
TMIC: $\hat{\boldsymbol{\Sigma}}_m^{-1} = \sum_{k\in\Psi(m)} \lambda_{m,k}\boldsymbol{\Sigma}_{\text{node},k}^{-1}$	3.49%	3.43%
TMC: $\hat{\boldsymbol{\Sigma}}_m = \sum_{k\in\Psi(m)} \lambda_{m,k}\boldsymbol{\Sigma}_{\text{node},k}$	3.70%	3.60%
TOC: $\hat{\boldsymbol{\Sigma}}_m = \text{diag}(\boldsymbol{\Sigma}_m) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\boldsymbol{\Sigma}_{\text{node},k} - \text{diag}(\boldsymbol{\Sigma}_{\text{node},k})\right]$	3.39%	3.22%
TIOC: $\hat{\boldsymbol{\Sigma}}_m^{-1} = \text{diag}(\boldsymbol{\Sigma}_m^{-1}) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\boldsymbol{\Sigma}_{\text{node},k}^{-1} - \text{diag}(\boldsymbol{\Sigma}_{\text{node},k}^{-1})\right]$	3.50%	3.27%

The baseline is 6 mixture diagonal covariance systems with 4.09% WER. Global prototype is nontree-based prototypes that Gaussian components, and tree-based prototypes are different for each Gaussian component.

2) Tree-Based Prototypes: In this experiment, we compare the performance between global prototypes and our proposed tree-based prototypes. As for the nontree-based global prototypes, we first build a fixed number of prototype matrices which are shared among all Gaussian components, as in MIC [14], SPAM [16], and others. The results are shown in Table III. The first column lists the covariance estimation equation. The second and the third column are word error rate of linear combination for global prototypes and tree-based prototypes, respectively. The number of global prototype is 39. Note that if we do linear combination in inverse covariance space and use global prototypes, it is equivalent to the MIC [14] algorithm.

By comparing the results in the second and the third column for each row, we can see that no matter which linear combination algorithm is used, the tree-based prototypes yields better performance than the global prototypes. These results demonstrate that our tree-based modeling approach is effective.

By comparing the results in the second and the third rows, we can see that linear combination in covariance space leads to a higher WER, which means the linear combination with inverse covariance prototypes is more preferable. However, if we only compensate the off-diagonal parameters of the full covariance matrix, or the TOC interpolation, the performance is even much better than TMIC. This indicates that we should not change the diagonal variances, which are already reliably estimated, in the covariance interpolation process. This partial compensation takes advantage of robustly estimated diagonal parameters and compensates only the off-diagonal terms and yields the best performance. Although TIOC has computation complexity advantage, its performance is not as good as that of TOC. The possible explanation is that in (13) we need to calculate $\text{diag}(\boldsymbol{\Sigma}_m^{-1})$, and when mixture number is high $\boldsymbol{\Sigma}_m$ does not have sufficient train data for a reliable estimation, in this case $\text{diag}(\boldsymbol{\Sigma}_m^{-1})$ is a good estimation as $\text{diag}(\boldsymbol{\Sigma}_m^{-1})$.

3) $T_{\text{childnodes}}$ Selection for Tree Growing: In the tree-building algorithm, each node is spitted into T_N child nodes until the weight of the node smaller than T_γ. Our experiments show that the recognition results are less sensitive to T_γ than to T_N. The results of different T_N are shown in Table IV.

TABLE IV
COMPARISON OF WER BETWEEN TOC WITH DIFFERENT CHILDNODES THRESHOLDS ON RM DATABASE

Number of child nodes	Number of tree nodes	Word Error rate
T_N =10	1754	3.50%
T_N =5	2202	3.23%
T_N =3	2561	3.22%

TABLE V
COMPARISON OF WER AMONG STC, HLDA, MIC, AND TOC ON RM DATABASE

		WER	WER reduction
Baseline		4.09%	--
STC	1 global transformation	3.53%	13.7%
	143 transformations	3.33%	18.6%
HLDA	1 global transformation	3.74%	8.56%
	143 transformations	3.51%	14.2%
MIC (39 prototypes)		3.49%	14.7%
TOC (T_N =3)		3.22%	21.27%

The HLDA system is from feature dimension 39 to dimension 30.

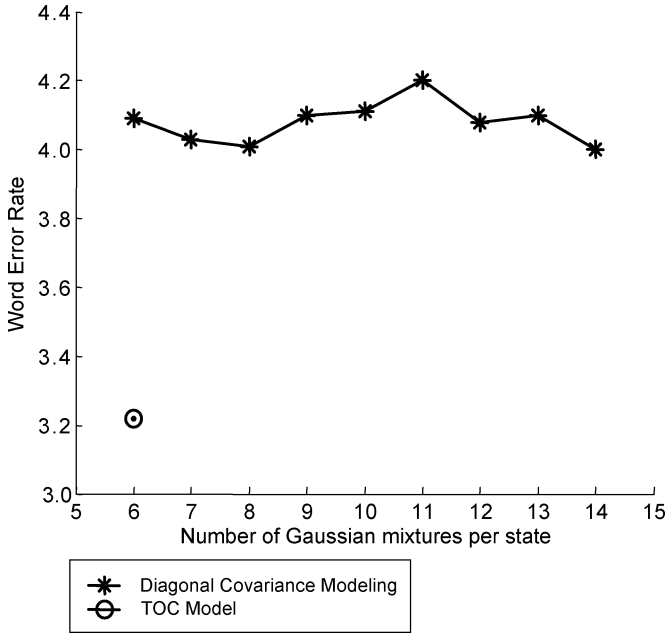


Fig. 2. Performance comparison between TOC and diagonal models with increasing number of mixtures on RM database.

From the table, there is a big recognition performance improvement from $T_N = 10$ to $T_N = 5$, but no significant WER difference for $T_N \leq 5$.

4) Compared With Other Related Techniques: We also compare our TOC modeling scheme with other existing covariance modeling techniques, such as STC, HLDA, and MIC. The results are shown in Table V. Both STC and HLDA are evaluated with one global transformation and multiple transformations. In multiple linear transformations STC and HLDA, the assignment of component to transform class is determined by which monophone state the component belongs to. The transformations of STC and HLDA are estimated directly using the nonlinear optimization scheme. For HLDA, we project the original 39 dimension features to 30 dimension features, and this projection reduce the number of model parameter by approximately 25%.

From Table V, we can see that all the covariance modeling techniques can achieve better performance than the baseline diagonal covariance model set, and among them, TOC is the best. We conjecture that by separating the full covariance modeling into diagonal and nondiagonal terms, best balance between training data efficiency and estimation robustness is achieved in the TOC method.

5) Compared With Standard Diagonal Model With More Mixture: In Fig. 2, we first show the recognition performance (in WER) of several diagonal models as a function of their number of mixture components in each tied-state. We increase the mixture number from six in the baseline model set up to 12. When the number of mixture components reaches eight (the error rate is 4.00%), the WER reduction compared with the baseline is only 1.96%. This shows that no significant improvements can be obtained by simply increasing the number of diagonal Gaussians. We also plot the TOC's performance in the figure as a reference point. It is clear that from the figure the TOC yields much better performance than all diagonal models.

It is worth to mention that the six mixtures full covariance matrices cannot be directly trained from data using (3) because the data is insufficient to reliably estimate large amount of parameters and the full covariance matrices are singular.

B. Switchboard Minitrain Database

1) Experiment Setup: In this part of experiments, the Switchboard minitrain database is used as our recognition task, which contains 23 hours training data. The speech signal is 8-kHz telephone data and the analysis frame is 25-ms long and shifted every 10 ms. For each frame, a 39-dimensional PLP feature vector is extracted, which consists of a 12 PLP static features, log energy, and their first and second order time difference. To reduce the effect of speaker variation on speech recognition results, mean and variance normalization is applied.

Position-dependent phone-set is used. The phone positions are defined as the beginning, the middle, and the ending of the word, or a single word phone. The total number of monophone is 170. The baseline is a decision-tree clustered tied state triphone HMM system. The training begins with a flat start single Gaussian mixture component, monophone system. The monophone models are cloned to produce a single mixture component triphone system. These initial triphone models are trained with four iterations of embedded training after which a decision-tree clustering is applied to produce a tied-state triphone system. The baseline HMM system is produced by standard iterative mixture splitting using four iterations of embedded training after each mixture increase. Finally, 12 Gaussian mixture components with diagonal covariance matrices are trained as the baseline model for each tied-state. This gave a total of 1978 tied states and about 23 k distinct Gaussian densities.

NIST's Hub-5 2000 conversational telephone speech corpus (Eval2000) is used as the testing corpus. The data comes from

TABLE VI
COMPARISON OF WER BETWEEN NONTREE-BASED PROTOTYPES AND TREE-BASED PROTOTYPES ON SWITCHBOARD DATABASE

Covariance estimation	global prototypes	Tree-based prototypes
TMIC: $\hat{\Sigma}_m^{-1} = \sum_{k\in\Psi(m)} \lambda_{m,k}\Sigma_{\text{node},k}^{-1}$	36.50%	36.20%
TMC: $\hat{\Sigma}_m = \sum_{k\in\Psi(m)} \lambda_{m,k}\Sigma_{\text{node},k}$	37.00%	36.60%
TOC: $\hat{\Sigma}_m = \text{diag}(\Sigma_m) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\Sigma_{\text{node},k} - \text{diag}(\Sigma_{\text{node},k})\right]$	36.20%	35.90%
TIOC: $\hat{\Sigma}_m^{-1} = \text{diag}(\Sigma_m^{-1}) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\Sigma_{\text{node},k}^{-1} - \text{diag}(\Sigma_{\text{node},k}^{-1})\right]$	36.40%	36.10%

The baseline is 12 mixture diagonal covariance systems with 38.00% WER. Global prototype is nontree-based prototypes that are shared by all Gaussian components, and tree-based prototypes are different for each Gaussian component

switchboard-like conversations. The decoding dictionary contains more than 20 K words, and an trigram language model is used with the Hapivite [28] decoder for recognition.

2) Tree-Based Prototypes: Like the RM database, we compare the performance between global prototypes and our newly proposed tree-based prototypes on Switchboard minitrain database. The results are shown in Table VI.

We can draw similar conclusion as we did in Table III: the tree-based prototypes perform better than the global prototypes. TOC performs the best. However, the relative improvement is smaller than that in the RM database. This may due to the fact that the test database is a spontaneous speech database and more versatile modeling is needed than just the covariance modeling.

3) Compared With Other Related Techniques: We compared TOC with other covariance modeling techniques, such as HLDA and STC, and MIC. The results are shown in Table VII.

From the table, we can also see that among all these techniques, the TOC still gives the best performance. Compared to the 38.00% baseline WER, MIC can achieve 36.50% WER, and it has 3.94% relative error reduction. Our TOC can achieve 35.90% WER, and it has 5.53% relative error reduction, which is higher than STC, HLDA, and MIC. Again, the spontaneous nature of the database hampers the effectiveness of all covariance modeling technique.

C. Chinese Dictation Task

1) Experiment Setup: In this section, the above TMC modeling scheme is evaluated on a Chinese dictation task. A total of 49 k sentences from 250 speakers (totally 75 h) recorded in a clean environment was used for training. The baseline system is the MSRA Mandarin Speech Toolbox [26]. It is a gender-independent, cross-word triphone mixture Gaussian tied-state HMM system. In this model set, all the speech models have three emitting states, left-to-right topology. The silence model was a fully connected three emitting state model used to represent longer periods of silence. The speech signal is 16 kHz sampled and the analysis frames are 25-ms wide with a 10-ms

TABLE VII
COMPARISON OF WER AMONG STC, HLDA, MIC, AND TOC ON SWITCHBOARD MINITRAIN DATABASE

	WER	WER reduction
Baseline	38.00%	--
STC	36.60%	3.68%
HLDA	37.20%	2.10%
MIC (39 prototypes)	36.50%	3.94%
TOC (T_N =3)	35.90%	5.53%

The HLDA is from feature dimension 39 to dimension 30.

shift. For each frame, a 39-dimensional feature vector, including 12-dimensional cepstral coefficients, log energy, and their first- and second-order time differences, was extracted. The baseline system training begins with a flat single Gaussian mixture component monophone system. The total number of monophone is 97. A decision-tree clustering is used to create 5114 speech tied-states. The baseline HMM system is produced by standard iterative mixture splitting using four iterations of embedded training per mixture configuration. Finally, 36 Gaussian mixture components with diagonal covariance matrices are used as the baseline model. This gives a total of 5114 tied states and 184 k mixture components.

A total of 500 sentences from 25 speakers are used for free syllable loop. The speakers are different with those in the training set and no speaker adaptation was performed here. The error rate of Chinese character of the baseline system is 19.54%.

2) Tree-Based Prototypes: We compare the performance between nontree-based global prototypes and our newly proposed tree-based prototypes on the Chinese dictation database. The results are shown in Table VIII.

It is shown again here that for all combination algorithms tried, the tree-based prototypes are always better than the nontree-based global prototypes. Also, TOC yields the best performance.

3) Compared With Other Related Techniques: We compare the TOC with other existing covariance modeling techniques, such as HLDA and STC. The results are shown in Table IX. Different with that in RM database, for the best results of multiple linear transformations STC and HLDA, the assignment of component to transform class was determined by which tied-state the component belongs to. If Gaussian component belongs to different tied-state, they have different STC and HLDA linear transformation matrices for decoding likelihood calculation. For our system, we have 5114 tied-states, so the total number of linear transformation is 5114. We use more classes on this database because more data is available for reliably estimating more STC and HLDA linear transformation matrices. From the table, we can also see that among all these techniques, the TOC still gives the best performance.

4) Compared With Diagonal Model With More Mixtures: In Fig. 3, we show the recognition performance (in SER) of several diagonal model sets as a function of their mixture numbers in each tied-state. It shows that it cannot significantly improve the performance over our baseline model by simply increasing the number of diagonal Gaussian mixtures. The TOC yields much

TABLE VIII
COMPARISON OF SYLLABLE ERROR RATE (SER) BETWEEN NONTREE-BASED PROTOTYPES AND TREE-BASED PROTOTYPES ON CHINESE DICTATION DATABASE

Covariance estimation	global prototypes	Tree-based prototypes
TMIC: $\hat{\Sigma}_m^{-1} = \sum_{k\in\Psi(m)} \lambda_{m,k} \Sigma_{\text{node},k}^{-1}$	17.91%	17.78%
TMC: $\hat{\Sigma}_m = \sum_{k\in\Psi(m)} \lambda_{m,k} \Sigma_{\text{node},k}$	18.01%	17.90%
TOC: $\hat{\Sigma}_m = \text{diag}(\Sigma_m) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\Sigma_{\text{node},k} - \text{diag}(\Sigma_{\text{node},k})\right]$	17.74%	16.95%
TIOC: $\hat{\Sigma}_m^{-1} = \text{diag}(\Sigma_m^{-1}) + \sum_{k\in\Psi(m)} \lambda_{m,k}\left[\Sigma_{\text{node},k}^{-1} - \text{diag}(\Sigma_{\text{node},k}^{-1})\right]$	17.80%	17.62%

The baseline is 36 mixture diagonal covariance systems with 19.54% SER. Global prototype is nontree-based prototypes that are shared by all Gaussian components, and tree-based prototypes are different for each Gaussian component.

TABLE IX
COMPARISON OF SER AMONG STC, HLDA, MIC, AND TOC ON CHINESE DATABASE

		SER	SER reduction
Baseline		19.54%	--
STC	1 global transformation	18.01%	7.83%
	5114 transformations	17.32%	11.36%
HLDA	1 global transformation	18.20%	6.86%
	5114 transformations	17.71%	9.37%
MIC (39 prototypes)		17.91%	8.34%
TOC (T_N=3)		16.95%	13.25%

The HLDA system is from feature dimension 39 to dimension 30.

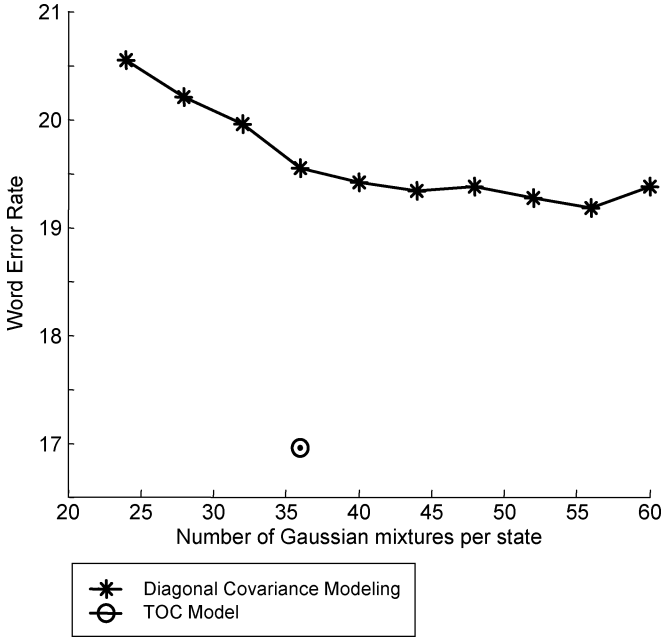


Fig. 3. Comparison between TOC and diagonal models with increasing number of mixtures on Chinese database.

TABLE X
COMPARISON OF SER WITH STANDARD FULL COVARIANCE SYSTEM

	SER	SER reduction
Baseline	19.54%	--
Full covariance system with 3 mixtures (direct training)	18.15%	7.12%
Full covariance system with 36 mixtures (diagonal backup)	17.62%	9.83%
TOC with 36 mixtures	16.95%	13.25%

better performance than the diagonal model set with even larger number of mixtures.

5) Compared With Standard Full Covariance Model: In many cases, we have a choice to directly train a full covariance matrix system. In these cases, we usually cannot estimate a full covariance matrix system with the same number of Gaussians as a diagonal covariance system. With the increased mixture number, the training samples that belong to each mixture component will decrease. If the training sample is not enough, the full covariance estimated with (3) cannot be reliably estimated, or even singular. If full covariance matrices must be used, we have to limit the total number of Gaussian components in the system. However, when a smaller number of Gaussian components is used, the resultant model may not be able to characterize the true data distributions at an acceptable level of precision. One possible solution is to mix full covariance matrices with diagonal covariance matrices; for those components with inadequate training samples, we resort to diagonal covariance matrices and for those components with sufficient training data, we use full covariance matrices.

Comparing to the directly trained full covariance system, our tree-based covariance modeling provides a framework to reliably estimate full covariance models which can have the same level of mixture numbers as the standard diagonal covariance models. If train data is enough for training a full covariance system of certain mixture number per tied-state, it will be enough for training a diagonal covariance system with much higher mixture number. Our TOC method starts with the diagonal covariance system and only compensates the off-diagonal system, and it will generate a full covariance system which has the same number of Gaussian mixtures as that of diagonal covariance baseline.

We compare our TOC approach with the directly trained full covariance model on Chinese dictation task. The results are shown in Table X. Using the training data, we can train a full covariance system with three mixtures per state. We cannot directly train a full covariance system with four mixture components because a large number of full covariance matrices estimated from (3) becomes singular. Using the mixture of full and diagonal covariance, i.e., and backing up to diagonal covariance if the estimated full covariance is singular, we can increase the mixture component up to 36. From the results, we can see that our TOC clearly outperforms the direct full covariance modeling method.

D. Computational Complexity

The strong benefit of STC, HLDA, and MIC is that their computation complexity during decoding is close to that of diagonal covariance system. Although our tree-based covariance modeling approaches perform better than other covariance modeling

TABLE XI
SUMMARY OF WORD ERROR REDUCTION BETWEEN TOC AND THE DIAGONAL COVARIANCE BASELINE

Database	Training data	Mixture number per states	Total number of Gaussians	language model	Relative WER reduction
RM	3.8 hours	6	10K	word-pair	21.27%
Switchboard minitrain	23 hours	12	23K	trigram	5.53%
Chinese dictation	75 hours	36	184K	word-loop	13.25%

techniques, the computation cost during decoding is higher. In this section, we discuss the computation complexity problem.

Here, we list the computation cost for different approaches. We denote feature dimension as D, the tied-state number as S, the number of mixture components of each tied-state as M, and the feature frame as N. For decoding without pruning, we compute the likelihood for every different frame-state pair.

1) For baseline diagonal model, the computation cost is on the order of $N*S*M*2D$ additions and multiplications. The $M*2D$ is the cost of calculating (1), which needs D multiplications and D additions for calculating the likelihood of each Gaussian component.
2) For STC and HLDA system, the cost is on the order of $N*(S*M*2D+L*2D^2)$ additions and multiplications, where L is number of transformation matrices. The front-end overload $L*2D^2$ is that we need to apply the linear transformations to feature vector before calculating the likelihood. In STC and HLDA system with global linear transformation, this overhead is marginal and can be ignored. With the increase of L, the overhead is continuously increased.
3) For MIC system, the cost is on the order of $N*(S*M*(D+K)+(1/2)KD^2)$ additions and multiplications, where K is number of global prototypes. The overhead is on computing the likelihood of the observation frame and the MIC prototypes.
4) For TMIC and TIOC system, the cost is on the order of $N*(S*M*(D+K_1)+(1/2)K_2D^2)$ additions and multiplications, where K_1 is number of parent nodes that Gaussian leaf nodes belong to and is equal to the layers of the tree, and K_2 is number of root node and all middle layer nodes. Because we use more prototypes, the computation cost is higher than MIC.
5) For TOC system, the cost is on the order of $N*(S*M*D*(1/2)D^2)$ additions and multiplications. This is equivalent to a full covariance system. Because the interpolation is made in covariance space instead of inverse covariance space, we cannot take advantage of computation saving in decoding.

From the above analysis, we can see that although the decoding performance of TOC is better than STC and MIC, the major drawback of the TOC is its computational complexity in decoding with the full covariance models. There exists one possible way of improving the decoding speed. In [30], full covariance decoding is based on the Cholesky decomposition of the inverse covariance matrix. This allows pruning the likelihood computation for a mixture component as soon as the partial sum across dimensions falls below a threshold. Second, A hierarchical Gaussian evaluation is described in [31]. By combining these two approaches, the decoding time for full covariance models has been sped up to is brought down to 3.3 times real-time without loss in accuracy [30]. This approach can be similarly applied on our TOC algorithm for faster decoding.

As far as the training time is concerned, our proposed TOC is faster than standard MIC because interpolation weights estimation can be cast into a decomposed rather than large joint optimization procedure.

In general, the TIOC scheme is recommended for building a fast system, and the TOC method is the choice to achieve the best recognition performance with sacrificing decoding efficiency.

VII. CONCLUSION

In this paper, we propose a tree-based covariance modeling (TCM) algorithm to estimate full covariance matrices of HMMs for speech recognition applications. In this TCM framework, we also derive four linear combination methods, namely, tree-based mixture of inverse covariance (TMIC), tree-based mixture of covariance (TMC), tree-based off-diagonal compensation (TOC), and tree-based inverse covariance off-diagonal compensation (TIOC) to interpolate the full covariance matrices. We evaluate our tree-based schemes on three different ASR tasks, and the results of our TOC compared to the diagonal covariance baseline is summarized in Table XI.

Based on experimental results, we have found the following.

1) For all linear combination algorithms tested, the tree-based covariance prototypes performs better than the nontree-based global prototypes in ASR.
2) The diagonal terms of covariance matrix which are the most reliably estimated parameters in a full covariance matrix should not be changed.
3) TOC yields best performance when compared with other full covariance modeling methods.

APPENDIX

In the numerical optimization method, we need to calculate gradient of the objective function $Q(\boldsymbol{\lambda}_m;\boldsymbol{\lambda}_m^{(n)})$ with respect to unknown interpolation weights $\boldsymbol{\lambda}_m$. In the following, we derive the formula to compute gradient for each of the above tree-based covariance modeling schemes.

1) TMIC: The gradient of the first part of (15) is

$$\frac{\partial \log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right|}{\partial \lambda_{m,k}} = \mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m \boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}\right\}$$

and the gradient of the second part of (15) is

$$\frac{\partial \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right)}{\partial \lambda_{m,k}} = \mathrm{Tr}\left(\frac{\partial \hat{\boldsymbol{\Sigma}}_m^{-1}}{\partial \lambda_{m,k}}\boldsymbol{\Sigma}_m\right) = \mathrm{Tr}\left(\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}\boldsymbol{\Sigma}_m\right).$$

Combine above two equations, the gradient of the objective function is

$$\frac{\partial Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m)}{\partial \lambda_{m,k}} = \mathrm{Tr}\left\{\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}(\hat{\boldsymbol{\Sigma}}_m - \boldsymbol{\Sigma}_m)\right\}.$$

2) TMC: The gradient of the first part of (15) is

$$\frac{\partial \log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right|}{\partial \lambda_{m,k}} = -\frac{\partial \log|\hat{\boldsymbol{\Sigma}}_m|}{\partial \lambda_{m,k}} = -\mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_{\mathrm{node},k}\right\}$$

and the gradient of the second part of (15) is

$$\begin{aligned}\frac{\partial \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right)}{\partial \lambda_{m,k}} &= \mathrm{Tr}\left(\frac{\partial \hat{\boldsymbol{\Sigma}}_m^{-1}}{\partial \lambda_{m,k}}\boldsymbol{\Sigma}_m\right)\\ &= -\mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_{\mathrm{node},k}\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right\}.\end{aligned}$$

Combine above two equations, the gradient of the objective function is

$$\frac{\partial Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m)}{\partial \lambda_m^{(i)}} = \mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_{\mathrm{node},k}\left[\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m - \boldsymbol{I}\right]\right\}.$$

3) TOC: The gradient of the first part of (15) is

$$\begin{aligned}\frac{\partial \log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right|}{\partial \lambda_{m,k}} &= -\frac{\partial \log|\hat{\boldsymbol{\Sigma}}_m|}{\partial \lambda_{m,k}}\\ &= -\mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m^{-1}\left[\boldsymbol{\Sigma}_{\mathrm{node},k} - \mathrm{diag}(\boldsymbol{\Sigma}_{\mathrm{node},k})\right]\right\}\end{aligned}$$

and the gradient of the second part of (15) is

$$\begin{aligned}&\frac{\partial \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right)}{\partial \lambda_{m,k}}\\ &= \mathrm{Tr}\left(\frac{\partial \hat{\boldsymbol{\Sigma}}_m^{-1}}{\partial \lambda_{m,k}}\boldsymbol{\Sigma}_m\right)\\ &= -\mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m^{-1}\left[\boldsymbol{\Sigma}_{\mathrm{node},k} - \mathrm{diag}(\boldsymbol{\Sigma}_{\mathrm{node},k})\right]\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right\}.\end{aligned}$$

Combine above two equations, the gradient of the objective function is

$$\begin{aligned}\frac{\partial Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m)}{\partial \lambda_{m,k}} = \mathrm{Tr}\Big\{&\hat{\boldsymbol{\Sigma}}_m^{-1}\left[\boldsymbol{\Sigma}_{\mathrm{node},k} - \mathrm{diag}(\boldsymbol{\Sigma}_{\mathrm{node},k})\right]\\ &\times\left[\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m - \boldsymbol{I}\right]\Big\}.\end{aligned}$$

4) TOIC: The gradient of the first part of (13) is

$$\frac{\partial \log\left|\hat{\boldsymbol{\Sigma}}_m^{-1}\right|}{\partial \lambda_{m,k}} = \mathrm{Tr}\left\{\hat{\boldsymbol{\Sigma}}_m\left[\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1} - \mathrm{diag}\left(\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}\right)\right]\right\}$$

and the gradient of the second part of (13) is

$$\frac{\partial \mathrm{Tr}\left(\hat{\boldsymbol{\Sigma}}_m^{-1}\boldsymbol{\Sigma}_m\right)}{\partial \lambda_{m,k}} = \mathrm{Tr}\left\{\left[\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1} - \mathrm{diag}\left(\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}\right)\right]\boldsymbol{\Sigma}_m\right\}.$$

Combine above two equations, the gradient of the objective function is

$$\frac{\partial Q(\boldsymbol{\lambda}_m;\hat{\boldsymbol{\lambda}}_m)}{\partial \lambda_{m,k}} = \mathrm{Tr}\left\{\left[\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1} - \mathrm{diag}\left(\boldsymbol{\Sigma}_{\mathrm{node},k}^{-1}\right)\right][\hat{\boldsymbol{\Sigma}}_m - \boldsymbol{\Sigma}_m]\right\}.$$

ACKNOWLEDGMENT

The authors would like to thank all the members in the Speech Group of Microsoft Research Asia, especially Dr. F Soong, for valuable suggestion and help.

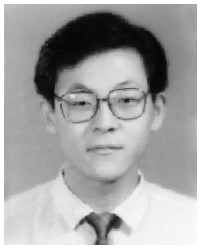
REFERENCES

[1] A. Ljolje, "The importance of cepstral parameter correlation in speech recognition," *Comput. Speech Lang.*, vol. 8, pp. 223–232, 1994.
[2] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-28, no. 4, p. 357, Aug. 1980.
[3] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1972.
[4] R. Haeb-Umbach and H. Ney, "Linear discriminant analysis for improved large vocabulary continuous speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1992, vol. 1, pp. 13–16.
[5] X. Aubert, R. Haeb-Umbach, and H. Ney, "Continuous mixture densities and linear discriminant analysis for improved context-dependent acoustic models," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1993, vol. 2, pp. 27–30.
[6] G. Saon, M. Padmanabhan, R. Gopinath, and S. Chen, "Maximum likelihood discriminant feature spaces," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2000, vol. 2, pp. 1129–1132.
[7] R. A. Gopinath, "Maximum likelihood modeling with Gaussian distributions for classification," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1998, vol. 2, pp. 661–664.
[8] N. Kumar, "Investigation of silicon-auditory models and generalization of linear discriminant analysis for improved speech recognition," Ph.D. dissertation, , Johns Hopkins Univ., Baltimore, MD, 1997.
[9] X. Liu, M. F. J. Gales, and P. C. Woodland, "Automatic complexity control for HLDA systems," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2003, vol. 1, pp. 132–135.
[10] B. Doherty, S. Vaseghi, and P. McCourt, "Full covariance modeling and adaptation in sub-bands," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2000, vol. 2, pp. 969–972.
[11] J. A. Bilmes, "Factored sparse inverse covariance matrices," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2000, vol. 2, pp. 1009–1012.
[12] 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.
[13] ——, "Maximum likelihood multiple subspace projections for hidden Markov models," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 2, pp. 37–47, Feb. 2002.
[14] V. Vanhoucke and A. Sankar, "Mixtures of inverse covariance," *IEEE Trans. Speech Audio Process.*, vol. 12, no. 3, pp. 250–264, May 2004.
[15] P. A. Olsen and R. A. Gopinath, "Modeling inverse covariance matrices by basis expansion," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 12, no. 1, pp. 37–46, Jan. 2004.
[16] S. Axelrod, R. Gopinath, and P. Olsen, "Modeling with a subspace constraint on inverse covariance matrices," in *Proc. Int. Conf. Spoken Language Processing*, 2001, vol. 9, pp. 2177–2180.
[17] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 7, no. 2, pp. 257–286, Feb. 1989.
[18] K. Shinoda and C. H. Lee, "A structural Bayes approach to speaker adaptation," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 3, pp. 276–287, Mar. 2001.
[19] M. Afify and O. Siohan, "Constrained maximum likelihood linear regression for speaker adaptation," in *Proc. Int. Conf. Spoken Language Processing*, 2000, pp. 861–864.

[20] C. J. Leggetter and P. C. Woodland, "Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models," *Comput. Speech Lang.*, vol. 9, pp. 171–185, 1995.
[21] T. Watanabe, K. Shinoda, K. Takagi, and E. Yamada, "Speech recognition using tree-structured probability density function," in *Proc. Int. Conf. Spoken Language Processing*, 1994, pp. 223–226.
[22] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, pp. 461–464, 1973.
[23] W. H. Press, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge University Press, 1986.
[24] P. Price, W. Fisher, J. Bernstein, and D. Pallett, "A database for continuous speech recognition in a 1000-word domain," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 1988, vol. 2, pp. 651–654.
[25] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book (for HTK Version 3.0)*. : , 2000 [Online]. Available: http://htk.eng.cam.ac.uk/
[26] E. Chang, Y. Shi, J. Zhou, and C. Huang, "Speech lab in a box: a Mandarin speech toolbox to jump start speech related research toolbox," in *Proc. Eur. Conf. Speech Communication and Technology*, 2001, pp. 2782–2799.
[27] E. Chang, J. Zhou, C. Huang, and K. F. Lee, "Large vocabulary Mandarin speech recognition with different approaches in modeling tones," in *Proc. Int. Conf. Spoken Language Processing*, 2000, pp. 983–986.
[28] R. Morton, D. Whitehouse, and D. Ollason, *The HAPI Book*. Cambridge, U.K.: Entropic Cambridge Research Laboratory.
[29] A.-V. I. Rosti and M. J. F. Gales, "Factor analyzed hidden Markov models for speech recognition," *Comput. Speech Lang.*, vol. 18, no. 2, pp. 181–200, 2003.
[30] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig, "The IBM 2004 conversational telephony system for rich transcription," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2005, vol. 1, pp. 205–208.
[31] G. Saon, G. Zweig, B. Kingsbury, L. Mangu, and U. Chaudhari, "An architecture for rapid decoding of large vocabulary conversational speech," in *Proc. Eur. Conf. Speech Communication and Technology*, 2003, pp. 1977–1980.

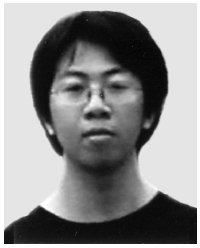
Ye Tian (M'03) received the B.Eng. degree from HuaZhong University of Science and Technology, Wuhan, China, in July 1998 and the Ph.D. degree from Tsinghua University, Beijing, China, in July 2003, both in electrical engineering.

From July 2003 to September 2005, he was with Microsoft Research Asia, Beijing, as a Postdoctoral Researcher. Since October 2005, he has been with Microsoft Speech and Natural Language, Redmond, WA, as a Software Design Engineer. His current research interests include all issues related to speech recognition and speech detection, especially robust voice activity detection, acoustic modeling, confidence measures, and speech recognition systems.



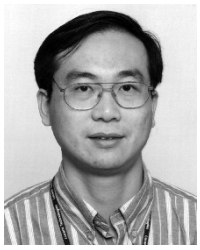
Jian-Lai Zhou received the B.S. and M.S. degrees from Xi'an University of Technology, Xi'an, China, in 1993 and 1996, respectively, and the Ph.D. degree from the Acoustic Institute, Chinese Academy of Science, Beijing, China, in 1999.

Since July 1999, he has been with the Speech Group, Microsoft Research Asia, Beijing, China, where he has worked on various aspects of both language-dependent and independent technologies for speech recognition. His current research interests include statistical learning, large-vocabulary continuous speech recognition, robust speech recognition in noisy environments, speech recognition on mobile devices, handwriting recognition, and multimodal input technology.



Hui Lin received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2003, where he is currently pursuing the M.S. degree in information and communication engineering.

From March 2004 to October 2004, he was a visiting student at Microsoft Research Asia, Beijing. His research interests include acoustic modeling for speech recognition, multimedia information retrieval, and graphic models for machine learning.



Hui Jiang (M'00) received the B.Eng. and M.Eng. degrees from the University of Science and Technology of China (USTC), Hefei, China, and the Ph.D. degree from the University of Tokyo, Tokyo, Japan, in September 1998, all in electrical engineering.

From October 1998 to April 1999, he was a Researcher in the University of Tokyo. From April 1999 to June 2000, he was with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, as a Postdoctoral Fellow. From 2000 to 2002, he was with Dialogue Systems Research, Multimedia Communication Research Laboratory, Bell Labs, Lucent Technologies, Inc., Murray Hill, NJ. Since fall 2002, he has been with the Department of Computer Science and Engineering, York University, Toronto, ON, as an Assistant Professor. His current research interests include all issues related to speech recognition and understanding, especially acoustic modeling, robust speech recognition, confidence measures, adaptive modeling of speech, spoken language systems, and speaker recognition/verification.