# 論文・著書情報

<table>
<thead>
<tr>
<th>項目</th>
<th>件名</th>
</tr>
</thead>
<tbody>
<tr>
<td>タイトル</td>
<td>A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing Systems</td>
</tr>
<tr>
<td>著者</td>
<td>Nakamasa Inoue, Koichi Shinoda</td>
</tr>
<tr>
<td>発行日</td>
<td>2011, 11</td>
</tr>
<tr>
<td>版权</td>
<td>Copyright (c) 2011 Association for Computing Machinery</td>
</tr>
<tr>
<td>検証</td>
<td>Copyright (C)2011 Association for Computing Machinery(ACM),. This is the author's version of the work. It is posted here by personal use. Not for redistribution. The definitive version of Record was published in Nakamasa Inoue, Koichi Shinoda, ACM Multimedia'11,2011,pp. 1357-1360.<a href="http://dx.doi.org/10.1145/2072298.2072014">http://dx.doi.org/10.1145/2072298.2072014</a></td>
</tr>
</tbody>
</table>

このファイルは著者（最終）版です。

 Powered by T2R2 (Tokyo Institute Research Repository)
A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing Systems

Nakamasa Inoue, Koichi Shinoda
Dept. of Computer Science,
Tokyo Institute of Technology, Japan
{inoue, shinoda}@ks.cs.titech.ac.jp

ABSTRACT
We propose a fast maximum a posteriori (MAP) adaptation technique for a GMM-supervectors-based video semantic indexing system. The use of GMM supervectors is one of the state-of-the-art methods in which MAP adaptation is needed for estimating the distribution of local features extracted from video data. The proposed method cuts the calculation time of the MAP adaptation step. With the proposed method, a tree-structured GMM is constructed to quickly calculate posterior probabilities for each mixture component of a GMM. The basic idea of the tree-structured GMM is to cluster Gaussian components and approximate them with a single Gaussian. Leaf nodes of the tree correspond to the mixture components, and each non-leaf node has a single Gaussian that approximates its descendant Gaussian distributions. Experimental evaluation on the TRECVID 2010 dataset demonstrates the effectiveness of the proposed method. The calculation time of the MAP adaptation step is reduced by 76.2% compared to that of a conventional method and resulting accuracy (in terms of Mean average precision) was 10.2%.

Categories and Subject Descriptors
I.4.9 [Computing Methodologies]: Image processing and computer vision—Applications

General Terms
Algorithm, Experimentations

Keywords
Video Semantic Indexing, GMM Supervectors, MAP Adaptation

1. INTRODUCTION
Video semantic indexing is one of the fundamental challenges in the field of computer vision. Its goal is to develop a generic method for automatically assigning semantic tags to video data. Detecting semantic concepts including objects, events and scenes is a challenging task due to within-class variations in shapes, colors, illuminations and backgrounds.

Many studies have started to focus on large-scale video recognition—a problem that requires a large amount of computational resources in recent years. For example, in the TREC Video Retrieval Evaluation (TRECVID) [1] workshop, which provides researchers with common tasks of video analysis, the size of target video resources has been doubling every year. While developing an accurate classification method is a top priority, a fast method for processing such a large amount of data is also needed. In response to these requirements, the motivation of the present study is to develop a fast and accurate semantic indexing system.

In recent researches, supervector coding of local features [2] has been proposed as an image-annotation method and applied to video recognition. For example, using a combination of Gaussian mixture model (GMM) supervectors and support vector machines (SVMs) has achieved better performance than that achieved by the bag-of-visual-words approach [3, 4]. With this method, a probability density function (pdf) of extracted local features is estimated by a GMM using maximum a posteriori (MAP) estimation. This MAP estimation step is called MAP adaptation. Aiming to reduce computational costs, most previous studies focused on reducing the cost of feature extraction (e.g., GPU implementations in [5]). The calculation time of the MAP adaptation step, however, has become the bottleneck of a system.

In the present study, we propose a fast maximum a posteriori (MAP) adaptation technique for a GMM-supervectors-based video semantic indexing system. We evaluated the technique by testing it on the TRECVID 2010 dataset.

2. PROPOSED METHOD
2.1 Gaussian Mixture Model
A probability distribution function (pdf) of local visual (or audio) features is estimated for each video shot. Here, Gaussian mixture models (GMMs), whose pdf is given by

$$p(x|\theta) = \sum_{k=1}^{K} w_k g_k(x), \quad g_k(x) = \mathcal{N}(x|\mu_k, \Sigma_k),$$

are employed where $x$ is a local feature, $\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is a set of parameters, $K$ is the number of mixture components, $w_k$ is a mixture coefficient, and $g_k(x)$ is a pdf with a mean vector $\mu_k$ and a covariance matrix $\Sigma_k$. 
The GMM parameters are estimated by using an expectation maximization (EM) algorithm with a maximum a posteriori (MAP) criterion. For MAP adaptation, a GMM for prior distribution, namely a universal background model (UBM), is first needed. The UBM presents how the features are distributed in the general case; therefore, the parameters used for the UBM, \( \theta^{(U)} \), is estimated by applying the EM algorithm to all features in training videos.

With the proposed method, only mean vectors are adapted for each shot. The MAP solution gives the following equations:

\[
\hat{\mu}_k^{(v)} = \frac{\tau \hat{\mu}_k^{(v)}}{\tau + \sum_{i=1}^{\ell} c_{ik} x_i}, \quad c_{ik} = \frac{w_k g_k(x_i)}{\sum_{k=1}^{N} w_k g_k(x_i)} \tag{2}
\]

where \( X_v = \{ x_i \}_{i=1}^{\ell} \) is a set of (one type of) feature vectors extracted from the \( s \)-th shot, \( \hat{\theta}^{(v)} \) is the UBM parameter, and \( \tau \) is a predefined hyper-parameter.

### 2.2 Tree-structured GMMs

Given the UBM, a tree structure of Gaussian components that makes calculation of Eq. (2) efficient is constructed. The basic idea is to cluster Gaussian components by a single Gaussian. For given Gaussian components \( g_m = N(\mu_m, \Sigma_m) \) and combination coefficients \( \alpha_m \) (\( \alpha_m \geq 0, \sum \alpha_m = 1 \) \( m = 1, 2, \cdots, M \)), we define a combined single Gaussian by

\[
\bar{\mu} = \sum_{m=1}^{M} \alpha_m \mu_m, \quad \bar{\Sigma} = \sum_{m=1}^{M} \alpha_m (\Sigma_m + \mu_m \mu_m^T) - \bar{\mu} \bar{\mu}^T \tag{3}
\]

If only a set of Gaussians components \( G = \{ g_1, g_2, \cdots, g_M \} \) is given,

\[
g(G) = \prod_{m=1}^{M} \frac{1}{M} g_1 + \frac{1}{M} g_2 + \cdots + \frac{1}{M} g_M \tag{5}
\]

is simply used. Figure 1 shows an example of a tree-structured GMM. Each leaf node corresponds to a Gaussian component of the GMM, and each other node has a single Gaussian obtained by combining corresponding Gaussian pdfs of descendant nodes. As for the following tree-construction algorithm, it is assumed that the maximum number of child nodes for each layer (with the exception of the leaf layer) is given. For example, if the maximum number of child nodes for the first layer is two and that for the second layer is three, the resulting tree will be designed as shown in Figure 1. In this case, a tree with a depth of three (including the leaf node) is obtained. This tree-structured GMM is denoted as \( T_{(2,3)} = (V, E, G_{\text{tree}}) \) where \( V \) is a set of nodes, \( E \) is a set of edges, and \( G_{\text{tree}} = \{ g^{(v)} | v \in V \} \) is a set of Gaussian pdfs. In general, an node at the \( t \)-th layer of a tree \( T_{(P_1, P_2, \cdots, P_r)} \) has, at most, \( P \) child nodes.

The node pdfs \( g^{(v)} \) of a tree \( T_{(P_1, P_2, \cdots, P_r)} \) are created by the following algorithm. Note that \( G_{\text{GMM}} = \{ g_1, g_2, \cdots, g_M \} \) is a set of mixture components of the UBM, \( G^{(v)} \) is a subset of \( G_{\text{GMM}} \) corresponding to node \( v, g^{(v)} \) is a Gaussian pdf for node \( v \), and \( d(g_m, g_n) \) is the sum of Kullback-Leibler divergence from \( g_m \) to \( g_n \) and that of from \( g_n \) to \( g_m \).

#### 1. Prepare a queue and enqueue \((r, G^{(r)})\), where \( r \) is the root node, and \( g^{(r)} = g(G^{(r)}) \) (\( G^{(r)} = G_{\text{GMM}} \)).

#### 2. (Min-max Initialization) Dequeue \((v, G^{(v)})\). Let \( \{ c_p \}_{p=1}^{P} \) be the child nodes of node \( v \). For \( p = 1, 2, \cdots, P \), initialize child Gaussian pdfs as follows:

\[
g^{(c_p)} = \alpha g^{(c_p)} + (1 - \alpha) g^{(v)} \tag{6}
\]

where \( 0 \leq \alpha \leq 1 \), and \( g^{(c_p)} \) is chosen from \( G^{(v)} \) by

\[
g^{(c_p)} = \begin{cases} 
\alpha g^{(v)} & (p = 1) \\
\arg \max_{g \in G^{(v)}} \min_{g \in G^{(c_p)}} d(g, g^{(c)}) & (\text{otherwise})
\end{cases}
\]

3. (Clustering by \( k \)-means) Assign each Gaussian component \( G^{(c_p)} \) to the nearest child node, i.e.,

\[
G^{(c_p)} = \arg \min_{g \in G^{(c_p)}} d(g, g^{(c_p)}). 
\]

4. For \( p = 1, 2, \cdots, P \), enqueue \((c_p, G^{(c_p)})\) if \( c_p \) is not in the \((T + 1)\)-th layer and \( |G^{(c_p)}| > 1 \). Go to step 5 if the queue is empty; otherwise, return to step 2.

5. For each node \( v \) in the \((T + 1)\)-th layer, create leaf nodes \( f \) for each \( g \in G^{(c)} \subseteq G_{\text{GMM}} \) and set \( g^{(f)} = g \).

#### 2.3 Fast MAP Adaptation

A fast MAP adaptation technique which estimates \( c_{ik} \) in Eq. (2) efficiently by using a tree-structured GMM is explained in the following. For a constructed tree-structured GMM \( T_{(P_1, P_2, \cdots, P_r)} \), node weights are first defined as follows.

1. For each leaf node, set \( w^{(f)} = w_k \) if \( g^{(f)} = g_k \in G_{\text{GMM}} \).

2. Calculate weights for \( t = T + 1, T, \cdots, 1 \) as follows. For each node \( v \) in the \( t \)-th layer,

\[
w^{(v)} = \sum_{c \in G^{(v)}} w^{(c)}, \tag{7}
\]
where \( C(v) \) is a set of child nodes of the node \( v \).

The algorithm for estimating \( c_{ik} \) for feature vector \( x_i \) is described as follows. The key idea is to construct a GMM of active nodes \( V_A \), which means vector \( x_i \) will be assigned to descendants of nodes in \( V_A \). \(|V_A|\) is kept small by obtaining active nodes from the root node.

1. Set \( V_A \leftarrow \{r\} \), where \( r \) is the root node.
2. Expand active nodes by making child nodes of the active nodes active:
\[
V_A \leftarrow \bigcup_{v \in V_A} C(v),
\]
where \( C(v) \) is a set of child nodes of the node \( v \). Here, \( C(\ell) = \{\ell\} \) is used for leaf nodes \( \ell \) to keep the leaf nodes active.
3. Consider an active GMM given by
\[
p(x|V_A) = \sum_{v \in V_A} \hat{w}^{(v)}(x) \cdot \mu^{(v)} = \sum_{v \in V_A} w^{(v)}(x).
\]
Calculate\[
\hat{c}_{i}^{(v)} = \frac{w^{(v)}(x_i)}{\sum_{v \in V_A} w^{(v)}(x_i)} = \frac{w^{(v)}(x_i)}{N} \]
4. Keep a node \( v \) active if \( \hat{c}_{i}^{(v)} \) is larger than the pre-determined threshold \( c_{TH} \), i.e.
\[
V_A \leftarrow \{v \in V_A \mid \hat{c}_{i}^{(v)} > c_{TH}\}
\]
5. If all nodes in \( V_A \) are leaf nodes, output
\[
\hat{c}_{ik} = \begin{cases} 
\hat{c}_{i}^{(\ell)} & (\ell \in V_A, g^{(\ell)} = g_k) \\
0 & \text{(otherwise)}
\end{cases}
\]
Otherwise, return to step 2.

Since the observed \( \hat{c}_{ik} \) for non-active nodes is 0, the sum in Eq. (2) can be calculated for non-zero \( \hat{c}_{ik} \) only. Moreover, calculation speed and limits of approximation can be controlled by selecting the threshold in \( 0 < c_{TH} \leq 1 \). Note that the same \( c_{ik} \) as given by Eq. (2) is obtained if \( c_{TH} \) is set to 0 (because all leaf nodes will be active at the final step).

### 2.4 GMM Supervisor

The combination of GMM supervisors and support vector machines (SVMs) was first proposed as a speaker recognition method [6] and has been applied to image and video recognition [3, 4]. GMM supervisors are created for each shot given by
\[
\phi(X_s) = \begin{pmatrix} 
\hat{\mu}_1^{(s)} \\
\vdots \\
\hat{\mu}_K^{(s)} 
\end{pmatrix}, \quad \hat{\mu}_k^{(s)} = \sqrt{w_k^{(s)}(\Sigma_k^{(s)})^{-\frac{1}{2}}} \tilde{\mu}_k^{(s)},
\]
where \( \tilde{\mu}_k^{(s)} \) is an adapted mean vector obtained from Eq. (2), and \( \tilde{\mu}_k^{(s)} \) is the GMM parameter for the UBM. Finally, SVMs are trained for each semantic concept and for each type of features by using RBK kernels.

Note that while the scope of this paper is fast creation of the GMM supervectors, the proposed technique can be used for creating the Fisher vectors [7].

### 3. EXPERIMENTS

#### 3.1 Database and Task

Our experiments were conducted on the TRECVID 2010 dataset [1]. The dataset consists of 400 hours of Internet archive videos with creative commons licenses. The shot boundaries are automatically detected and provided with the video data. Half of the videos were used for training, and the others were used for testing. The number of shots was 119,685 for training and 146,788 for testing. The task of semantic indexing in the experiment is to detect the 30 semantic concepts listed in Table 1. The target 30 concepts (including objects, events and scenes) were selected at the TRECVID 2010 workshop. They are considered to be useful for video searching.

The evaluation measures are Mean average precision (Mean AP) and calculation time (CT) of the testing phase. Mean AP is defined as the mean of APs over all 30 target concepts, and APs are estimated by using a method called inferred average precision (Inf AP), proposed in [8].

#### 3.2 Experimental Conditions

The following four types of visual and audio features are extracted from video data: 1) SIFT features with Harris-Affine detector (SIFT-Har), 2) SIFT features with Hessian-Affine detector (SIFT-Hes), 3) SIFT and hue histogram with dense sampling (SIFT-Dense), and 4) MFCC audio features (MFCC).

The number of mixtures (vocabulary size) \( K \) for GMMs was 512 for visual features and 256 for audio features. For computational efficiency, it was assumed that covariance matrices are diagonal. Hyper parameter \( \tau \) in the MAP adaptation was set to 20.0. SVMs were trained for each semantic concepts by using the libSVM implementation [9]. For tree-structured GMMs, the optimal tree structure \( T_{opt} \) was selected as
\[
T_{opt} = \arg\max_T CT(T), \quad \tau \in S
\]
where \( CT(T) \) is calculation time when the tree \( T \) is used. The trees \( T_{4(4,5)} \), \( T_{5(5,5)} \), \( T_{4(4,4,5)} \) and \( T_{3(3)} \) were selected for SIFT-Har, SIFT-Hes, SIFT-Dense and MFCC, respectively. Parameter \( \alpha \) in Eq. (6) was fixed to 0.1. Threshold \( c_{TH} \) for the fast MAP adaptation was set to 0.001. Here, a low threshold was set so as to keep detection performance
Table 2: Comparison of Mean Inf AP (%) in terms of different features.

<table>
<thead>
<tr>
<th>Feature</th>
<th>No tree</th>
<th>$T_{opt}$</th>
<th>$T_{binary}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT-Har</td>
<td>6.30</td>
<td>6.32</td>
<td></td>
</tr>
<tr>
<td>SIFT-Hes</td>
<td>5.96</td>
<td>6.08</td>
<td></td>
</tr>
<tr>
<td>SIFTH-Dense</td>
<td>7.10</td>
<td>6.95</td>
<td></td>
</tr>
<tr>
<td>MFCC</td>
<td>1.99</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>Fusion</td>
<td>10.15</td>
<td>10.16</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Calculation time (sec) for MAP adaptation.

<table>
<thead>
<tr>
<th>Feature</th>
<th>No tree</th>
<th>$T_{opt}$</th>
<th>$T_{binary}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT-Har</td>
<td>1.62</td>
<td>0.49</td>
<td>0.98</td>
</tr>
<tr>
<td>SIFT-Hes</td>
<td>1.67</td>
<td>0.48</td>
<td>1.00</td>
</tr>
<tr>
<td>SIFTH-Dense</td>
<td>2.89</td>
<td>0.81</td>
<td>1.89</td>
</tr>
<tr>
<td>MFCC</td>
<td>0.22</td>
<td>0.03</td>
<td>0.08</td>
</tr>
</tbody>
</table>

In the experiments, calculation time was measured by using one Intel Xeon 2.93 GHz CPU.

3.3 Results

3.3.1 Mean Inf APs

Table 2 summarizes obtained Mean Inf APs for each feature and a fusion method. The Fusion combines four features by calculating the weighted sum of detection scores (fusion weights were optimized by using two-fold cross validation on training data). As a result, we can see that the Mean Inf APs using tree-structured GMMs are comparable to those using no trees.

Table 4: Comparison of Mean Inf AP (%), calculation time (sec) for MAP adaptation (CT), number of leaf nodes $|V_A|$ and MAE by using different thresholds $c_{TH}$ (for SIFTH-Dense).

| $c_{TH}$ | Mean Inf AP | CT | $|V_A|$ | MAE  |
|----------|-------------|----|--------|------|
| 0.001    | 6.95        | 0.81| 17.0   | 0.32 |
| 0.01     | 6.99        | 0.68| 11.2   | 0.53 |
| 0.1      | 6.60        | 0.59| 7.3    | 0.80 |
| 0.5      | 6.41        | 0.53| 5.4    | 0.98 |

As $c_{TH}$ gets higher, the calculation time shortens, but Mean Inf AP was decreased when $c_{TH} = 0.1$ and 0.5. Moreover, the number of active leaf nodes decreases, and MAE increases. It can thus be concluded that calculation time should be reduced not by setting a high threshold $c_{TH}$ but by selecting a better-structured tree to keep detection performance high. In particular, $c_{TH}$ should be equal to or smaller than 0.01 in this case.

4. CONCLUSION

A fast MAP adaptation technique using tree-structured GMMs for a video semantic indexing system was proposed. The detection time was reduced by 56.6%, compared to that of conventional method, while high detection performance was maintained. The best result, in terms of Mean Inf AP, attained by our fusion method tested on the TRECVID 2010 dataset was 10.16% when tree-structured GMMs were used and 10.15% when no trees were used. Our future work will focus on a GPU implementation of the fast MAP adaptation. As a result, feature extraction and MAP adaptation will be independently conducted for each local feature by GPUs.

5. REFERENCES