論文 / 著書情報
Article / Book Information

題目(和文)	大規模映像資源のための高速・高性能なセマンティックインデクシング
Title(English)	Efficient and Effective Semantic Indexing for Large-Scale Video Resources
著者(和文)	井上中順
Author(English)	Nakamasa Inoue
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第9552号, 授与年月日:2014年3月26日, 学位の種別:課程博士, 審査員:篠田　浩一,佐藤　泰介,徳永　健伸,村田　剛志,杉山　将
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第9552号, Conferred date:2014/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Efficient and Effective Semantic Indexing for Large-Scale Video Resources

Nakamasa Inoue

Supervised by Professor Koichi Shinoda

Department of Computer Science
Graduate School of Information Science and Engineering
Tokyo Institute of Technology

Dissertation submitted to the Tokyo Institute of Technology for the degree of Doctor of Engineering

Feb. 2014

Abstract

Video semantic indexing aims to assign semantic concepts to a video segment and is one of the fundamental and important problems in computer vision. In this study, we propose an efficient and effective semantic indexing system, which extends the bag-of-visual-words system to a probabilistic framework using a Gaussian mixture model (GMM). To improve the modeling accuracy, we introduce a q-Gaussian mixture model, which controls the tail-heaviness of the GMM. To improve the speed of the system, we propose two complementary techniques: fast parameter estimation using a tree-structured GMM and Neighbor-to-Neighbor (NTN) search.

We evaluated our system on the TRECVID video benchmark. We achieved Mean Average Precision of 0.321, while the computational cost of parameter estimation is reduced by 76.2%.

Acknowledgments

I would like to express my sincerest thanks and gratitude to my advisor Professor Koichi Shinoda for his support during my study at Tokyo Institute of Technology.

I would like to express my gratitude to Professor Sadaoki Furui and Associate Professor Takahiro Shinozaki for their support and kind help in my research. I would also like to thank every member of the Shinoda Laboratory for their help and discussion.

I would like to extend my appreciation to my thesis committee for their advices and comments: Professor Taisuke Sato, Professor Takenobu Tokunaga, Professor Tsuyoshi Murata, and Professor Masashi Sugiyama.

Last but not least, I would like to thank my family for all their support through this study at Tokyo Institute of Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recently, a large amount of video data has been made available through online archives. For example, over 6 billion hours of video are viewed each month on YouTube. Since it is often difficult to find relevant video manually from such archives, an effective video-search system is in demand. Current search engines require users to develop metadata such as key-words and a short summarization in text to efficiently locate video in their archives. However, developing metadata is costly for users so attached metadata is often too simple and not enough for video-search purposes. Furthermore, detailed metadata is required for video-segment retrieval aiming at finding a specific scene or a specific object in video. Video-segment retrieval is still known to be in a nascent stage [1] and thus is hardly served by present-day search engines.

In this study, we focus on semantic indexing which is necessary as a basis for automatically generating metadata. Semantic indexing aims to assign semantic concepts, which include objects, actions, and scenes such as *airplane*, *bus*, *singing*, *dancing*, *cityscape*, and *nighttime*, to video segments. This has been a challenging task due to the semantic gap [2]: "the lack of correspondence between the low-level features and the high-level semantic concepts", where low-level features present color, texture, shape, and spatiotemporal features extracted from video.

Most previous studies used a statistical model to construct the relationship between low-level features and semantic concepts. In particular, the bag-of-visual-words (BoW) [3, 4], a statistical method for image recognition, has been proven to be effective for video semantic indexing. It repre-

sents an image by a histogram of low-level image descriptors such as scale-invariant feature transform (SIFT) [5]. In BoW, vector quantization (VQ) is needed to extract a histogram. However, quantization errors often degrade the indexing performance. To solve this problem, several soft clustering methods [6, 7, 8] are introduced. In particular, Gaussian mixture models (GMMs) [8] are often preferred since they are a straightforward extension of VQ to a probabilistic framework.

In video search, we should consider not only image information but also audio information to find video that the user wishes to view [9, 10, 11]. For example, a combination of a music and an action sometimes has a specific meaning in a dance scene. Some previous studies have analyzed speech in news video [12, 13] by using speech recognition methods. However, they are not very effective for video other than news such as consumer generated video in which speech is often spontaneous with noise or a background music. Furthermore, acoustic information other than speech, for example sounds from a musical instrument, can not be captured by the speech recognition methods. To capture visual and audio information effectively, a multi-modal method is necessary for semantic indexing.

On the other hand, since video has more information than an image, computational cost becomes a problem in real applications. Since vector quantization is the most computationally expensive part in the BoW algorithm [3], many studies have been done to develop fast VQ algorithms. For example, approximate nearest neighbor (ANN) algorithms such as randomized kd-trees [14] and hierarchical k-means tree [15] are known to provide speed-ups VQ with only minor loss in accuracy. To improve both speed and accuracy, we need to extend them to a probabilistic framework, for example to GMMs.

In this study, we propose a fast and accurate semantic indexing system. We have four main contributions as follows. 1) A multi-modal semantic indexing system which uses GMMs : we extend the BoW to a probabilistic framework where a video shot is represented by a GMM. Audio and visual features are extracted from video. 2) A fast parameter estimation technique using a tree-structured GMM : we propose fast maximum a posteriori (MAP) adaptation for estimating GMM parameters. 3) An improved modeling by a q-GMM : we improve the accuracy of modeling by introducing a parameter q to control the tail-heaviness of a GMM. 4) A fast search technique namely

Neighbor-to-Neighbor (NTN) search : we further improve the speed of the system especially when densely-sampled image descriptors are used.

The remainder of the dissertation is organized as follows. Chapter 2 gives an over view of semantic indexing. Chapter 3 describes the multimodal semantic indexing system. Chapter 4 describes the fast MAP estimation technique using a tree-structured GMM . Chapter 5 describes the extension of the system to q-Gaussian mixture models. Chapter 6 describes the NTN search. Finally, conclusions and future work are described in Chapter 7.

Chapter 2

Semantic Indexing

In this chapter, the fundamental approaches to semantic indexing will be discussed. The most common framework for semantic indexing consists of the three steps in Figure 2.1. In the first step, low-level features are extracted. Here, video is represented by a set of low-level features $X = \{x_i\}_{i=1}^{N}$ where $x_i \in \mathbb{R}^d$ is a vector describing local information such as RGB-color and texture, and N is the number of low-level features. In the second step, video is represented by a vector $\phi(X)$. This vector is so-called a video representation, for example a histogram representation in the bag-of-visual-word approach [3, 4]. In the third step, a detection score $f(\phi(X))$ is calculated where f is a discriminative function often trained based on supervised learning methods. Relation between semantic concepts sometimes considered in this step. The following sections discuss previous approaches for each step.

2.1 Low-Level Feature Extraction

Low-level feature extraction typically consists of two phases: key-point detection and feature description.

Key-Point Detection

In the first phase, key-point detectors are applied to find interest regions in video or in an image. Many detectors are proposed not for semantic indexing but for other computer-vision applications such as image matching [16, 17, 18], texture recognition [19], and robot servoing [20, 21]. However, they are also often effective for semantic indexing.

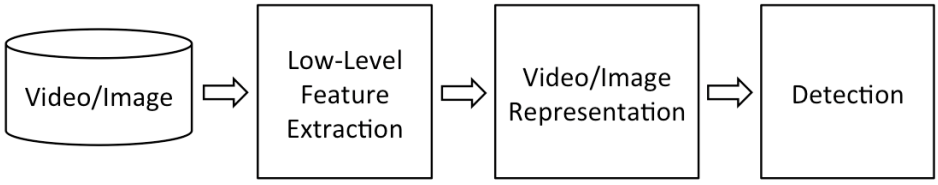

Figure 2.1: The most common framework for semantic indexing: 1) Low-level features are extracted from video, 2) A Video/Image representation is extracted, 3) A detection score is calculated.

For image matching, Lowe [5] proposed Scale Invariant Feature Transform (SIFT), which is the most widely used algorithm in many applications. In SIFT, key-points are detected based on the Difference-of-Gaussian (DoG) detector which is a simplification of the Laplacian-of-Gaussian (LoG) detector [5] to detect edges in a scale invariant way. Mikolajczyk et al. [22] proposed the Harris-affine detector which is robust against affine transforms. They also proposed the Hessian-affine detector to detect blob structure, which is complementary to the Harris-affine detector. Maximally stable external region (MSER) [17] focused on segmentation in an image and is a precise blob detector while its computational cost is higher than the Hessian-affine detector. For image classification, dense sampling which uses grid points have shown to be effective [23, 37]. The combination of dense sampling with above key-point detectors often improve the performance of image classification. For action recognition in video, some works focus on temporal information. Space-Time Interest Points (STIP) [24] extends the Harris-corner detector to video. Dense Trajectories [25] tracks objects in video at grid points.

Feature Description

In the second phase, a feature vector is extracted from each local region around a key-point. SIFT [5] extracts local gradient vectors which are robust against changes in scale, rotation, and illumination. A SIFT vector is 128 dimensional which consists of 8-dimensional histograms for 4x4 blocks. Histogram of oriented gradients (HOG) [26] proposed for human detection also extracts local gradient vectors as SIFT, but its dimension is often larger than SIFT. A typical HOG vector concatenates 9-dimensional histograms for 7x15 blocks. Local binary pattern (LBP) is another feature robust against

changes in illumination. The combination of HOG and LBP is proposed in [27] for human detection. Note that these features are extracted from a gray-scale image. To capture color information, J. Weijer proposed hue histogram [28] and its combination with SIFT. Extensions of SIFT to color spaces such as HSV space and Opponent SIFT are proposed in [29].

Some works focused on fast algorithms and implementations of SIFT. Speeded-Up Robust Features (SURF) improved the speed of Hessian detector and used it instead of the DoG detector in SIFT. BRIEF [30] and ORB [31], which are specialized for image matching, further improved the speed by introducing binary descriptors. GPU implementations [32, 67, 68] are known to be effective for large-scale image processing.

2.2 Modeling

In the bag-of-visual-words (BoW) approach [3], an image is represented as a histogram of visual words obtained by applying VQ to each low-level descriptors. k-means clustering is often used for training a visual codebook that consists of several thousands of visual words.

Soft-assignment approaches are effective for reducing quantization errors in VQ and thus they outperform BoW. Gemert *et al.* [6] proposed a kernel codebook in which each low-level descriptor is assigned to all visual words in a soft manner with weighting. Hang *et al.* [33, 7] used sparse coding which assigns a low-level descriptor to several tens of visual words by solving a constrained least square fitting problem. Perronnin *et al.* [8] used a Gaussian mixture model (GMM) for a codebook which holds mean and variance information for each visual word. The GMM is a straight-forward extension of BoW to a probabilistic framework.

Recently, high-dimensional image representations have been proven to be effective in image classification. Vector of locally aggregated descriptor (VLAD) [34] and super-vector coding [23] use the first order differences between low-level descriptors and visual words in addition to the BoW histogram. Fisher vector [35] represents an image as a concatenation of parameters of a parametric probability model. Perronnin *et al.* [36, 37] achieved the best performance in the PASCAL VOC image classification challenge by using a GMM as the parametric probability model for the Fisher vector. They reported that an normalization technique [37] is needed since the Fisher

vectors become sparser as the number of Gaussians increases. They proposed the L2+power normalization to make the Fisher vectors dense. The normalized fisher vector is called "improved Fisher kernel (IFK)". Chatfield *et al.* [38] reported that IFK is the best of these recent image representations.

On the other hand, several previous works focused on applying Tsallis statistics [39, 40, 41] to image processing. Tsallis q-entropy, which is a generalization of the Boltzmann-Gibbs (BG) entropy, is used for image thresholding for foreground extraction in [42, 43, 44, 45]. These works show that long-range correlation between foreground pixels can be modeled by using the Tsallis q-entropy. Fabbrib *et al.* [46] applied Tsallis q-entropy to image texture classification. They reported that texture classification accuracy is improved by using Tsallis q-entropies for multiple q-values as a feature vector. A q-Gaussian distribution, which is a generalization of Gaussian distribution and Student's t-distribution, is expected to effectively represent a distribution of low-level features in the bag-of-visual-words framework. Since a mixture of long-tailed distirbutions such as a t-mixture model [49] improves the performance of image registration [47] and image segmentation [48], we expect it will improve the performance of semantic indexing. To the best of our knowledge, we are the first to apply Tsallis statistics to the bag-of-visual-words framework.

To reduce the cost of this step, many previous studies have shown that a tree-sturucture is effective. Nistér *et al.* [50] propose a "vocabulary tree" which uses a hierarchical k-means tree. Lowe [5] uses a kd-tree for nearest neighbor search of SIFT descriptors. Muja and Lowe [15] have proposed an automatic selection method from the recent two approximate nearest neighbor (ANN) algorithms: randomized kd-trees [14], and hierarchical k-means tree [15]. They have provided it as a fast software library for approximate nearest neighbors (FLANN). Sparse coding [33] assigns several tens of codewords to an input vector by solving a constrained least square fitting problem. J. Wang el al. [7] introduced k-nearest neighbor (k-NN) search as the preprocessing to the sparse coding. The tree-structured GMM in this study extends the hierarchical k-means to a GMM framework.

2.3 Detection

In the detection step, a discriminative function to compute confidense scores is trained on labeled data. Here, we focus on supervised learning methods.

In semantic indexing, a learning method must handle imbalance in the number of positives and negatives since the number of positive samples are often limited. Moreover, it also must handle the problem of the curse of dimensionality since the dimension of a video/image representation is often very high. In this demands, the most well-known and effective method is the support vector machine (SVM), which separates a feature space into two different classes, positive and negative. To estimate a posterior probability, a probability for a particular semantic concept which is not given by an SVM, Platt [51] and Lin et al. [52] suggest a sigmoid fitting of SVM scores where parameters in the sigmoid function are estimated based on maximum likelihood criteria. The logistic regression (LR) is also known to estimates the posterior probability. While the LR sometimes performs better than the SVM, its computational cost is also larger than the SVM. Hence, the SVM has become the default choice in most semantic indexing schemes.

In many supervised learning methods including the SVM and the LR, a kernel trick which enables non-linear classification often improves the classification performance significantly. In general, the (Gaussian) radial basis function (RBF) kernel is almost the best choice. However, Zhang [53] showed that earth-movers-distance kernel [54] and χ^2 kernel are more optimal than it in the bag-of-visual-words framework. Since the BoW represents an image by a histogram of visual words, which sometimes have frequent but meaningless words, these kernels consider a normalization of the histogram to obtain better performance than the RBF. Some recent works are focusing on kernel learning. For example, multiple kernel learning (MKL) [55] enables to learn a linear combination of pre-dfined kernels, e.g. kernels for different types of features. Multiple kernel Fisher discriminant analysis (MK-FDA) [56] is an extension of the FDA to multiple kernels. However, these methods are computationally more expensive than the SVM. The linear combination of SVM scores is known to be a reasonable alternative for them.

Chapter 3

Multi-Modal Semantic Indexing

3.1 Overview

In this chapter, we describe our multi-modal semantic indexing system, which uses Gaussian-mixture-model (GMM) supervectors. The overview of our semantic indexing system is shown in Fig. 3.1. We assume videos are automatically segmented into shots in preprocessing. A shot consists of continuous frames without switching between cameras.

Our system consists of three parts. First, visual and audio low-level features are extracted from a video shot. The SIFT features are extracted by using three different interest point detectors: Harris-Affine [22], Hessian-Affine [22], and Dense [57]. The MFCCs, which describe the short-time spectral shape of audio frames, are extracted to capture audio information. The details of low-level feature extraction are described in Sec 3.2. Second, a GMM supervector is created for each type of low-level features. A GMM models the distribution of low-level features extracted from a video shot. Its parameter is estimated by using MAP adaptation. Third, SVM scores for each type of low-level features are fused to compute a final score.

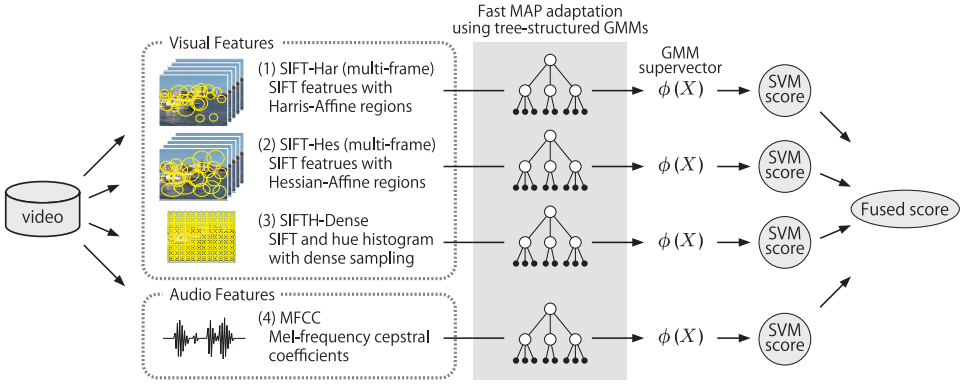

Figure 3.1: Overview of our semantic indexing system. Our system consists of three parts: 1) low-level feature extraction, 2) GMM supervector extraction by using fast MAP adaptation, and 3) SVM classification. First, visual and audio features are extracted. Second, GMM parameters are estimated by using MAP adaptation. Tree-structured GMMs are used to improve the speed of MAP adaptation. Third, the outputs of SVMs for the four feature types are fused to compute a final score.

3.2 Feature Extraction

3.2.1 Visual Features

SIFT features with Harris-Affine detector (Har-SIFT)

Scale-invariant feature transform (SIFT) [5] is a low-level feature extraction method that is widely used for object categorization. The extracted features are invariant to image scaling and changing illumination. The Harris-Affine local region detector [22], which is an extension of the Harris corner detector, provides affine-invariant local regions.

Let $I(x, y)$ be the luminosity value at a point (x, y). A point (x, y) is a Harris corner if the following value R is larger than a predetermined threshold (Figure 3.2),

$$R = \det M - k\,(\mathrm{trace}M)^2 = \lambda_1\lambda_2 - k\,(\lambda_1 + \lambda_2)^2 \tag{3.1}$$

where k is a parameter, λ_1 and λ_2 are eigenvalues of the following matrix M:

$$M = \sigma_D^2 g(\sigma_I) * \begin{bmatrix} I_x^2(x, y, \sigma_D) & I_x I_y(x, y, \sigma_D) \\ I_x I_y(x, y, \sigma_D) & I_y^2(x, y, \sigma_D) \end{bmatrix}. \tag{3.2}$$

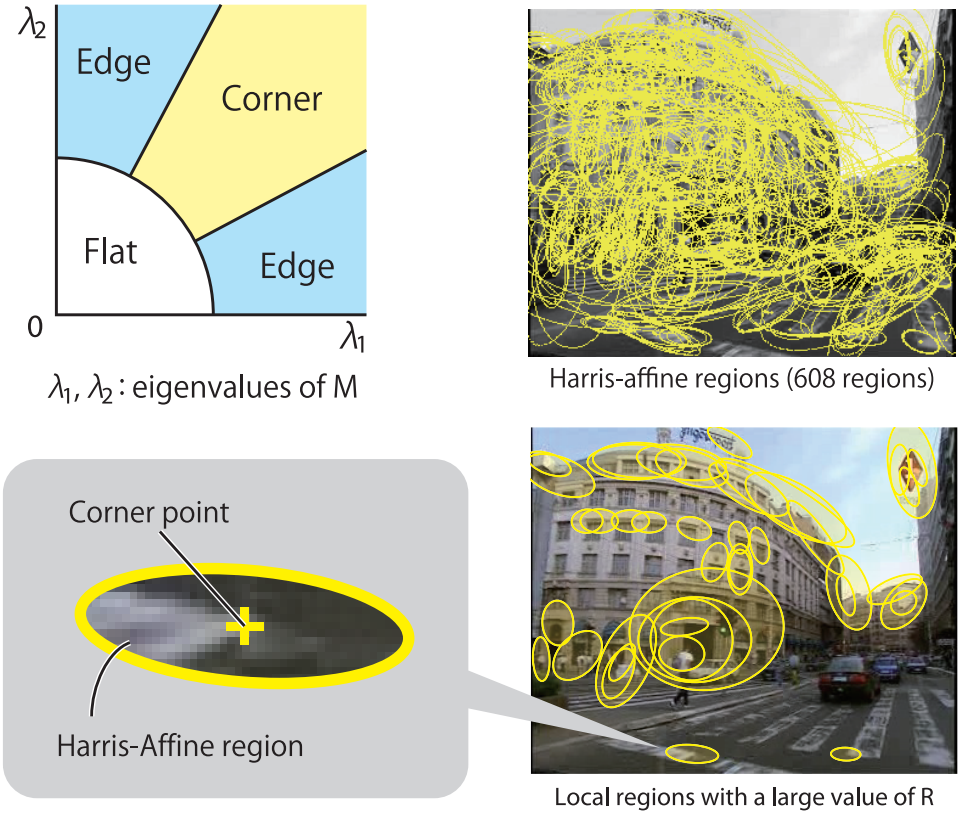

Figure 3.2: Harris corner detector

Here, $g(\sigma_I)$ is a Gaussian fuction with a window size of σ_I and I_x, I_y are the partial derivative of $I(x, y)$ given by

$$I_x(x, y, \sigma_D) = I(x + 1, y, \sigma_D) - I(x - 1, y, \sigma_D) \tag{3.3}$$

$$I_y(x, y, \sigma_D) = I(x, y + 1, \sigma_D) - I(x, y - 1, \sigma_D). \tag{3.4}$$

For each corner point, a SIFT descriptor is extracted as follows (Figure 3.3).

1. Compute a gradient $m(x, y)$ and an orientation $\theta(x, y)$ by

$$m(x, y) = \sqrt{(I_x(x, y, \sigma_D))^2 + (I_y(x, y, \sigma_D))^2}, \tag{3.5}$$

$$\theta(x, y) = \tan^{-1} \frac{I_y(x, y, \sigma_D)}{I_x(x, y, \sigma_D)}. \tag{3.6}$$

2. Split a local region into 4x4 blocks to compute a histogram of oriented gradients of 8 directions $\{\theta_i = \frac{\pi}{4} i\}_{i=0}^{7}$ for each block.

$$h_i = \sum_{x,y} w(x, y)\delta(\theta_i, \theta(x, y)), \tag{3.7}$$

$$w(x, y) = G(x, y, \sigma)m(x, y). \tag{3.8}$$

3. Concatenate 16 histograms to obtain 128-dimensional vector.

The proposed method uses 32-dimension SIFT features, whose dimensions are reduced from 128 to 32 by applying principal component analysis (PCA). The SIFT features are extracted from every other frame in a video shot.

SIFT features with Hessian-Affine detector (Hes-SIFT)

A Hessian-Affine detector [22] is complementary to the Harris-Affine detector. A point (x, y) is a Hessian key-point if the following det$H$ and trace$H$ take an extreme value at (x, y),

$$\text{det}H = I_{xx}I_{yy}(x, y, \sigma_D) - I_{xy}^2(x, y, \sigma_D), \tag{3.9}$$

$$\text{trace}H = I_{xx}(x, y, \sigma_D) + I_{yy}(x, y, \sigma_D), \tag{3.10}$$

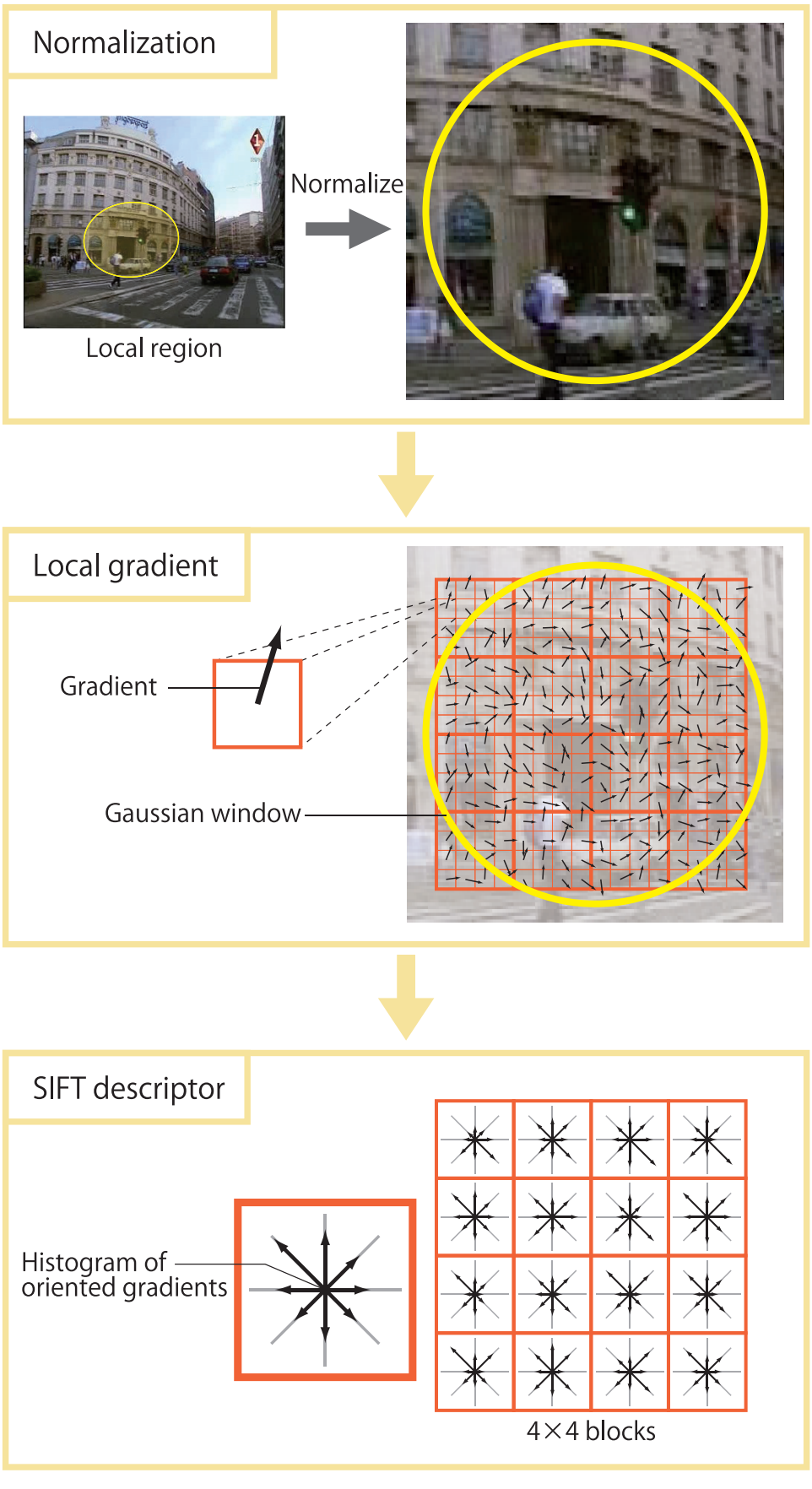

Figure 3.3: SIFT descriptor

where H is an Hessian matrix given by

$$H = \begin{bmatrix} I_{xx}(x, y, \sigma_D) & I_{xy}(x, y, \sigma_D) \\ I_{xy}(x, y, \sigma_D) & I_{yy}(x, y, \sigma_D) \end{bmatrix} \tag{3.11}$$

The combination of several different detectors can improve the robustness against noise. SIFT descriptors are extracted in the same way as SIFT with a Harris-Affine detector. PCA is applied to reduce their dimensions from 128 to 32.

SIFT and hue histogram with dense sampling (Dense-SIFTH)

To capture color information, SIFT features and hue histograms [28] are combined. As a result, 164 dimensional low-level features (which consist of 128-dimension SIFT features and 36-dimension hue histograms) are obtained. PCA is also used to reduce the dimensions to 32. This feature is extracted only from key frames by using dense sampling, which provides a much larger number of low-level features than sparse sampling such as the Harris-Affine and Hessian-Affine detectors.

HOG with dense sampling (Dense-HOG)

32-dimensional histogram of oriented gradients (HOG) are extracted from up to 100 frames per shot by using dense sampling with 2x2 blocks. PCA is applied but dimensions of the HOG features are kept to 32.

LBP with dense sampling (Dense-LBP)

Local Binary Patterns (LBPs) [27] are extracted from up to 100 frames per shot by using dense sampling with 2x2 blocks to capture texture information. We follow the procedure in [27] to extract LBP features. PCA is applied to reduce dimensions from 228 to 32.

3.2.2 Audio Features

MFCC audio features (MFCC)

Mel-frequency cepstral coefficients (MFCCs), which describe the short-time spectral shape of audio frames, are extracted to capture audio informa-

tion. Semantic concepts related to people speaking, talking, and singing can be detected by using MFCCs since MFCCs are effective for speech recognition and audio classification. The 38-dimension audio features consist of 12-dimension MFCCs, 12-dimension Δ MFCCs, 12-dimension $\Delta\Delta$ MFCCs, 1-dimension Δ log-power and 1-dimension $\Delta\Delta$ log-power are extracted. Here, "Δ" means the derivative of the feature.

3.3 Detection Methods

3.3.1 Log-Likelihood Ratio

A detection score of a log-likelihood ratio detector is given by

$$L = \frac{p(X|H=+1)}{p(X|H=-1)}, \tag{3.12}$$

where X is a set of low-level features extracted from a video shot and H is a random variable that takes $+1$ if a targeted semantic concept appears in the shot and otherwise -1.

A probability $p(\cdot|H)$ is estimated by using a Gaussian mixture model (GMM). The probability density function (pdf) of a GMM is given by

$$p(x|\theta) = \sum_{k=1}^{K} w_k \mathcal{N}(x|\mu_k, \Sigma_k), \tag{3.13}$$

where $x \in X$ is a low-level feature, $\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is a set of GMM parameters, K is the number of Gaussian components (vocabulary size), w_k is a mixture coefficient, and $\mathcal{N}(x|\mu_k, \Sigma_k)$ is a Gaussian pdf with a mean vector μ_k and a covariance matrix Σ_k. Here, we assume $X = \{x_i\}_{i=1}^{N}$ is i.i.d. to obtain

$$p(X|\theta) = \prod_{i=1}^{N} p(x_i|\theta). \tag{3.14}$$

EM Algorithm

The expectation-maximization (EM) algorithm [58] is a standard way to find the maximum likelihood (ML) solution for models with latent variables. For each model parameter in θ, we set the derivatives of $\log p(X|\theta)$ to zero.

Note that this is the condition that a ML solution must satisfy. With respect to the mean vector μ_k of a GMM, by setting the derivative of $\log p(X|\theta)$ to zero, we obtain

$$0 = -\sum_{i=1}^{N} \frac{w_k N(x_i|\mu_k, \Sigma_k)}{\sum_{k'} w_{k'} N(x_i|\mu_{k'}, \Sigma_{k'})} \Sigma_k (x_i - \mu_k), \tag{3.15}$$

where c_{ik} is a responsibility (posterior probabilities) given by

$$c_{ik} = \frac{w_k N(x_i|\mu_k, \Sigma_k)}{\sum_{k'} w_{k'} N(x_i|\mu_{k'}, \Sigma_{k'})}. \tag{3.16}$$

By assuming that Σ_k^{-1} is a nonsingular matrix, we obtain

$$\hat{\mu}_k = \frac{1}{C_k} \sum_{i=1}^{N} c_{ik} x_i, \tag{3.17}$$

by multiplying Σ_k^{-1} to Eq (3.15), where C_k is the sum of responsibilities given by

$$C_k = \sum_{i=1}^{N} c_{ik}. \tag{3.18}$$

This shows that the ML solution of the mean vector $\hat{\mu}_k$ is the weighted sum of all vectors in X with weighting factor c_{ik}.

With respect to the covariance matrix Σ_k of a GMM, by setting the derivative of $\log p(X|\theta)$ to zero, we obtain

$$\hat{\Sigma}_k = \frac{1}{C_k} \sum_{i=1}^{N} c_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T. \tag{3.19}$$

This also shows that the ML solution is the weighted sum of the ML solution of a single Gaussian. which has the same form as the corresponding result for a single Gaussian fitted to the data set, but again with each data point weighted by the corresponding posterior probability and with the denominator given by the effective number of points associated with the corresponding component.

For the mixture coefficient w_k, we use a Lagrange multiplier since the sum of the coefficients has to be one. By introducing a Lagrange multiplier

λ, instead of $\log p(X|\theta)$, we maximize

$$\log p(X|\theta) + \lambda \left(\sum_{k'=1}^{K} w_{k'} - 1 \right), \tag{3.20}$$

which gives

$$0 = \sum_{i=1}^{N} \frac{N(x_i|\mu_k, \Sigma_k)}{\sum_{k'} w_{k'} N(x_i|\mu_{k'}, \Sigma_{k'})} + \lambda. \tag{3.21}$$

By taking sum over k, we obtain $\lambda = -N$. The ML solution for w_k is given by

$$w_k = \frac{C_k}{\sum_{k'} C_{k'}}. \tag{3.22}$$

The EM algorithm has two steps: E step and M step. In the E step, responsibilities are evaluated with the current parameters. In the M step, parameters are updated by using responsibilities obtained at E step. The following summarizes the EM algorithm for a GMM.

1. Initialize mean vectors $\hat{\mu}_k$, covariance matrixes $\hat{\Sigma}_k$ and mixture coefficients $\hat{w}_k$. Compute the initial value for the likelihood function.
2. **E step**. Compute the responsibilities as

$$c_{ik} = \frac{\hat{w}_k N(x_i|\hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{k'=1}^{K} \hat{w}_{k'} N(x_i|\hat{\mu}_{k'}, \hat{\Sigma}_{k'})}, \tag{3.23}$$

by using the current parameters.
3. **M step**. Update the parameters as

$$\hat{\mu}_k \leftarrow \frac{1}{C_k} \sum_{i=1}^{N} c_{ik} x_i, \tag{3.24}$$

$$\hat{\Sigma}_k \leftarrow \frac{1}{C_k} \sum_{i=1}^{N} c_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T, \tag{3.25}$$

$$\hat{w}_k \leftarrow \frac{C_k}{\sum_{k'} C_{k'}}, \tag{3.26}$$

where

$$C_k = \sum_{i=1}^{N} c_{ik}, \tag{3.27}$$

by using the current responsibilities.
4. Compute the log likelihood function by

$$\log p(X|\hat{\theta}) = \sum_{i=1}^{N} \log \left\{ \sum_{k=1}^{K} \hat{w}_k N(x_i|\hat{\mu}_k, \hat{\Sigma}_k) \right\}, \tag{3.28}$$

to check for convergence. Return to the step 2 until convergence.

3.3.2 GMM Supervector SVM

The GMM supervector SVM is another powerful model for detecting semantic concepts. A GMM supervector, which is used as an input of a support vector machine, is the concatenation of GMM parameters estimated for each video shot.

MAP Adaptation for GMM

As described above, the GMM parameters are often estimated by using the EM algorithm with the maximum likelihood criterion. However, a set of extracted low-level feature vectors may not be enough to estimate the parameters precisely. In such cases, the alternative way is to use maximum a posteriori (MAP) adaptation. MAP adaptation, a parameter estimation using the MAP criterion, is robust against over-fitting caused by limited data since it uses a prior distribution. A GMM for prior distribution, namely a universal background model (UBM), is first estimated by applying the EM algorithm to all the training data. The UBM presents the feature distribution for the whole database.

The MAP solution for GMM means, namely MAP adaptation, is given by

$$\hat{\mu}_k = \frac{\tau \hat{\mu}_k^{(\mathrm{U})} + \sum_{i=1}^{n} c_{ik} x_i}{\tau + C_k}, \tag{3.29}$$

$$c_{ik} = \frac{w_k g_k(x_i)}{\sum_{k=1}^{K} w_k g_k(x_i)}, \tag{3.30}$$

$$C_k = \sum_{i=1}^{n} c_{ik}, \tag{3.31}$$

$$g_k(x) = \mathcal{N}(x|\hat{\mu}_k^{(\mathrm{U})}, \hat{\Sigma}_k^{(\mathrm{U})}), \tag{3.32}$$

where $X = \{x_i\}_{i=1}^{n}$ is a set of feature vectors extracted from a shot, τ is a predefined hyper-parameter, and $\hat{\theta}^{(\mathrm{U})}$ is the parameter for a universal background model (UBM). The UBM presents how the features are distributed in the general case: therefore, the parameter $\hat{\theta}^{(\mathrm{U})}$ is estimated by using all features in the training set.

GMM Supervector

Finally, GMM supervectors are created for each shot and are given by

$$\phi(X) = \begin{pmatrix} \tilde{\mu}_1 \\ \tilde{\mu}_2 \\ \vdots \\ \tilde{\mu}_K \end{pmatrix}, \tag{3.33}$$

$$\tilde{\mu}_k = \sqrt{w_k^{(\mathrm{U})}} \left(\Sigma_k^{(\mathrm{U})}\right)^{-\frac{1}{2}} \hat{\mu}_k, \tag{3.34}$$

where $\hat{\mu}_k$ is an adapted mean vector, and $\theta^{(\mathrm{U})}$ is the GMM parameter for the UBM. The dimension of GMM supervectors is Kd, where K is the number of Gaussian components and d is the dimension of the low-level feature vector. The weighted sum of Mahalanobis distances between corresponding Gaussian pairs is obtained by calculating the squared Euclidean distance between two GMM supervectors. RBF kernels are used for SVM classification:

$$k(X, X') = e^{-\gamma \|\phi(X) - \phi(X')\|^2}, \tag{3.35}$$

where γ is a kernel parameter.

Support Vector Machine

A support vector machine (SVM) is trained for each semantic concept and for each type of features. An SVM is a binary classifier which focuses on modeling the boundary between two classes [59, 60]. A hyperplane which separates d-dimensional data into two classes is trained by an SVM. To improve the classification accuracy, SVMs introduce a kernel feature space which maps a sample into a higher dimensional space where the data becomes linearly separable.

Given a N number of training samples $\{x_i, y_i\}_{i=1}^{N}$, each sample $x_i \in \mathbb{R}^d$, and a class label $y_i \in \{-1, 1\}$, all the hyperplanes in $\mathbb{R}^d$ are parametrized by a vector w and a constant b ,

$$w^T x + b = 0 \tag{3.36}$$

where w is a vector orthogonal to the hyperplane. For a given (w, b), a decision (score) function of an SVM is given by

$$f(x) = \text{sign}(w^T x + b) \tag{3.37}$$

which classifies the training samples. A *canonical hyperplane* is defined to separate training samples from the hyperplane by a distance of 1, while satisfying the following two conditions:

$$w^T x_i + b \geq +1 \quad \text{when} \quad y_i = +1, \tag{3.38}$$

$$w^T x_i + b \leq -1 \quad \text{when} \quad y_i = -1. \tag{3.39}$$

This is simplified as,

$$y_i(w^T x_i + b) \geq 1 \tag{3.40}$$

Then, the magnitude of w is normalized to obtain geometric distance from the hyperplane to a sample. The distance is obtained as

$$d((w, b), x_i) = \frac{y_i(w^T x_i + b)}{\|w\|} \geq \frac{1}{\|w\|} \tag{3.41}$$

By minimizing $\|w\|$ a hyperplane that maximizes the geometric distance

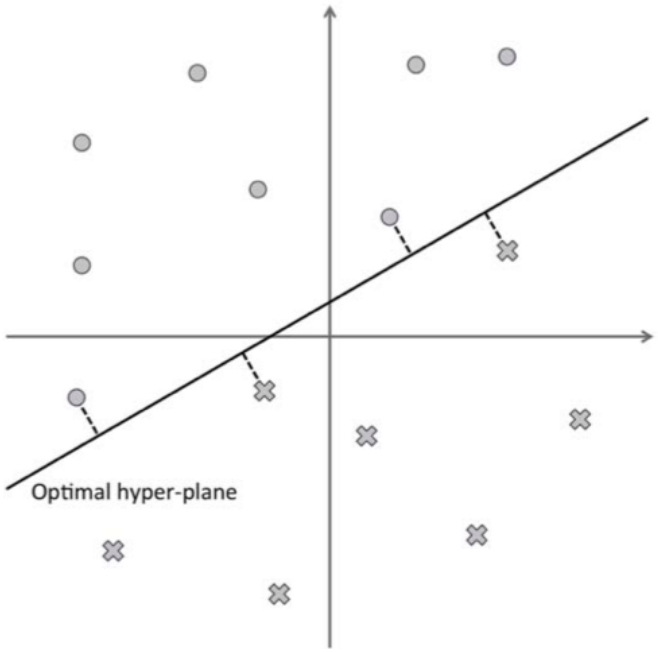

Figure 3.4: A hyperplane that maximizes the margin.

is obtained to the closets samples as shown in Figure 3.4.

Lagrange multipliers [59, 61] are introduced to solve the problem as:

$$\begin{aligned} &\text{minimize:} && W(\alpha) = -\sum_i \alpha_i + \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j x_i^T x_j \\ &\text{subject to:} && \textstyle\sum_i y_i \alpha_i = 0 \\ & && 0 \le \alpha_i \le C \quad (\forall i) \end{aligned}$$

where α_i is a parameter of an SVM, and C is a regulalization parameter to tune the margins for the hyperplane. Above problem is reduced as follows by introducing a matrix $H_{ij} = y_i y_j x_i^T x_j$,

$$\begin{aligned} &\text{minimize:} && W(\alpha) = -\alpha^T \mathbf{1} + \frac{1}{2}\alpha^T H \alpha \\ &\text{subject to:} && \alpha^T y = 0 \\ & && 0 \le \alpha \le C\mathbf{1} \quad (\forall i) \end{aligned}$$

This is solved by using quadratic programming techniques. Finally, the

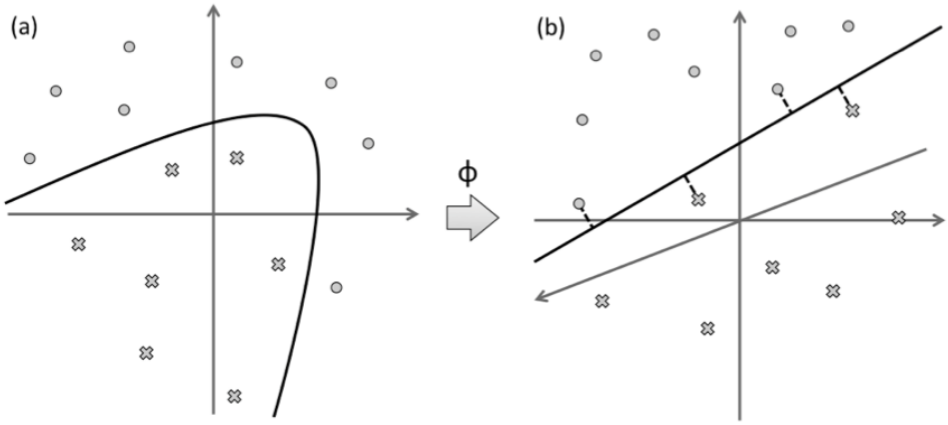

Figure 3.5: An example of a kernel trick. Data is not linearly separable in (a). The data becomes linearly separable in (b) after transforming the input data to a new feature space.

optimal hyperplane is given by

$$w = \sum_i \alpha_i y_i x_i \tag{3.42}$$

which shows that w is a linear combination of the training samples of $\alpha_i \neq 0$. Training samples which have $\alpha_i > 0$ are called support vectors and are used to calculate the decision function.

If the data is not linearly separable, a kernel trick is used to transform a d-dimensional input vector into a higher d'-dimensional vector $\phi(x)$ as shown in Figure 3.5.

By using the kernel trick, Eq.(3.42) is replaced as follows,

$$w = \sum_i \alpha_i y_i \phi(x_i)$$

where $\phi(x_i)$ is a mapping functionby. The decision function is obtained by

$$f(x) = \sum_i \alpha_i y_i k(x_i, x) + b \tag{3.43}$$

where k is a kernel function given as a inner product of the mapped samples. The same process to find the optimal hyperplane is applied in a high

dimensional feature space after transforming the problem in this way. Since a kernel function is often non-linear, classification accuracy is expected to be improved by using it.

3.4 Experiments

3.4.1 Experimental Conditions

In our experiments, the TRECVID 2009 High-Level Feature Extraction Dataset and TRECVID 2011 Semantic Indexing Dataset are used to evaluate our semantic indexing system. Each dataset is divided into a training set and a test set. The evaluation measure is Mean Average Precision (Mean AP) among 20 semantic concepts (Figure 3.6). Mean AP is the mean of AP for each semantic concept given by

$$\text{AP} = \frac{1}{R}\sum_{r=1}^{N}\text{Pr}(r)\text{Rel}(r), \tag{3.44}$$

where R is the number of positive samples (appearances), N is the number of testing samples, $\text{Pr}(r)$ is the precision at the rank of r and $\text{Rel}(r)$ takes a value of one if the r-th shot is positive; otherwise, it takes zero. The number of appearances is shown in Figure 3.6.

3.4.2 Detection Accuracy

Figure 3.7 shows Mean APs for different numbers of Gaussian components for SIFT features (Harris-affine detector) with a log-likelihood ratio detector (SIFT-LR) or a GMM-supervector SVM (SIFT-SVM), and MFCC features with a LR detector (MFCC-LR) or a GMM-supervector SVM (MFCC-SVM). Detailed AP for each semantic concept is shown in Figure 3.8. As can be seen, SIFT-SVM outperforms the others for 19 of 20 semantic concepts, and MFCC-SVM outperforms the others for "Singing". This shows that visual information is often more important than audio information for detecting semantic concepts.

When both of SIFT and MFCC are used, Mean AP is improved from 0.141 (for SIFT-SVM) to 0.173. This shows that MFCC captures complementary information to SIFT. Table 3.1 compares the following seven fusion methods:

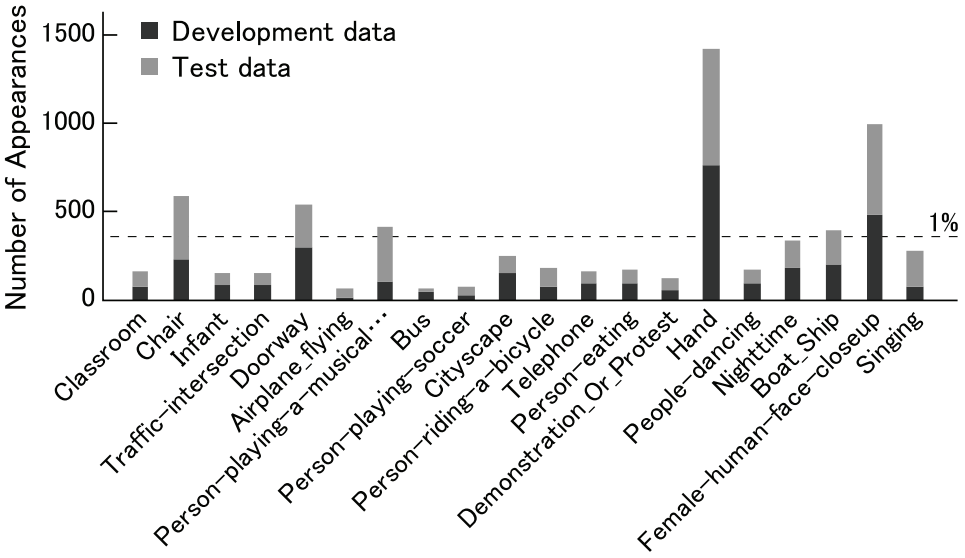

Figure 3.6: Number of appearances of semantic concepts

Fusion 1: SIFT-LR + SIFT-SVM
Fusion 2: MFCC-LR + MFCC-SVM
Fusion 3: SIFT-LR + MFCC-LR
Fusion 4: SIFT-SVM + MFCC-SVM
Fusion 5: SIFT-LR + SIFT-LR + MFCC-LR + MFCC-SVM
Fusion 6: SIFT-SVM + Hes-SIFT-SVM
Fusion 7: SIFT-LR + SIFT-LR + MFCC-LR + MFCC-SVM + Hes-SIFT-SVM
The best performance is obtained by Fusion 7.

3.4.3 Multi-frame Feature Extraction

Figure 3.8 shows Mean AP for different number of image frames, from which SIFT features are extracted. SIFT-SVM shows better performance than BoW especially when the number of image frames is larger than 15 since GMMs used in SIFT-SVM can estimate the distribution of SIFT features more precisely than than histogram used in BoW.

3.4.4 Error Analysis

Figure 3.9 shows resulting APs for Har-SIFT, Hes-SIFT, Dense-SIFTH and MFCC. We conclude that the four types of features are complementary to

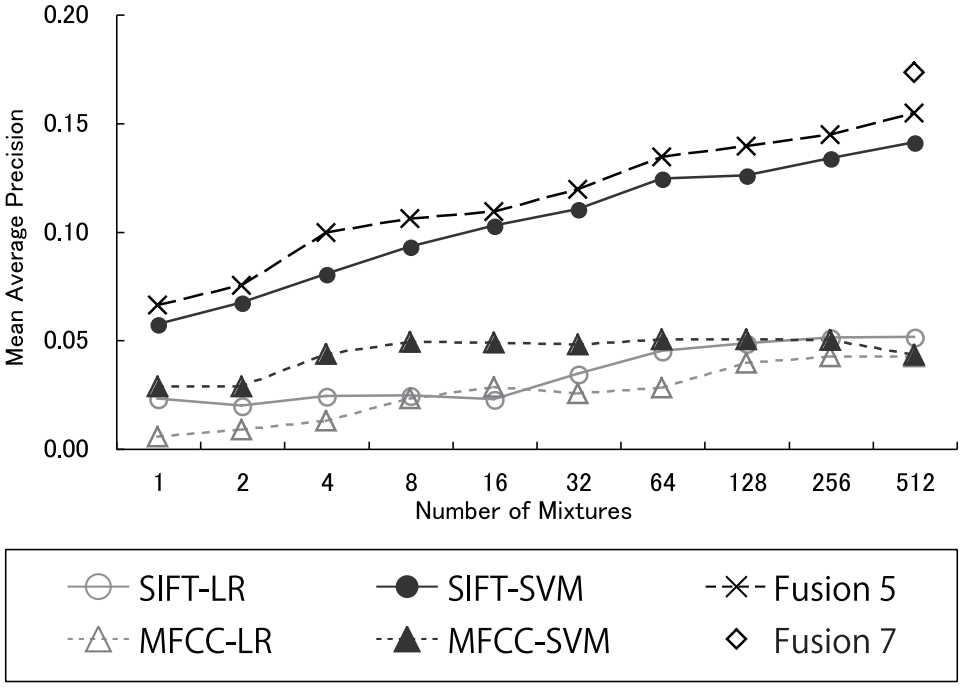

Figure 3.7: Comparison of Mean APs for different schemes.

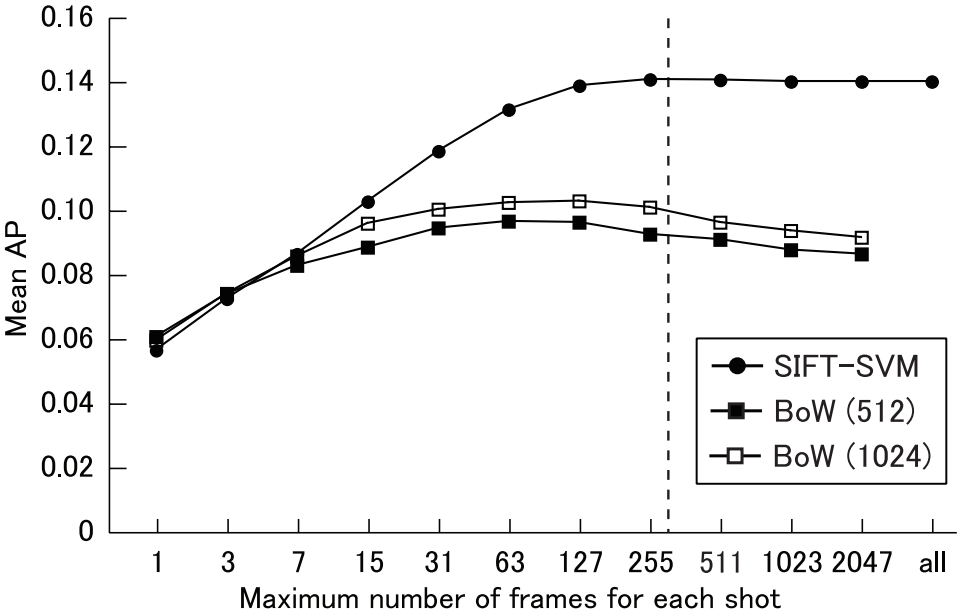

Figure 3.8: Mean APs with different numbers of frames. The dotted line indicates the average number of frames in a shot.

Table 3.1: Mean APs for different schemes ($K = 512$). R denotes a result of Randomization test ($p = 0.05$).

Method	R	Mean AP	Singing	Female.	M.I.
Fusion 8	14	0.186	0.233	0.271	0.149
Fusion 7	13	0.173	0.213	0.266	0.132
Fusion 6	10	0.155	0.038	0.206	0.048
Fusion 5	9	0.154	0.216	0.261	0.102
Fusion 4	8	0.145	0.151	0.205	0.022
SIFT-SVM	8	0.141	0.032	0.192	0.038
Fusion 1	7	0.138	0.037	0.193	0.063
$SIFT_{hes}$-SVM	7	0.129	0.025	0.163	0.044
BoW(Multiframe)	6	0.097	0.015	0.115	0.041
Fusion 3	3	0.064	0.113	0.100	0.096
BoW	2	0.060	0.004	0.049	0.020
Fusion 2	2	0.050	0.163	0.066	0.016
SIFT-LR	0	0.051	0.015	0.045	0.053
MFCC-SVM	0	0.043	0.126	0.020	0.010
MFCC-LR	0	0.042	0.095	0.067	0.028

each other: Har-SIFT is effective for detecting rigid objects and close-up objects, Hes-SIFT is effective for detecting objects with a background, Dense-SIFTH is effective for detecting scenes, and MFCC is effective for detecting human events and actions (Figure3.10). The followings are our detailed observations and error analysis regarding each type of feature:

Har-SIFT

Har-SIFT achieved the highest InfAP (with the exception of the fusion methods) for 12 concepts: “Bus” (Figure 3.11), “Female Human Face Closeup” (Figure 3.12), “Ground Vehicles”, “Singing”, “Doorway”, “Hand”, “Animal”, “Old People”, “Cheering”, “Flowers”, “Telephones”, and “Sitting Down”. Since Har-SIFT features are extracted from corner points, rigid objects such as cars and buses were successfully detected. For detecting people and animals, corner points for their eyes helped improving performance. However, we still have some errors in classifying fine-grained categories, e.g. “Truck” and “Bus”, “Female” and “Male”.

Hes-SIFT

Hes-SIFT achieved the highest InfAP for 7 concepts: "Airplane Flying" (Figure 3.13), "Running" (Figure 3.14), "Vehicle", "Walking", "Throwing", "Bicycling", and "Classroom". By introducing the Hessian-Affine detector, objects on a flat background, for example, an airplane in the sky and a car on a highway, were successfully detected. However, it is still difficult to detect an object from a complex background since our modeling can not separate foreground and background.

Dense-SIFTH

Dense-SIFTH achieved the highest InfAP for 9 concepts: "Cityscape" (Figure 3.15), "Mountain" (Figure 3.16), "Demonstration Or Protest", "Nighttime", "Boat Ship", "Dancing", "Car Racing", "Swimming", and "Explosion Fire". The hue histogram of Dense-SIFTH detected scenes with few colors, e.g. explosion fire with red and yellow, nighttime with black, and a mountain with green.

MFCC

MFCC achieved the highest InfAP for two concepts: "Dark-skinned People" (Figure 3.17), and "Asian People". For "Singing" (Figure 3.18), AP was improved by 5.2% by combining MFCC with the three visual features. Since MFCC captured speech and music in video, people speaking or singing were detected even if they are partially occluded by some objects. This shows that MFCC captured complementary information with the visual features. However, false detections occur when video has a background music that is not related to the video contents.

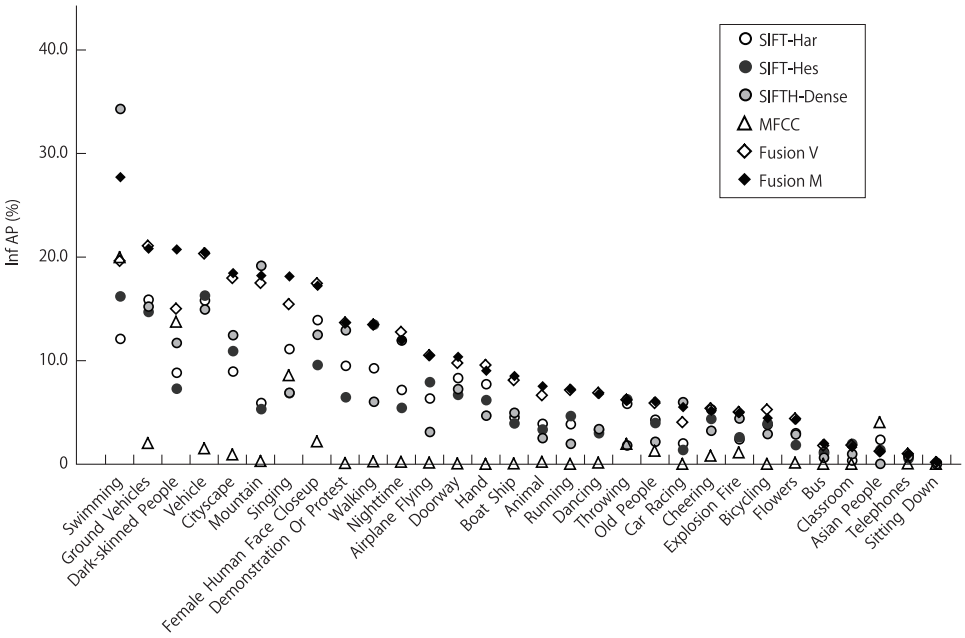

Figure 3.9: InfAP by semantic concepts on the TRECVID 2011 Semantic Indexing Dataset.

3.4.5 Comparison with Other Methods

Our results in the semantic indexing task on TRECVID 2011 and 2012 are shown in Figure 3.19 and 3.20. The best result was 0.173 and 0.321 of Mean InfAP, respectively, which is ranked first in the TRECVID 2011 and 2012 official runs. This shows that our multi-modal system based on the GMMs is powerful and accurate compared with the others.

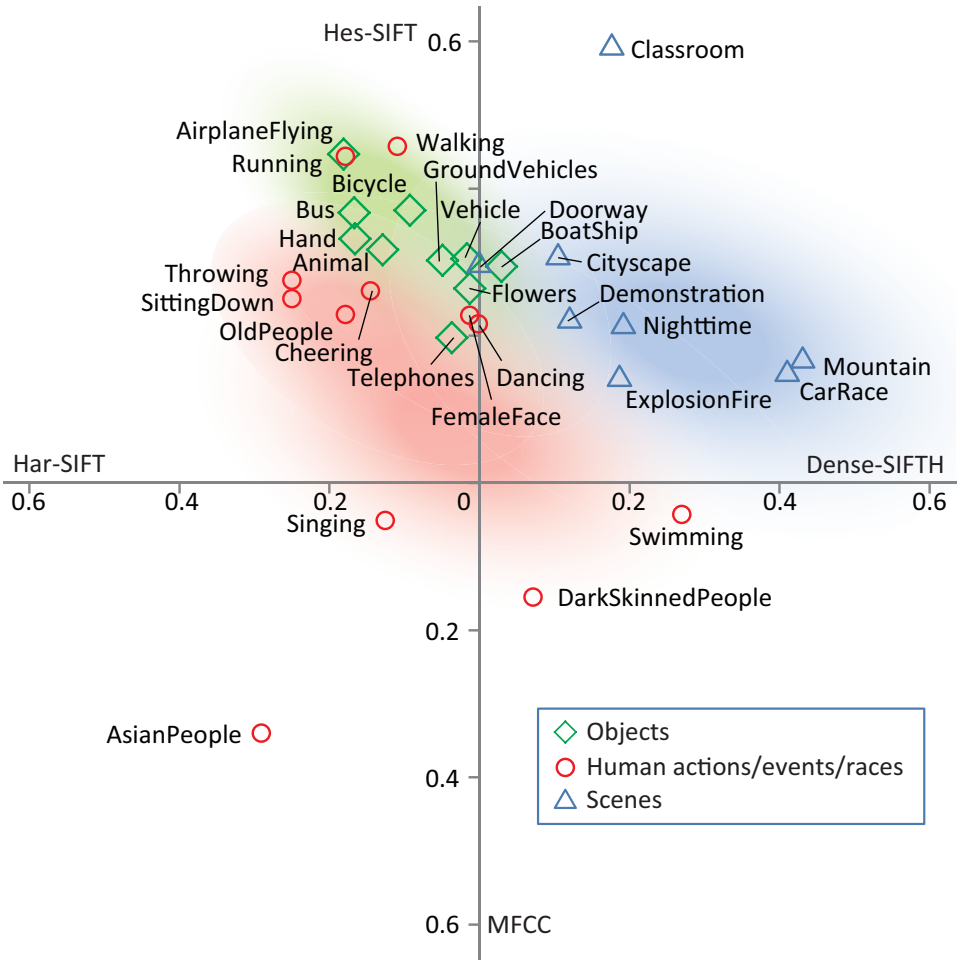

Figure 3.10: 2D visualization of 4D Average Precisions (APs) for Har-SIFT, Hes-SIFT, Dense-SIFTH and MFCC. To compare the effectiveness of each type of feature, the four AP values are normalized so that their sum is one.

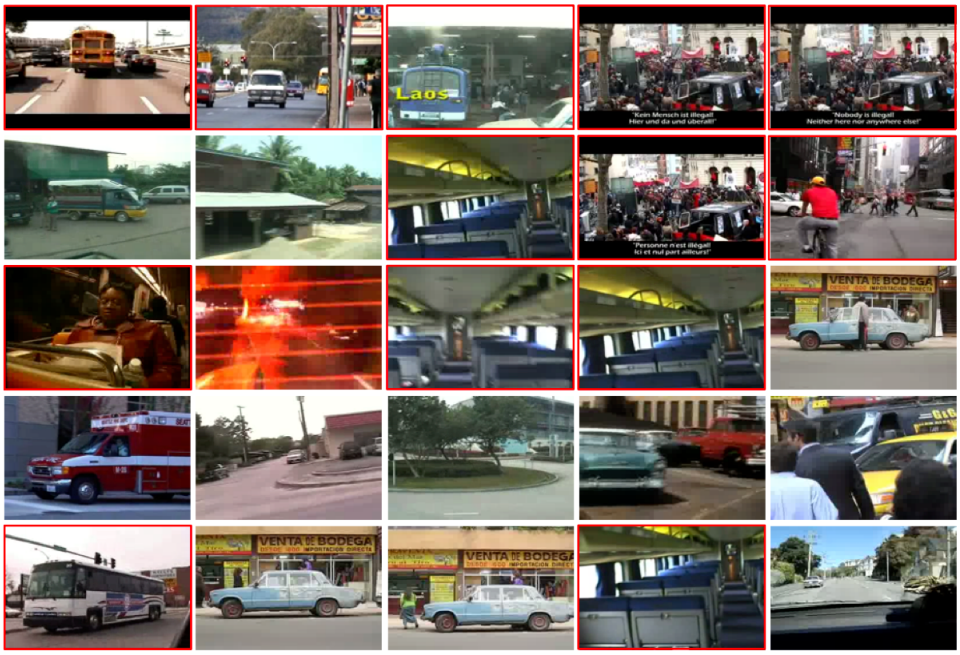

Figure 3.11: Top 25 video shots for "Bus". Correct shots are marked in red.

Figure 3.12: Top 25 video shots for "Female Human Face Closeup". Correct shots are marked in red.

Figure 3.13: Top 25 video shots for "Airplane Flying". Correct shots are marked in red.

Figure 3.14: Top 25 video shots for "Running". Correct shots are marked in red.

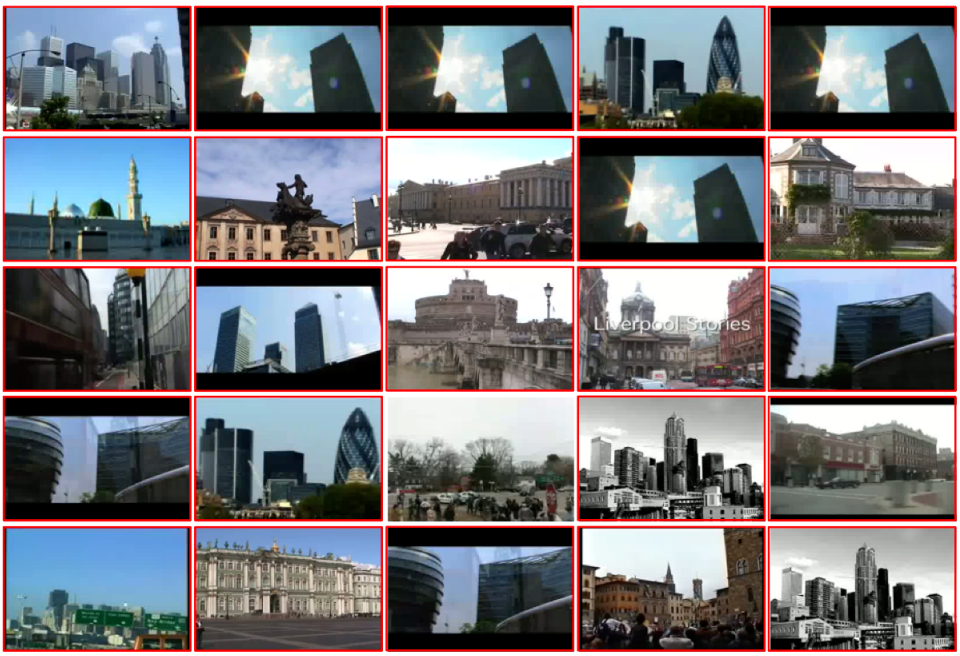

Figure 3.15: Top 25 video shots for "Cityscape". Correct shots are marked in red.

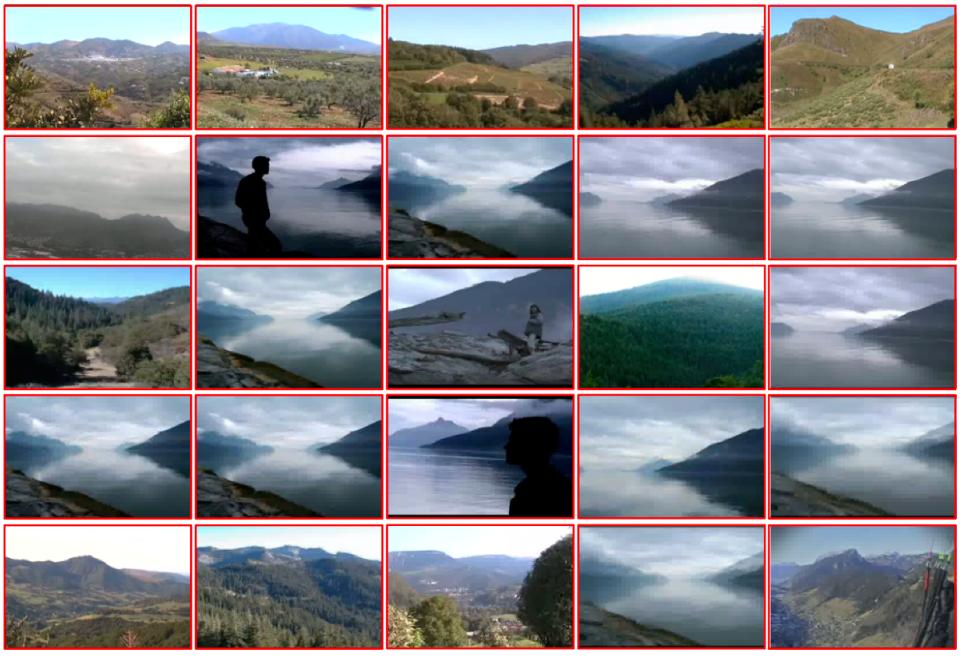

Figure 3.16: Top 25 video shots for "Mountain". Correct shots are marked in red.

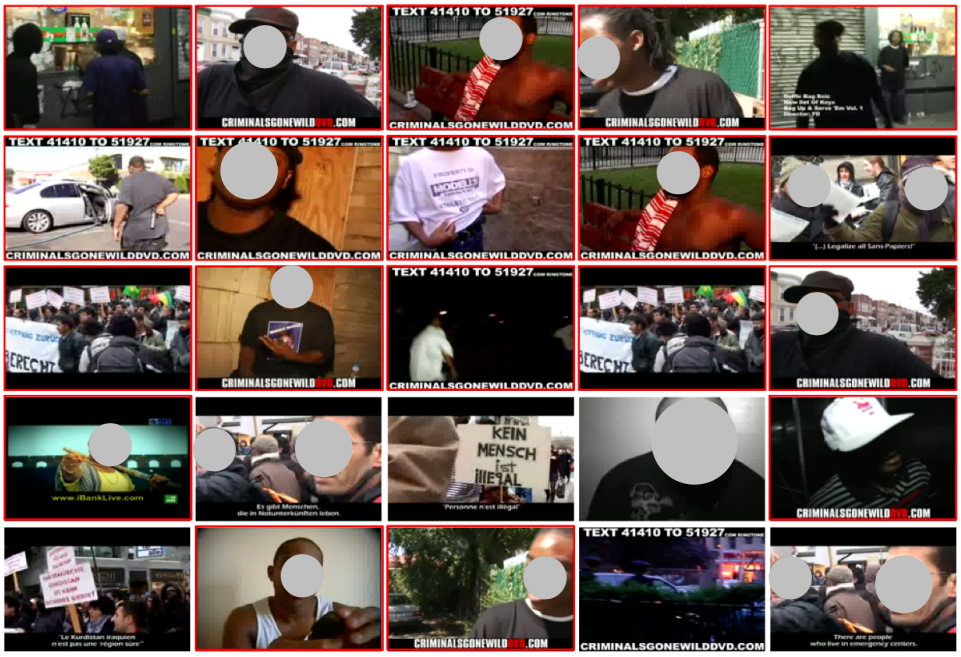

Figure 3.17: Top 25 video shots for "Dark-skinned People". Correct shots are marked in red.

Figure 3.18: Top 25 video shots for "Singing". Correct shots are marked in red.

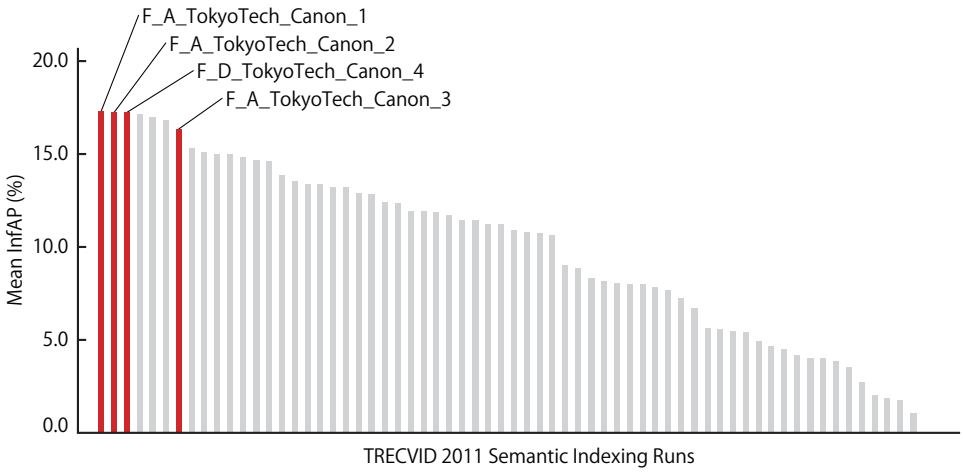

Figure 3.19: Comparison with other methods on TRECVID 2011.

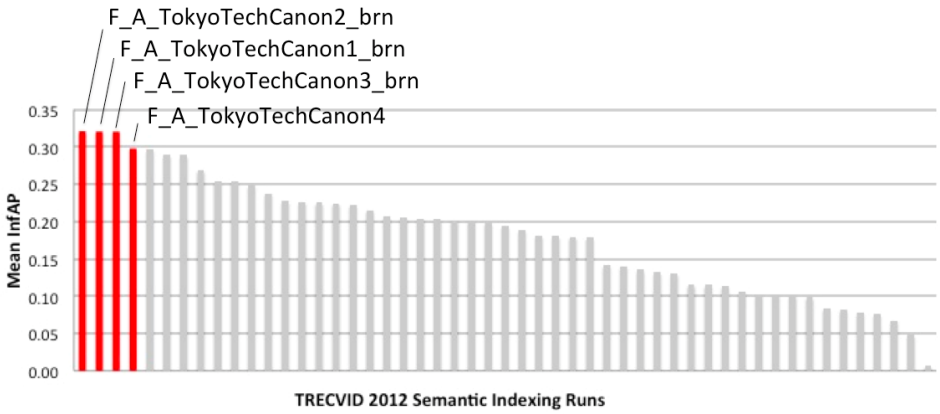

Figure 3.20: Comparison with other methods on TRECVID 2012.

3.5 Conclusion

In this chapter, we proposed a high-performance semantic indexing system using Gaussian mixture model (GMM) supervectors with the six audio and visual features. Our best result on the TRECVID 2011 and 2012 was Mean AP of 0.173 and 0.312, respectively. However, since our system, especially the GMM-parameter-estimation step, is computationally expensive, we need to improve the speed of the system.

Chapter 4

Tree-structured Gaussian Mixture Models

4.1 Overview

In this chapter, we describe a fast maximum a posteriori (MAP) adaptation technique for a GMM-supervectors-based video semantic indexing system. As shown in Chapter 3, 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.

4.2 Tree-structured GMMs

Let $X = \{x_i\}_{i=1}^{n}$ be a set of (one type of) low-level features extracted from a video shot. A probability distribution function (pdf) of a Gaussian mixture

model (GMM) is given by

$$p(x|\theta) = \sum_{k=1}^{K} w_k \mathcal{N}(x|\mu_k, \Sigma_k), \tag{4.1}$$

where x is a low-level feature vector, $\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is a set of parameters, K is the number of mixture components, w_k is a mixture coefficient. $\mathcal{N}(\cdot|\mu_k, \Sigma_k)$ is a Gaussian pdf with a mean vector μ_k and a covariance matrix Σ_k.

As described in Chapter 3, MAP adaptation, a parameter estimation using the MAP criterion, is robust against over-fitting caused by limited data since it uses a prior distribution. The MAP solution gives the following equations:

$$\hat{\mu}_k = \frac{\tau \hat{\mu}_k^{(\mathrm{U})} + \sum_{i=1}^{n} c_{ik} x_i}{\tau + \sum_{i=1}^{n} c_{ik}}, \tag{4.2}$$

$$c_{ik} = \frac{w_k^{(\mathrm{U})} g_k^{(\mathrm{U})}(x_i)}{\sum_{k=1}^{K} w_k^{(\mathrm{U})} g_k^{(\mathrm{U})}(x_i)}, \tag{4.3}$$

$$g_k^{(\mathrm{U})}(x) = \mathcal{N}(x|\mu_k^{(\mathrm{U})}, \Sigma_k^{(\mathrm{U})}), \tag{4.4}$$

where n is the number of feature vectors, and $\hat{\mu}_k^{(\mathrm{U})}$ is a mean vector of UBM, $g_k^{(\mathrm{U})}$ is a Gaussian component, c_{ik} is a *responsibility* of a Gaussian component g_k for a feature vector x_i, which is the posterior probability of x_i being at k-th Gaussian component, and τ is a predefined hyper-parameter.

A tree structure of Gaussian components that makes calculation of Eq. (4.3) efficient is constructed from the UBM. Fig. 4.1 shows an example of a tree-structured GMM. Each leaf node corresponds to a Gaussian component of the UBM, and each other node has a single Gaussian obtained by combining corresponding Gaussian pdfs of descendant nodes. This tree structure is constructed by top-down clustering of Gaussian components. For a given set of Gaussian components $G = \{g_1, g_2, \cdots, g_K\}$, we define a combined single

Gaussian $\mathcal{G}(G)$ by

$$\mathcal{G}(G) \stackrel{\text{def}}{=} \mathcal{N}(\cdot|\bar{\mu}, \bar{\Sigma}), \tag{4.5}$$

$$\bar{\mu} = \frac{1}{K}\sum_{k=1}^{K}\mu_k, \tag{4.6}$$

$$\bar{\Sigma} = \frac{1}{K}\sum_{k=1}^{K}(\Sigma_k + \mu_k\mu_k^{\mathsf{T}}) - \bar{\mu}\bar{\mu}^{\mathsf{T}}. \tag{4.7}$$

To find a pair of Gaussian components which are close to each other, a distance measure between them is needed. The sum of Kullback-Leibler divergence from g_k to $g_{k'}$ and that of from $g_{k'}$ to g_k is used for this measurement as follows:

$$d(g_k, g_{k'}) = \int g_k(x)\log\frac{g_k(x)}{g_{k'}(x)}dx + \int g_{k'}(x)\log\frac{g_{k'}(x)}{g_k(x)}dx \tag{4.8}$$

$$\begin{aligned}=&\frac{1}{2}\Big((\mu_k - \mu_{k'})^{\mathsf{T}}(\Sigma_k^{-1} + \Sigma_{k'}^{-1})(\mu_k - \mu_{k'}) \\ &+ \mathrm{tr}(\Sigma_{k'}^{-1}\Sigma_k) + \mathrm{tr}(\Sigma_k^{-1}\Sigma_{k'}) - 2d\Big).\end{aligned} \tag{4.9}$$

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 (which only has a root node) is two and that for the second layer is three, the resulting tree will be designed as shown in Fig. 4.1. In this case, a tree with a depth of three (including the leaf layer) is obtained. This tree-structured GMM is denoted as

$$\mathcal{T}_{(2,3)} = (V, E, G_{\text{TREE}}), \tag{4.10}$$

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, a node at the t-th layer of a tree $\mathcal{T}_{(P_1,P_2,\cdots,P_T)}$ has, at most, P_t child nodes.

The node pdfs $g^{(v)}$ of a tree $\mathcal{T}_{(P_1,P_2,\cdots,P_T)}$ are created by the following algorithm. The basic idea is to apply hierarchical k-means clustering for Gaussian components. Note that $G_{\text{UBM}} = \{g_1^{(\text{U})}, g_2^{(\text{U})}, \cdots, g_K^{(\text{U})}\}$ is a set of mixture components of the UBM, $G^{(v)}$ is a subset of G_{UBM} corresponding to node v, and $g^{(v)}$ is a Gaussian pdf for node v.

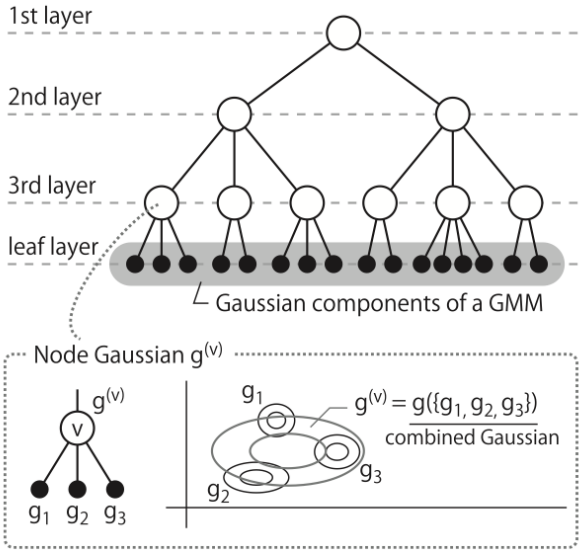

Figure 4.1: An example of a tree-structured GMM $\mathcal{T}_{(2,3)}$.

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

2. (Initialization for k-means) Dequeue $(v, G^{(v)})$. Let $\{c_p\}_{p=1}^{P}$ be the child nodes of node v. Select the initial cluster centers $g^{(c_p)}$ from the given set of Gaussian components $G^{(v)}$, where the min-max selection method is used instead of random selection. The min-max selection method is explained in Appendix.

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

$$G^{(c_p)} \leftarrow \{g \in G^{(v)} \mid p = \operatorname*{argmin}_{p'} d(g, g^{(c_{p'})})\}. \tag{4.11}$$

   Update $g^{(c_p)}$ by using Eq. (4.5) as follows:

$$g^{(c_p)} \leftarrow \mathcal{G}(G^{(c_p)}), \tag{4.12}$$

Repeat this step until the following sum of distance converges:

$$D = \sum_{p=1}^{P} \sum_{g \in G^{(c_p)}} d(g, g^{(c_p)}). \tag{4.13}$$

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 ℓ for each $g_k^{(\mathrm{U})} \in G^{(v)} \subset G_{\mathrm{UBM}}$ and set

$$g^{(\ell)} = g_k^{(\mathrm{U})}. \tag{4.14}$$

For the initialization for k-means clustering (Step 2 in the tree-construction algorithm,we use the min-max selection method. This method is known to provides better initial values than random selection. This method first selects from $G^{(v)}$ a node set whose nodes are distant from each other, and then sets a cluster center at an internal dividing point between node v and each of the selected nodes.

2-1) Choose the mixture component $\tilde{g}^{(c_1)}$ that has the largest distance to $g^{(v)}$, i.e.,

$$\tilde{g}^{(c_1)} = \operatorname*{argmax}_{g \in G^{(v)}} \; d(g, g^{(v)}). \tag{4.15}$$

2-2) For $p = 2, \cdots, P$, choose $\tilde{g}^{(c_p)}$ from the rest of mixture components which belong to the node v and not yet assigned to any child node, i.e.,

$$\tilde{g}^{(c_p)} = \operatorname*{argmax}_{g \in G_{p-1}^{(v)}} \min_{1 \le p' < p} d(g, \tilde{g}^{(c_{p'})}), \tag{4.16}$$

where $G_{p-1}^{(v)} = G^{(v)} \setminus \{\tilde{g}^{(c_1)}, \cdots, \tilde{g}^{(c_{p-1})}\}$. If $G_{p-1}^{(v)}$ is an empty set, the child node is deleted from the tree.

2-3) For $p = 1, 2, \cdots, P$, set the parameters of child Gaussian pdfs $g^{(c_p)}$ as

follows:

$$g^{(c_p)} \leftarrow \mathcal{N}(\cdot|\bar{\mu}, \bar{\Sigma}), \tag{4.17}$$

$$\bar{\mu} = \alpha\tilde{\mu}^{(c_p)} + (1-\alpha)\mu^{(v)}, \tag{4.18}$$

$$\begin{aligned}\bar{\Sigma} =& \alpha(\tilde{\Sigma}^{(c_p)} + \tilde{\mu}^{(c_p)}(\tilde{\mu}^{(c_p)})^{\mathsf{T}}) \\ &+ (1-\alpha)(\Sigma^{(v)} + \mu^{(v)}(\mu^{(v)})^{\mathsf{T}}) \\ &- \bar{\mu}\bar{\mu}^{\mathsf{T}},\end{aligned} \tag{4.19}$$

where $0 \leq \alpha \leq 1$ is a weight parameter to mix the selected pdf $\tilde{g}^{(c_p)} = \mathcal{N}(\cdot|\tilde{\mu}^{(c_p)}, \tilde{\Sigma}^{(c_p)})$ and their parent pdf $g^{(v)} = \mathcal{N}(\cdot|\mu^{(v)}, \Sigma^{(v)})$.

4.3 Fast MAP Adaptation

A fast MAP adaptation technique which estimates c_{ik} in Eq. (4.3) efficiently by using a tree-structured GMM is explained in the following. For a constructed tree-structured GMM $\mathcal{T}_{(P_1,P_2,\cdots,P_T)}$, node weights are first defined as follows:

a) For each leaf node ℓ, set

$$w^{(\ell)} = w_k^{(\mathrm{U})}, \tag{4.20}$$

if $g^{(\ell)} = g_k^{(\mathrm{U})} \in G_{\mathrm{UBM}}$.

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

$$w^{(v)} = \sum_{c \in C(v)} w^{(c)}, \tag{4.21}$$

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:

$$V_A \leftarrow \bigcup_{v \in V_A} C(v), \tag{4.22}$$

where $C(v)$ is a set of child nodes of the node v. Here, $C(\ell) = \{\ell\}$ is used for leaf nodes ℓ to keep the leaf nodes active.

3. Consider a GMM for active node V_A given by

$$p(x|V_A) = \sum_{v \in V_A} \tilde{w}^{(v)} g^{(v)}(x), \tag{4.23}$$

where

$$\tilde{w}^{(v)} = \frac{w^{(v)}}{\sum_{v \in V_A} w^{(v)}}. \tag{4.24}$$

Calculate

$$c_i^{(v)} = \frac{\tilde{w}^{(v)} g^{(v)}(x_i)}{\sum_{v \in V_A} \tilde{w}^{(v)} g^{(v)}(x_i)} = \frac{w^{(v)} g^{(v)}(x_i)}{\sum_{v \in V_A} w^{(v)} g^{(v)}(x_i)}. \tag{4.25}$$

4. Keep a node v active if $c_i^{(v)}$ is larger than the predetermined threshold c_{TH}, i.e.

$$V_A \leftarrow \{v \in V_A \mid c_i^{(v)} > c_{\text{TH}}\}. \tag{4.26}$$

5. If all nodes in V_A are leaf nodes, output

$$\hat{c}_{ik} = \begin{cases} c_i^{(\ell)} & (\ell \in V_A,\ g^{(\ell)} = g_k^{(\text{U})}) \\ 0 & (\text{otherwise}) \end{cases}. \tag{4.27}$$

Otherwise, return to Step 2.

Since the observed $\hat{c}_{ik}$ for non-active nodes is 0, the sum in Eq. (4.2) can be calculated for non-zero $\hat{c}_{ik}$ only. Our fast MAP adaptation requires $O(n \log K)$ time which improves $O(nK)$ time for the basic MAP adaptation. Moreover, calculation speed and levels of approximation can be controlled by selecting the threshold c_{TH} in $0 < c_{\text{TH}} \leq 1$. Note that the same c_{ik} as given

by Eq. (4.3) is obtained when c_{TH} is set to 0 (because all leaf nodes will be active at the final step).

Finally, GMM supervectors are created for each shot in the same way as Chapter 3. Note that the proposed method can be used for creating the Fisher vectors [35, 36].

SVMs for the four types of features described in Section 3.2 are combined by calculating the weighted sum of detection scores as

$$f(X) = \sum_{\substack{\mathrm{F} \in \{\text{SIFT-Har, SIFT-Hes,} \\ \text{SIFTH-D, MFCC}\}}} \beta_{\mathrm{F}} f_{\mathrm{F}}(X), \tag{4.28}$$

where β_{F} is a combination weight and $f_{\mathrm{F}}(X)$ is a detection score for the feature F. The combination weights are optimized for each semantic concept by cross validation.

4.4 Experiments

4.4.1 Database and Task

Our experiments were conducted on the TRECVID 2010 dataset [64, 13]. The dataset consists of 400 hours of Internet archive videos with creative commons licenses. The shot boundaries and key-frame images 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.

We evaluated our system on the TRECVID 2010 Semantic Indexing benchmark. The task is to detect the 30 semantic concepts (including objects, events and scenes) listed in Table 4.1. They are considered useful for video searching.

The annotated labels for the 30 concepts are also provided with the video data (including testing videos for evaluation). The labels for training data are created using a collaborative annotation system [65]; however, some of the training shots are still unlabeled. It was assumed that the unlabeled samples are negative since the annotation system is based on an active learning method that requires shots appearing to be positive samples to be annotated preferentially. On the other hand, labels for testing videos are attached on

Table 4.1: The 30 target semantic concepts in the TRECVID 2010 dataset

Airplane Flying	Female Human Face Closeup
Animal	Flowers
Asian People	Ground Vehicles
Bicycling	Hand
Boat ship	Mountain
Bus	Nighttime
Car Racing	Old People
Cheering	Running
Cityscape	Singing
Classroom	Sitting down
Dancing	Swimming
Dark-skinned People	Telephones
Demonstration Or Protest	Throwing
Doorway	Vehicle
Explosion Fire	Walking

Table 4.2: The average numbers of extracted features.

Feature	# of features per shot
SIFT-Har	19,536
SIFT-Hes	18,986
SIFTH-Dense	30,000
MFCC	5,160

the basis of the submission pool of TRECVID 2010, which allows precise estimation of average precision.

The evaluation measures are Mean average precision (Mean AP) and the calculation time of the testing phase. The AP is estimated by using a method called inferred average precision (Inf AP), proposed in [66].

4.4.2 Experimental Conditions

SiftGPU [67] and Mikolajczyk's implementation [22] were used for SIFT feature extraction. MFCC features were extracted by using a speech recognition toolkit HTK [69]. The average numbers of features per shot are summarized in Table 4.2.

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

τ in the MAP adaptation was set to 20.0, which is the standard value of the toolkit HTK. Parameter γ in the RBF kernel was $\gamma = \bar{d}^{-1}$, where $\bar{d}$ is pre-calculated average distance between two GMM supervectors in training data. SVMs were trained for each semantic concept by using the libSVM implementation [70]. Combination weights for the fusion in Eq. (4.28) were optimized by using two-fold cross validation on training data.

For tree-structured GMMs, the optimal tree structure $\mathcal{T}_{\text{opt}}$ was selected as

$$\mathcal{T}_{\text{opt}} = \underset{\mathcal{T} \in \mathcal{S}}{\operatorname{argmin}} \, \text{CT}(\mathcal{T}), \tag{4.29}$$

where $\text{CT}(\mathcal{T})$ is calculation time when the tree $\mathcal{T}$ is used and

$$\mathcal{S} = \{\mathcal{T}_{(P_1,P_2,\cdots,P_T)} \mid 1 \leq T \leq 5, 1 \leq P_t \leq 5\}. \tag{4.30}$$

The trees $\mathcal{T}_{(4,4,5)}$, $\mathcal{T}_{(4,5,5)}$, $\mathcal{T}_{(3,4,4,5)}$ and $\mathcal{T}_{(3,3)}$ were selected for SIFT-Har, SIFT-Hes, SIFTH-Dense and MFCC, respectively. Parameter α in Eq. (4.17) 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 high. Experiments using different thresholds were also conducted (see Subsection 4.4.3).

In the experiments, calculation time was measured by using a single core of Intel Xeon 2.93 GHz CPU. Calculation time without feature extraction time is reported since some features were pre-extracted by using GPUs. The average feature extraction time per shot was 0.38 sec by using a GPU NVIDIA Tesla M2050.

4.4.3 Results

Mean Inf APs

Table 4.3 and Table 4.4 summarizes obtained Inf APs and Mean Inf AP for each types of low-level features and two fusion methods: visual fusion and multi-modal fusion. The visual fusion is a combination of three types of visual features (SIFT-Har, SIFT-Hes, and SIFTH-Dense). The multi-modal fusion combines the MFCC in addition to the visual features. As a result, we can see that the Mean Inf APs using tree-structured GMMs are comparable to those using no trees. Some example video shots for training and testing

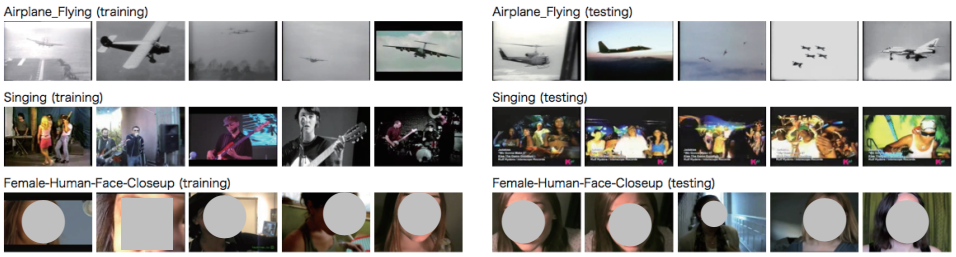

Figure 4.2: Example video shots for training and testing sets. The top 5 results obtained by using our system (multi-modal fusion) are shown in the right side of the figure.

sets are shown in Fig. 4.2.

Calculation time

Table 4.5 lists calculation times for MAP adaptation using different features and different trees. The results for binary trees ($\mathcal{T}_{\text{binary}}$) are also listed in the table. The calculation speed when the optimal tree is used on average 4.2 times faster than when trees were not used; that is, calculation time was reduced by 76.2%.

Fig. 4.3 shows calculation time for each step in the testing phase of the proposed semantic indexing system. The testing cost was reduced, on average, by 56.6% by using tree-structured GMMs. The second and third highest costs were for the PCA projection and the SVM prediction (including calculation of kernels). The SVM prediction can be speed up by using linear kernels instead of RBF kernels. To avoid decrease in detection accuracy, a possible compromise is to use linear kernels for roughly ranking shots and re-evaluate high-ranked shots by using RBF kernels.

Analysis of estimation error

Estimation errors of c_{ik} were evaluated from the mean absolute error (MAE), given as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}|\hat{c}_{ik} - c_{ik}|, \tag{4.31}$$

where $\hat{c}_{ik}$ and c_{ik} are given by Eq. (4.27) and Eq. (4.3), respectively. The MAE for SIFTH-Dense was 0.32 on average (note that $0 \leq \text{MAE} \leq 2$). Al-

Table 4.3: Resulting inferred average precisions (Inf APs) for each semantic concept and for each method. Mean Inf APs on the testing set and Mean APs on a two-fold cross-validation split of the training data are also shown.

Semantic concept	SIFT-Har		SIFT-Hes		SIFTH-Dense		MFCC	
	No tree	$\mathcal{T}_{\text{opt}}$	No tree	$\mathcal{T}_{\text{opt}}$	No tree	$\mathcal{T}_{\text{opt}}$	No tree	$\mathcal{T}_{\text{opt}}$
Airplane_Flying	0.064	0.064	0.080	0.078	0.032	0.030	0.001	0.001
Animal	0.039	0.041	0.034	0.035	0.026	0.020	0.002	0.002
Asian_People	0.024	0.029	0.014	0.015	0.001	0.002	**0.041**	**0.041**
Bicycling	0.039	0.041	0.040	0.033	0.029	0.026	0.000	0.000
Boat_Ship	0.046	0.044	0.040	0.041	0.050	0.049	0.000	0.000
Bus	0.012	0.012	0.011	0.013	0.007	0.009	0.000	0.000
Car_Racing	0.021	0.019	0.014	0.013	**0.060**	0.054	0.000	0.000
Cheering	**0.053**	0.051	0.044	0.045	0.033	0.037	0.008	0.008
Cityscape	0.090	0.098	0.109	0.110	0.125	0.108	0.009	0.009
Classroom	0.004	0.005	0.020	**0.022**	0.010	0.010	0.000	0.000
Dancing	0.034	0.036	0.030	0.028	0.034	0.033	0.001	0.001
Dark-skinned_People	0.089	0.088	0.073	0.071	0.118	0.133	0.138	0.139
Demonstration_Or_Protest	0.095	0.095	0.065	0.069	0.130	0.121	0.001	0.001
Doorway	0.084	0.082	0.068	0.067	0.073	0.068	0.000	0.001
Explosion_Fire	0.025	0.025	0.026	0.025	0.045	0.043	0.011	0.011
Female-Human-Face-Closeup	0.139	0.124	0.096	0.105	0.125	0.121	0.021	0.021
Flowers	0.030	0.028	0.019	0.017	0.029	0.028	0.001	0.001
Ground_Vehicles	0.159	0.165	0.148	0.150	0.153	0.151	0.021	0.020
Hand	0.078	0.073	0.062	0.073	0.047	0.055	0.000	0.000
Mountain	0.059	0.055	0.053	0.054	0.192	**0.194**	0.003	0.003
Nighttime	0.072	0.073	0.055	0.054	0.120	0.113	0.002	0.002
Old_People	0.043	0.045	0.040	0.041	0.022	0.023	0.013	0.011
Running	0.039	0.041	0.047	0.045	0.020	0.018	0.000	0.000
Singing	0.112	0.105	0.069	0.074	0.069	0.068	0.086	0.090
Sitting_Down	0.002	0.001	0.001	0.001	0.001	0.001	0.000	0.000
Swimming	0.121	0.131	0.162	0.164	**0.343**	0.339	0.199	0.199
Telephones	0.008	0.010	0.006	0.006	0.008	0.009	0.001	0.000
Throwing	0.059	0.059	0.063	0.063	0.019	0.016	0.019	0.020
Vehicle	0.158	0.150	0.163	0.172	0.150	0.146	0.015	0.014
Walking	0.093	0.103	0.135	0.138	0.061	0.060	0.002	0.002
Mean InfAP	0.063	0.063	0.060	0.061	0.071	0.070	0.020	0.020
Mean AP on validation set 1	0.078	0.078	0.081	0.082	0.105	0.107	0.028	0.028
Mean AP on validation set 2	0.084	0.085	0.092	0.091	0.111	0.111	0.028	0.027

though we have estimation errors of c_{ik} in the fast MAP adaptation algorithm, they can be cancelled when the distance in Eq. (3.35) is calculated since the same errors occur in training and testing phases.

Effect of using different thresholds

Table 4.6 lists the results obtained using different thresholds c_{TH} for the fast MAP adaptation. The number of leaf nodes that are active (at least once in Eq. (4.22)) and MAE are also listed in the table.

As c_{TH} gets higher, the calculation time shortens, but Mean Inf AP was decreased when $c_{\text{TH}} = 0.1$ 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.

Table 4.4: Resulting inferred average precisions (Inf APs) for each semantic concept and for each method. Mean Inf APs on the testing set and Mean APs on a two-fold cross-validation split of the training data are also shown.

Semantic concept	Visual fusion		Multi-modal fusion	
	No tree	$\mathcal{T}_{\text{opt}}$	No tree	$\mathcal{T}_{\text{opt}}$
Airplane_Flying	0.105	0.105	0.105	**0.117**
Animal	0.068	0.073	**0.076**	**0.076**
Asian_People	0.012	0.009	0.012	0.009
Bicycling	0.045	**0.056**	0.045	**0.056**
Boat_Ship	**0.085**	0.084	**0.085**	0.084
Bus	0.018	0.016	**0.021**	0.016
Car_Racing	0.040	0.040	0.056	0.043
Cheering	0.052	0.051	0.052	0.051
Cityscape	0.180	0.177	**0.185**	0.179
Classroom	0.015	0.011	0.017	0.021
Dancing	**0.068**	0.067	**0.068**	0.067
Dark-skinned_People	0.151	0.159	**0.208**	0.203
Demonstration_Or_Protest	**0.137**	0.132	**0.137**	0.132
Doorway	0.098	0.097	**0.104**	0.098
Explosion_Fire	**0.050**	0.047	**0.050**	0.047
Female-Human-Face-Closeup	0.169	0.175	0.173	**0.178**
Flowers	0.043	**0.044**	0.043	**0.044**
Ground_Vehicles	**0.211**	0.210	0.208	0.206
Hand	**0.092**	0.089	0.090	0.090
Mountain	0.180	0.169	0.182	0.164
Nighttime	0.127	**0.133**	0.120	0.132
Old_People	0.059	0.058	0.061	**0.063**
Running	0.073	**0.077**	0.073	**0.077**
Singing	0.154	0.158	0.182	**0.188**
Sitting_Down	0.002	0.003	0.003	**0.004**
Swimming	0.173	0.186	0.278	0.276
Telephones	0.010	0.012	0.010	**0.018**
Throwing	0.062	0.065	0.062	**0.066**
Vehicle	**0.205**	0.200	**0.205**	0.200
Walking	0.134	0.142	0.135	**0.143**
Mean InfAP	0.094	0.095	**0.102**	**0.102**
Mean AP on validation set 1	0.147	0.148	0.153	**0.154**
Mean AP on validation set 2	0.158	0.158	**0.162**	0.161

Effect of using different tree structures

Fig. 4.4 shows calculation time obtained using different tree structures. The tree of $\mathcal{T}_{(3,4,4,5)}$ was the best in terms of calculation time. We can see that the tree should not be too deep to improve the speed of MAP adaptation. Fig. 4.5 shows MAE obtained using different tree structures. MAE can be reduced by changing the tree structure. However, we conclude that any tree structures will not be the cause for decreasing final performance since there was no decrease in Mean Inf AP even in the case of MAE $= 0.53$ in Table 4.6.

Comparison With Other Methods

Fig. 4.6 compares Mean Inf APs obtained in the above-described experiment with those values obtained by the other methods used at TRECVID 2010. Our fusion methods got better results than the best result reported at TRECVID 2010 (0.900).

Fig. 4.7 shows the results of significance test obtained by applying par-

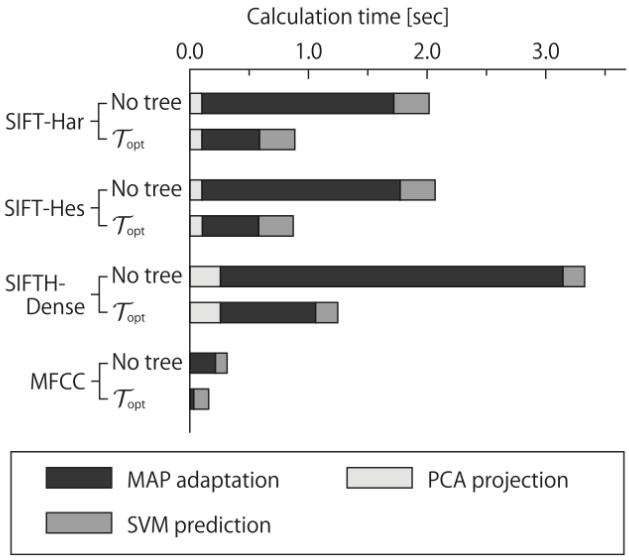

Figure 4.3: Calculation time for each step (The lower bars for each feature show the time in the case that the optimized tree was used)

Table 4.5: Calculation time (sec) for MAP adaptation. Calculation time was measured by using a single core of Intel Xeon 2.93 GHz CPU.

Feature	No tree	$\mathcal{T}_{\text{opt}}$	$\mathcal{T}_{\text{binary}}$
SIFT-Har	1.62	0.49	0.98
SIFT-Hes	1.67	0.48	1.00
SIFTH-Dense	2.89	0.81	1.89
MFCC	0.22	0.03	0.08

tial randomization test ($p < 0.05$). The muti-modal fusion was significantly better than the visual fusion. Our method performed better than the other methods in TRECVID 2010 for semantic concepts related to human and human actions such as “Singing” and “Dancing” since we used audio features. However, there was no significant difference between the muti-modal fusion and the top result in TRECVID 2010. This result shows that the performance can be improved by combining a larger number of visual features since the top ranked methods in TRECVID 2010 used more than 10 types of visual features.

Although our final goal is to develop a generic methods for automatically assigning semantic concepts to videos, overall performances are still low compared with that of human annotation. One future challenge is detection

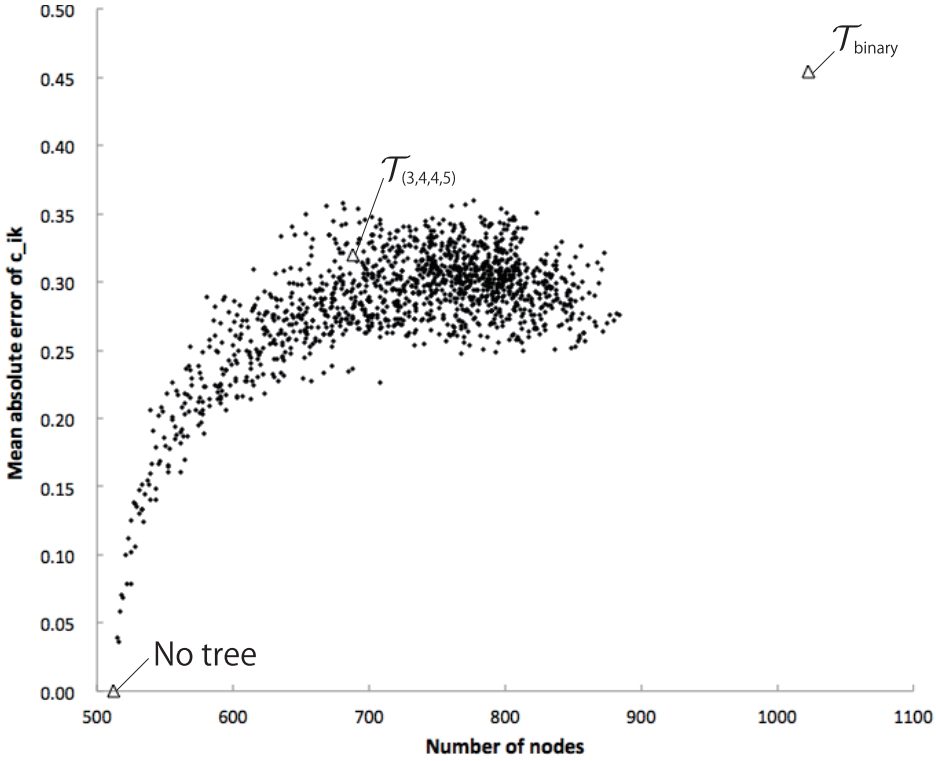

Figure 4.4: Mean absolute error (MAE) of c_{ik} obtained using different tree structures (the SIFTH-Dense feature and $c_{\text{TH}} = 0.001$ were used). 1,364 trees of depth at most 5 that have at most 5 children per node and the binary tree are tested. All MAE were less than 0.05.

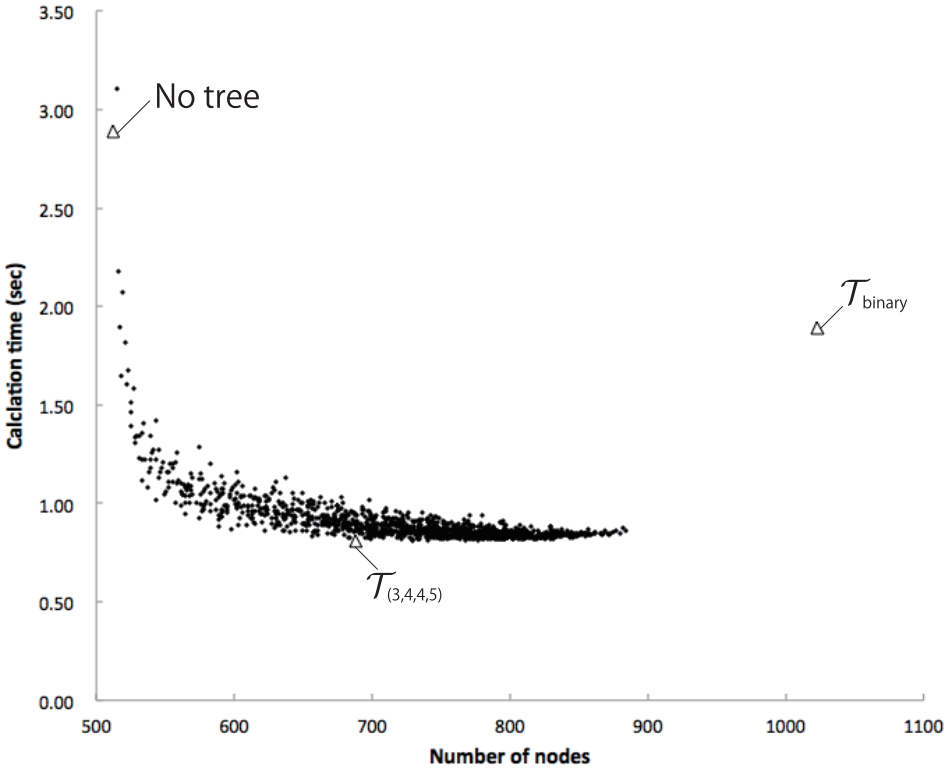

Figure 4.5: Calculation time obtained using different tree structures (the SIFTH-Dense feature and $c_{\text{TH}} = 0.001$ were used). 1,364 trees of depth at most 5 that have at most 5 children per node and the binary tree are tested. $\mathcal{T}_{(3,4,4,5)}$ was the best tree and was selected as the optimized tree.

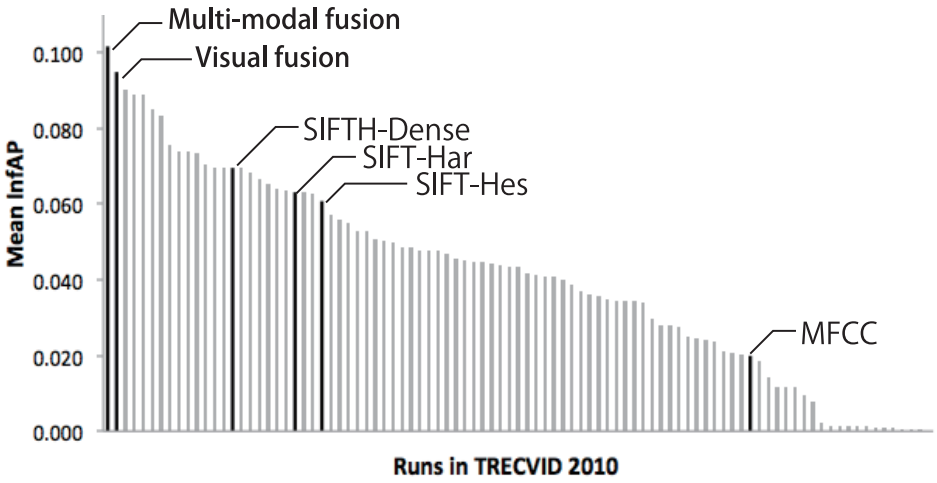

Figure 4.6: Comparison of Mean Inf AP with runs of the TRECVID 2010.

Table 4.6: Comparison of Mean Inf AP, calculation time (sec) for MAP adaptation, number of leaf nodes $|V_A|$ and Mean absolute error (MAE) of c_{ik} by using different thresholds c_{TH} for the SIFTH-Dense feature.

c_{TH}	Mean Inf AP	Calc. time	$\lvert V_A \rvert$	MAE
0.001	0.695	0.81	17.0	0.32
0.01	0.699	0.68	11.2	0.53
0.1	0.660	0.59	7.3	0.80
0.5	0.641	0.53	5.4	0.98

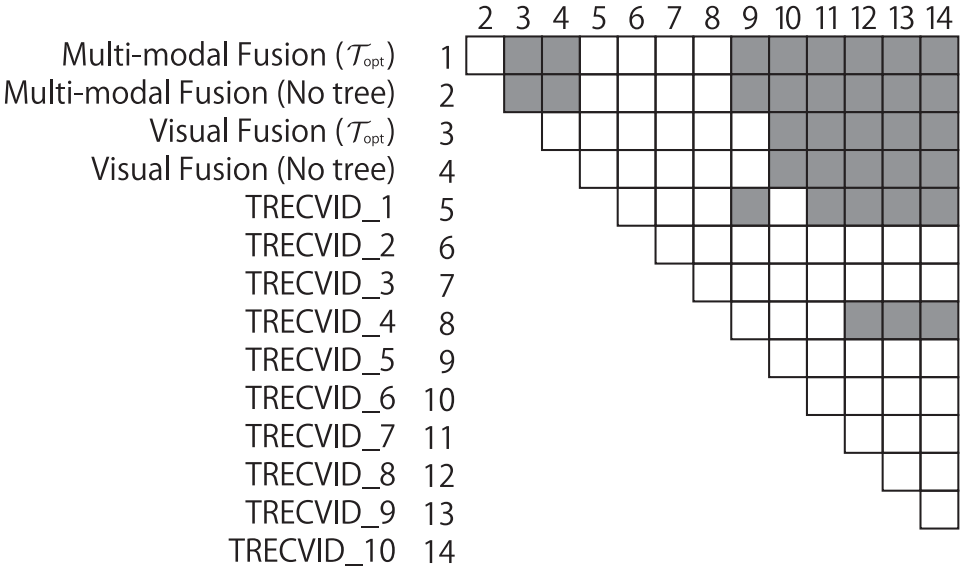

Figure 4.7: Results of partial randomization test ($p < 0.05$). Significant differences among top 10 runs in TRECVID 2010 and our fusion methods are shown. A black cell shows that there is significant difference between two methods.

of many kinds of semantic concepts; however, we have to consider which concepts are really useful for applications of video search.

4.5 Conclusion

A fast and accurate semantic indexing system using fast MAP adaptation and GMM supervectors was proposed. A tree-structured GMM was constructed to quickly calculate posterior probabilities for each mixture component of a GMM. The calculation time for MAP adaptation was reduced by 76.2% from the conventional method, while high detection performance was main-

tained. Our future work will focus on a GPU implementation of the fast MAP adaptation and feature extraction.

Chapter 5

q-Gaussian Mixture Models

5.1 Overview

As described in the previous chapters, Gaussian mixture models (GMMs) which extend the bag-of-visual-words (BoW) to a probabilistic framework are effective for image and video semantic indexing. Recently, the q-Gaussian distribution, which is derived in the non-extensive statistics, has been shown to be useful for representing patterns in many complex systems in physics such as fractals and cosmology. In this chapter, we extend our GMM-based system to q-Gaussian mixture models (q-GMMs), which are mixture models of q-Gaussian distributions, for image and video semantic indexing.

The procedure of the proposed image and video semantic indexing based on q-Gaussian mixture models (q-GMMs) is shown in Fig. 5.1. First, low-level features (e.g. SIFT features) are extracted from image/video data. Second, a q-GMM for a background model is estimated from low-level features in training data. Finally, each image is represented by a histogram of low-level features in which the background model is used as a visual codebook.

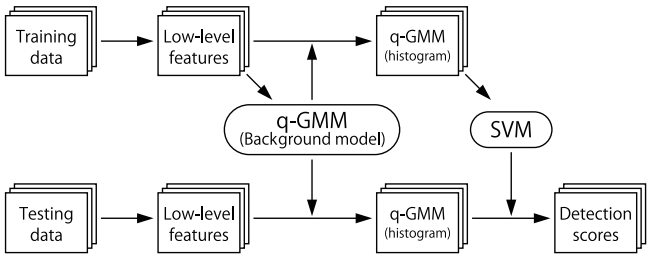

Figure 5.1: The framework of image and video semantic indexing using q-Gaussian mixture models.

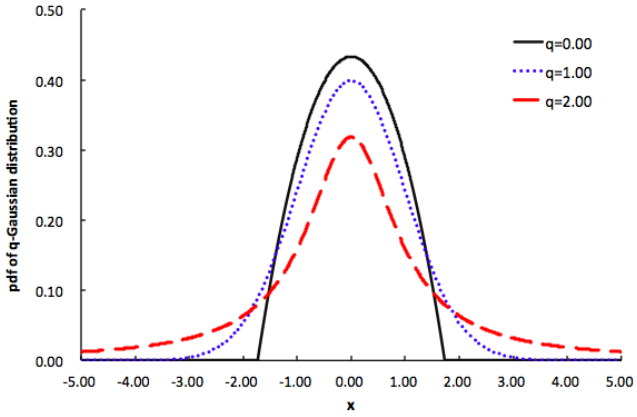

Figure 5.2: The q-Gaussian distributions. The (normal) Gaussian distribution is obtained when $q = 1$. The tail of a q-Gaussian distribution is longer than that of a Gaussian distribution when $q > 1$.

5.2 q-Gaussian Mixture Models

The q-Gaussian distribution, which has a parameter q to control its tail-heaviness, is derived by maximizing Tsallis q-entropy S_q given by

$$S_q = -\frac{1}{1-q}\left(1 - \int p(x)^q dx\right). \tag{5.1}$$

The Boltzmann-Gibbs (BG) entropy is obtained from Tsallis q-entropy S_q for $q \to 1$. The probability density function of the q-Gaussian distribution $\mathcal{N}_q$ is

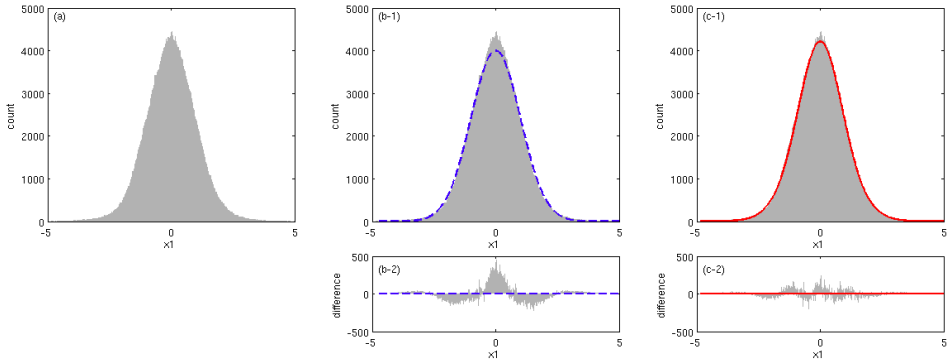

Figure 5.3: (a): Standardized histogram of the first elements of standardized SIFT descriptors. 1 million low-level descriptors are randomly sampled from training data of PASCAL VOC 2010 dataset. (b-1), (b-2): A fitting result by a Gaussian distribution and its residuals. (c-1), (c-2): A fitting result by a q-Gaussian distribution ($q = 1.12$) and its residuals.

given by

$$\mathcal{N}_q(x \mid \mu, \Sigma) = \begin{cases} \dfrac{1}{Z_q}\left(1 - \dfrac{1-q}{3-q}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)^{\dfrac{1}{1-q}}, \\ \qquad \text{if } (x-\mu)^T\Sigma^{-1}(x-\mu) < \dfrac{3-q}{1-q}, \\ 0, \qquad \text{otherwise}, \end{cases} \tag{5.2}$$

where μ is a mean vector, Σ is a covariance matrix, and Z_q is a normalizing constant to make the integral over Eq. (5.2) to 1. Note that the q-Gaussian distribution asymptotically approaches to the Gaussian distribution when $q \to 1$.

Fig. 5.2 shows the shape of q-Gaussian distributions for some q-values. The q-Gaussian distribution has a longer tail than the Gaussian distribution when $q > 1$. The long-tailed distribution obtained for $q > 1$ is expected to be effective for representing complexly correlated data. Fig. 5.3 shows a histogram of the first values of 1 million low-level descriptors with fitting results. The q-Gaussian distribution is more suitable than the Gaussian distribution to represent distribution of the low-level descriptors.

To further improve the expressiveness of the q-Gaussian distribution, we introduce a mixture model of q-Gaussian distributions, namely a q-Gaussian

mixture model (q-GMM), by

$$p_q(x \mid \theta) = \sum_{k=1}^{K} w_k \mathcal{N}_q(x \mid \mu_k, \Sigma_k), \tag{5.3}$$

where K is the number of mixtures, w_k is a mixture coefficient, and $\theta = \{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ is a set of q-GMM parameters.

5.3 Training q-GMM for a Background Model

From a set of low-level features $X = \{x_i\}_{i=1}^{N}$ in training data, we estimate q-GMM parameters for a background model which is used instead of a codebook for BoW. We propose the expectation maximization (EM) algorithm for q-GMMs as follows.

E-step

Evaluate posterior probabilities c_{ik} as follows:

$$c_{ik} = \frac{\hat{w}_k \mathcal{N}_q(x_i \mid \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{k=1}^{K} \hat{w}_k \mathcal{N}_q(x_i \mid \hat{\mu}_k, \hat{\Sigma}_k)}. \tag{5.4}$$

M-step

To derive the parameter-update rules for the M-step, we introduce a Q-function given by

$$Q(\theta) = \log \prod_i p_q(x_i \mid \theta) + \lambda \left(1 - \sum_{k'} w_{k'}\right) \tag{5.5}$$

where p_q is a pdf of a q-GMM defined by Eq. (5.3). A lagrangian multiplier λ is introduced to obtain w_k such that

$$\sum_{k'} w_{k'} = 1. \tag{5.6}$$

The derivations of the Q-function for each parameter are given by

$$\frac{\partial}{\partial w_k}Q(\theta) = \sum_i \frac{\mathcal{N}_q(x_i \mid \mu_k, \Sigma_k)}{\sum_{k'} w_{k'}\mathcal{N}_q(x_i \mid \mu_{k'}, \Sigma_{k'})} - \lambda \tag{5.7}$$

$$= \frac{1}{w_k}\sum_i c_{ik} - \lambda, \tag{5.8}$$

$$\frac{\partial}{\partial \mu_k}Q(\theta) = \sum_i \frac{a_{ik} w_k \mathcal{N}_q(x_i \mid \mu_k, \Sigma_k)}{\sum_{k'} w_{k'}\mathcal{N}_q(x_i \mid \mu_{k'}, \Sigma_{k'})}\Sigma_k^{-1}(x_i - \mu_k) \tag{5.9}$$

$$= \Sigma_k^{-1}\sum_i a_{ik}c_{ik}(x_i - \mu_k), \tag{5.10}$$

$$\frac{\partial}{\partial \Sigma_k}Q(\theta) = \frac{1}{2}\sum_i \frac{w_k \mathcal{N}_q(x_i \mid \mu_k, \Sigma_k)}{\sum_{k'} w_{k'}\mathcal{N}_q(x_i \mid \mu_{k'}, \Sigma_{k'})}\Sigma_k^{-2}\left(a_{ik}(x_i - \mu_k)(x_i - \mu_k)^T - \Sigma_k\right) \tag{5.11}$$

$$= \frac{1}{2}\Sigma_k^{-2}\sum_i c_{ik}\left(a_{ik}(x_i - \mu_k)(x_i - \mu_k)^T - \Sigma_k\right), \tag{5.12}$$

where

$$a_{ik} = \frac{2}{3 - q - (1-q)(x_i - \mu_k)^T\Sigma_k^{-1}(x_i - \mu_k)}. \tag{5.13}$$

The parameter-update rules for the M-step are obtained by setting the derivatives of Q to zero. For mixture coefficients w_k, we obtain

$$\hat{w}_k = \frac{C_k}{\sum_{k=1}^K C_k}, \tag{5.14}$$

from Eq. (5.8) where $C_k = \sum_i c_{ik}$.

However, $\hat{\mu}_k$ and $\hat{\Sigma}_k$ can not be obtained from Eqs. (5.10) and (5.12) analytically since μ_k and Σ_k appear in a_{ik}. Our preliminary experiments show that numerically solving those equations by the steepest descent (SD) method is time-consuming and it's difficult to optimize step size in SD in a high-dimensional space. Hence we assume a_{ik} is a constant A for all i and

k, and analytically solve the equations as

$$\hat{\mu}_k = \frac{1}{C_k} \sum_{i=1}^{N} c_{ik} x_i, \tag{5.15}$$

$$\hat{\Sigma}_k = \frac{A}{C_k} \sum_{i=1}^{N} c_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T. \tag{5.16}$$

We determine the value of A so that the expectation of $\hat{\Sigma}_k$, $\mathbb{E}[\hat{\Sigma}_k]$, is equal to Σ_k, a covariance matrix of a q-Gaussian distribution $\mathcal{N}_q(\cdot \mid \mu_k, \Sigma_k)$. The expectation of $\hat{\Sigma}_k$ is calculated as

$$\mathbb{E}\left[\hat{\Sigma}_k\right] = \mathbb{E}\left[\frac{A}{C_k} \sum_{i=1}^{N} c_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T\right] \tag{5.17}$$

$$= \frac{A}{C_k} \sum_{i=1}^{N} c_{ik} \mathbb{V}[x_i] \tag{5.18}$$

$$= A \frac{3-q}{5-3q} \Sigma_k. \tag{5.19}$$

where we use the fact

$$\mathbb{V}[x_i] = \frac{3-q}{5-3q} \Sigma_k. \tag{5.20}$$

Therefore, A is set to

$$A = \left(\frac{3-q}{5-3q}\right)^{-1}. \tag{5.21}$$

Here, we assume $q < \frac{5}{3}$ since a q-Gaussian distribution has an infinite variance if $q \geq \frac{5}{3}$.

5.4 q-GMM for histogram-based image representation

To represent an image by a feature vector, we create a histogram of low-level features $H(X')$ from a set of low-level features $X' = \{x_i\}_{i=1}^{N'}$ extracted from

an image as follows:

$$H(X') = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_K \end{pmatrix}, \; C_k = \sum_i c_{ik}, \tag{5.22}$$

where c_{ik} is the posterior probability of x_i being at the k-th q-Gaussian component. It is given by

$$c_{ik} = \frac{\hat{w}_k \mathcal{N}_q(x_i \mid \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{k=1}^{K} \hat{w}_k \mathcal{N}_q(x_i \mid \hat{\mu}_k, \hat{\Sigma}_k)}. \tag{5.23}$$

where $\hat{w}_k, \hat{\mu}_k$, and $\hat{\Sigma}_k$ are q-GMM parameters for the background model. The posterior probabilities c_{ik} can be viewed as weights in soft-assignment of visual words since they satisfy

$$\sum_{k=1}^{K} c_{ik} = 1, \; 0 \leq c_{ik} \leq 1. \tag{5.24}$$

Thus, the q-GMM can be regarded as an extension of BoW to a probabilistic framework.

We found that the assignment using the q-GMM comes close to the hard-assignment (i.e., only one of c_{ik} is equal to 1.0 and others are 0.0) as q decreases, and comes close to the uniform-assignment (i.e., all c_{ik} have equivalent values) as q increases. To measure how much the assignment is close to the hard-assignment, we introduce the assignment *hardness* h defined by

$$h = \frac{1}{N'} \sum_{i=1}^{N'} \frac{\max_k c_{ik} - K^{-1}}{1 - K^{-1}}. \tag{5.25}$$

The assignment hardness h is designed to reach 1.0 for the hard-assignment and 0.0 for the uniform-assignment.

Fig. 5.4 shows the assignment hardness with different q-values. To improve the final performance of image and video indexing, the assignment should not be too hard nor too uniform. Here, we employ a q-value of 1.05 that has the middle value (0.5) of assignment hardness.

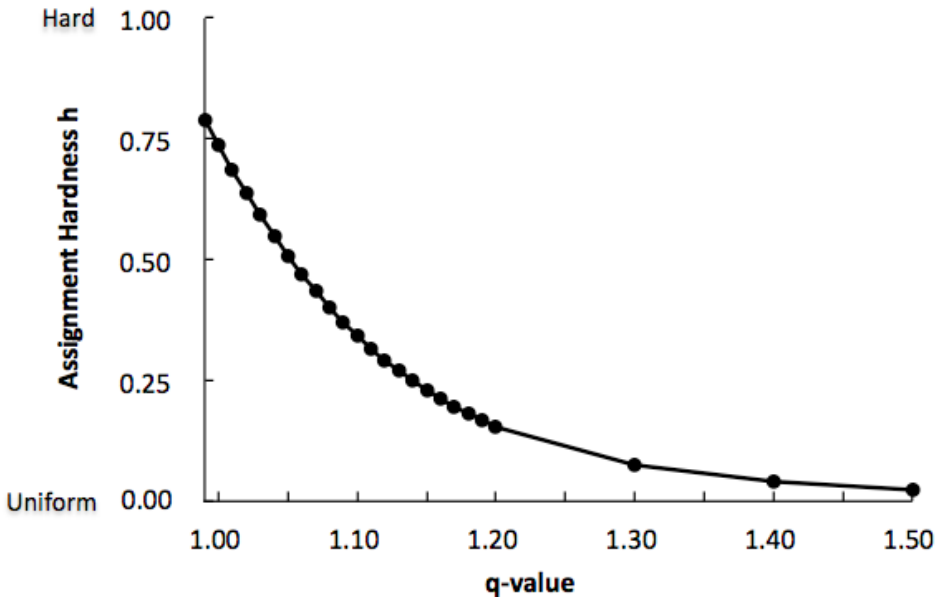

Figure 5.4: Assignment hardness h with different q-values.

5.5 q-GMM Kernel

Here, we introduce q-GMMs instead of the BoW histograms to represent images and videos. Generally, the number of low-level features extracted from an image is limited and may not be enough to estimate q-GMM parameters robustly. Thus, we use the maximum a posteriori criteria which provides robust parameter estimation. For each image that has low-level features $X' = \{x_i\}_{i=1}^{N'}$, we only update q-GMM mean vectors from the background model as follows:

$$\tilde{\mu}'_k = \frac{\tau\hat{\mu}_k + \sum_{i=1}^{N'} c_{ik}x_i}{\tau + \sum_{i=1}^{N'} c_{ik}}, \tag{5.26}$$

where N' is the number of the low-level features, $\hat{\mu}_k$ is a q-GMM parameter for the background model, τ is a prefixed hyper-parameter, and c_{ik} is the posterior probability given by Eq. (6.10).

For a kernel to train support vector machines (SVMs), we introduce the

following RBF-based kernel, namely q-GMM kernel,

$$k(X', X'') = \exp\left(-\gamma \sum_{k=1}^{K} \hat{w}_k (\tilde{\mu}'_k - \tilde{\mu}''_k)^T \hat{\Sigma}_k^{-1} (\tilde{\mu}'_k - \tilde{\mu}''_k)\right), \tag{5.27}$$

where X' is a set of low-level features extracted from an image, $\tilde{\mu}'_k$ is an updated q-GMM mean vector, $\hat{\Sigma}_k, w_k$ are q-GMM parameters for the background model, and γ is a scaling parameter. The weighted sum of Mahalanobis distance between the k-th q-Gaussian components is utilized in the q-GMM kernel.

To efficiently compute the q-GMM kernel, we define the following super-vector $\phi(X')$ and store it in storage.

$$\phi(X') = \begin{pmatrix} \sqrt{\hat{w}_1}\hat{\Sigma}_1^{-\frac{1}{2}}\tilde{\mu}'_1 \\ \sqrt{\hat{w}_2}\hat{\Sigma}_2^{-\frac{1}{2}}\tilde{\mu}'_2 \\ \vdots \\ \sqrt{\hat{w}_K}\hat{\Sigma}_K^{-\frac{1}{2}}\tilde{\mu}'_K \end{pmatrix}, \tag{5.28}$$

The dimension of the super-vector is Kd where K is the number of mixture components and d is the dimension of low-level descriptors. The super-vector immediately implies the following simplification of the q-GMM kernel.

$$k(X', X'') = \exp\left(-\gamma \|\phi(X') - \phi(X'')\|_2^2\right). \tag{5.29}$$

5.6 Experiments

5.6.1 Experimental Conditions

In this section, the proposed method is evaluated on two data sets: PASCAL VOC 2010 and TRECVID 2010. The PASCAL Visual Object Classes Challenge (VOC) [71] provides a benchmark for comparison of object classification methods. The PASCAL VOC 2010 classification (validation) challenge data set consists of 4,998 training images and 5,105 testing images of 20 object classes in Table 5.1. The evaluation measure is Mean average precision (Mean AP), which is the arithmetic mean of APs over all targeted object classes. The TREC Video Retrieval Evaluation (TRECVID) [64] provides a

Table 5.1: The targeted semantic concepts for PASCAL VOC 2010 and TRECVID 2010.

PASCAL VOC 2010		
Aeroplane	Bicycle	Bird
Boat	Bottle	Bus
Car	Cat	Chair
Cow	Diningtable	Dog
Horse	Motorbike	Person
Pottedplant	Sheep	Sofa
Train	Tvmonitor	

TRECVID 2010		
Airplane Flying	Animal	Asian People
Bicycling	Boat Ship	Bus
Car Racing	Cheering	Cityscape
Classroom	Dancing	Dark-skinned People
Demonstration Or Protest	Doorway	Explosion Fire
Female Human Face Closeup	Flowers	Ground Vehicles
Hand	Mountain	Nighttime
Old People	Running	Singing
Sitting down	Swimming	Telephones
Throwing	Vehicle	Walking

benchmark for comparison of video indexing methods. The TRECVID 2010 Semantic Indexing data set consists of 119,685 training video shots and 146,788 testing video shots. 30 semantic concepts of objects, actions, and scenes in Table 5.1 and their ground truth labels are provided. Officially provided key-frame images for each video shot is used in our experiments. The evaluation measure is Mean AP among the 30 semantic concepts.

To test our q-GMM on these benchmarks, the low-level image descriptors are densely sampled from 100x100 grid with 3 different scales on an image. The descriptor is a concatenation of 128-dimension SIFT descriptor [5] and 36-dimension hue-histogram descriptor [28]. The dimension of each descriptor is reduced to 32 by applying Principal Component Analysis (PCA) after the concatenation.

The q-GMM for a background model is constructed by applying the proposed EM algorithm to one million randomly sampled descriptors. The number of mixture components K and the hyper-parameter τ in Eq. (5.26) are set to 512 and 20.0, respectively, in all experiments except the experiment evaluating their influence. Note that the dimension of the q-GMM histogram

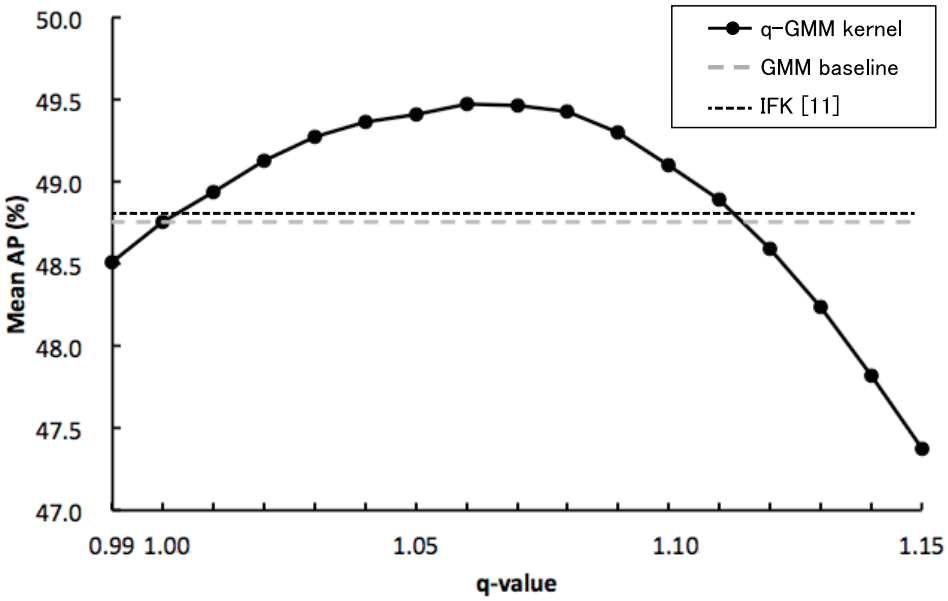

Figure 5.5: The performance comparison of q-GMM kernels with different q-values on the PASCAL VOC 2010 dataset. The q-GMM kernel outperforms the GMM baseline ($q = 1.00$) and the improved Fisher kernel [37] of GMM means.

representation and q-GMM super-vector is 512 and 16,384 (=512×32), respectively.

5.6.2 Experimental Results

PASCAL VOC 2010

We first compared our q-GMM histogram representation in Sec.5.4 with the hard-assignment BoW [3]. Mean AP of the BoW was 30.93% and the q-GMM histogram improved it to 32.03%. The q value is set to 1.05 which has the hardness of 0.5 as presented in Sec. 5.4. This shows that the q-GMM is more suitable than the hard-assignment BoW for representing an image by a histogram of low-level descriptors.

Next, we compared our q-GMM kernel in Sec.5.5 with three kernel methods: χ^2-kernel, Fisher kernel (FK) [36], and Improved Fisher kernel (IFK) [37]. For the χ^2-kernel, we computed χ^2-distance between the q-GMM histogram representations. For FK and IFK, we extracted Fisher vectors for GMM means, whose dimension is the same as our super-vector. We ap-

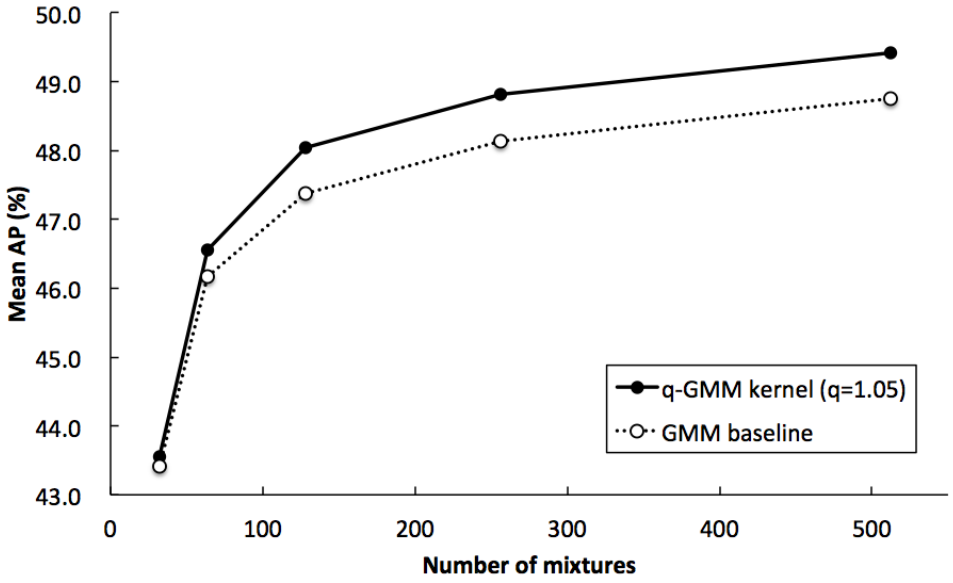

Figure 5.6: Mean AP on PASCAL VOC 2010 for different numbers of mixture components for q-GMM kernel.

plied L2 and power normalization as in [37] for IFK. The parameter of the power normalization was set to 0.4 which performed the best in our experiments. As shown in Table 5.2, the q-GMM kernel performed the best among these methods and achieved 49.42% in Mean AP. Figure 5.5 shows the performance of the q-GMM kernel for different q values. As can be seen, the q-GMM outperformed the normal GMM baseline ($q = 1.00$). The best q value and its Mean AP were 1.06 and 49.47%, respectively. This shows the effectiveness of the q-GMM kernel in image classification.

As described in [8], another idea to obtain a discriminative image representation is to train a class-specific model instead of the background model for a visual codebook. However, APs for aeroplane and bicycle were decreased by 1.30% and 2.09%, respectively, when a q-GMM is trained on one million descriptors sampled only from images of the targeted object. There was no significant performance improvement by concatenating both representations: AP for aeroplane was improved by 0.46% but that for bicycle was decreased by 0.31%. In conclusion, it is better to train a visual codebook on images of various object categories. A class-specific model could be useful for other problems such as dog breed classification that focus on a specific category.

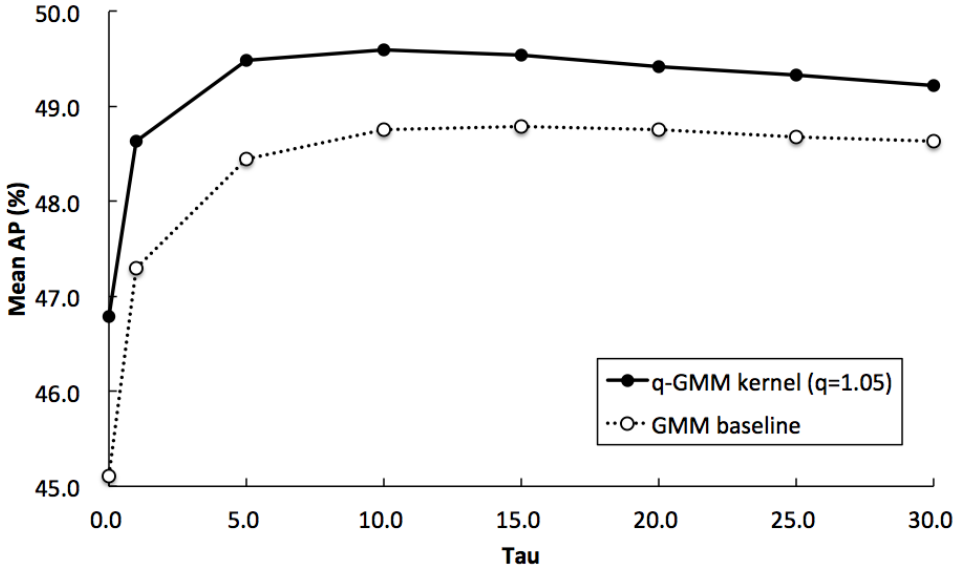

Figure 5.7: Mean AP on PASCAL VOC 2010 for different hyper-parameter τ in maximum a posteriori adaptation for q-GMM kernel.

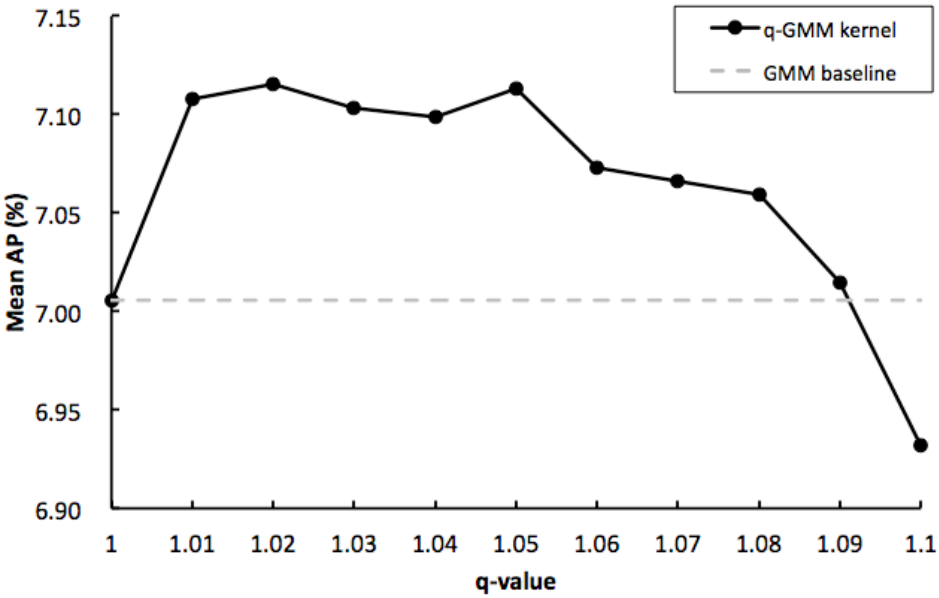

Figure 5.8: The performance comparison of q-GMM kernels with different q-values on the TRECVID 2010 dataset.

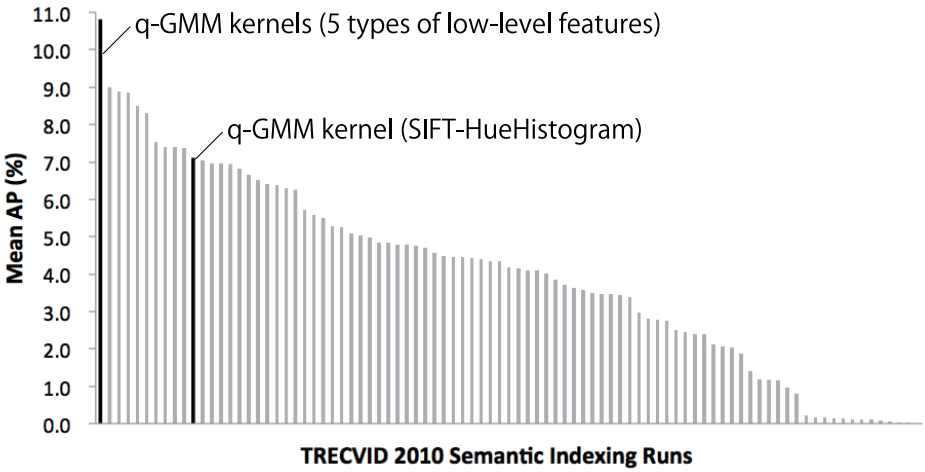

Figure 5.9: The performance comparison with other methods in TRECVID 2010. We achieved 0.071 in Mean AP by using a q-GMM kernel with SIFT-HueHistogram features and achieved 0.109 with additional 4 types of low-level features.

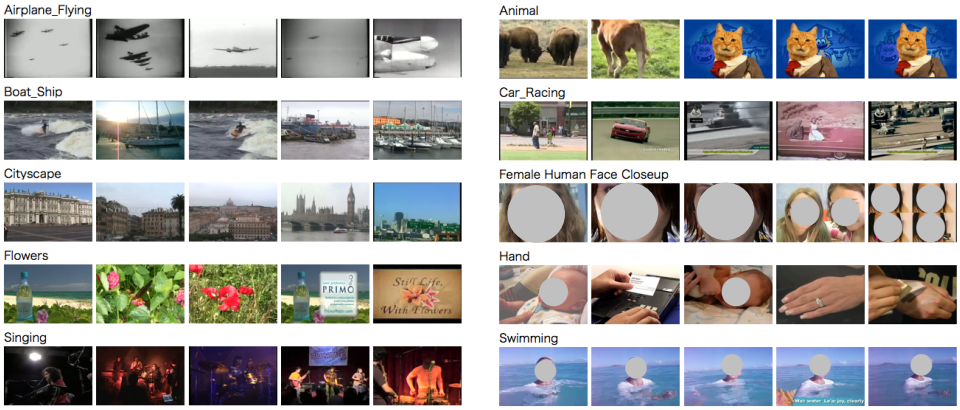

Figure 5.10: Examples of detected video shots in TRECVID 2010 dataset. Top 5 video shots are shown for ten semantic concepts.

Table 5.2: Performance comparison on PASCAL VOC 2010 dataset. BoW: bag-of-visual-words histogram representation [3] obtained by using vector quantization. Our histogram: q-GMM based histogram representation in Sec. 5.4 for $q = 1.00$ (GMM) and $q = 1.05$. χ^2 kernel: χ^2 kernel on q-GMM histogram representation. FK: Fisher kernel [36] of a GMM. IFK: improved Fisher kernel [37]. Our kernel: q-GMM kernel in Sec.5.5 for $q = 1.00$ (GMM) and $q = 1.05$.

Concept	BoW[3]	Our histogram		χ^2	FK [36]	IFK[37]	Our kernel	
		GMM	q-GMM				GMM	q-GMM
aeroplane	54.38	55.21	55.99	75.10	68.41	81.21	81.72	**82.34**
bicycle	36.53	37.88	38.20	42.52	46.79	52.07	52.62	**53.50**
bird	24.76	25.79	25.77	36.57	34.26	43.82	44.39	**45.07**
boat	37.41	38.29	39.65	50.51	50.37	**58.36**	56.51	57.32
bottle	9.71	10.03	10.28	16.36	18.06	20.51	21.91	**22.62**
bus	56.62	58.56	59.99	66.90	71.80	77.17	76.29	**77.50**
car	35.63	36.51	36.34	45.76	51.08	55.82	56.87	**57.24**
cat	41.75	42.33	42.05	49.62	52.09	56.96	57.18	**57.16**
chair	33.96	34.79	35.04	41.99	43.39	47.10	46.70	**47.76**
cow	7.12	7.69	7.66	12.24	13.48	20.35	21.77	**22.76**
diningtable	14.89	15.36	15.22	24.11	30.76	34.11	34.30	**34.65**
dog	24.36	24.84	24.63	35.30	39.41	44.30	44.61	**44.67**
horse	21.53	22.98	24.06	31.77	35.90	43.71	43.45	**43.81**
motorbike	27.30	28.55	28.19	43.41	49.64	56.27	57.17	**58.83**
person	68.63	69.23	69.29	73.42	73.16	76.69	76.84	**77.50**
pottedplant	8.54	8.38	8.04	10.85	14.31	15.78	16.82	**16.91**
sheep	19.68	20.42	19.37	30.07	32.64	**42.17**	41.63	42.09
sofa	16.17	16.66	17.10	24.09	26.97	33.38	33.47	**33.83**
train	44.59	45.71	46.17	51.33	57.97	62.93	63.17	**64.27**
tvmonitor	35.04	36.20	37.48	45.88	41.96	**48.79**	47.69	48.51
Mean AP	30.93	31.77	32.03	40.39	42.62	48.58	48.76	**49.42**

Analysis

Here, we analyze the influence of the number of mixture components and the hyper-parameter τ in Eq.(5.26) on the performance. Figure 5.6 compares the q-GMM kernel with the GMM baseline for different numbers of mixture components. It is shown that the q-GMM constantly performed better than the GMM baseline. We observed that the difference between the GMM and the q-GMM is large when the number of mixture components is large. This observation can be explained as follows. For a GMM, as the number of mixture components increases, fewer low-level descriptors are assigned with a significant posterior probability c_{ik} to each Gaussian, i.e., a matrix of c_{ik} becomes sparser and it decreases the performance. On the other hand, for a q-GMM, long-tailed q-Gaussian distributions prevent it from becoming sparse. This is the reason why we observed the large differ-

Table 5.3: Testing cost and Mean AP for each method. K is the number of mixture components, D is the dimension of low-level descriptor, and N is the averaged number of support vectors of an SVM.

Representation	Dimension	Kernel type	N	Testing cost	Mean AP
q-GMM histogram	512	linear	1540	$O(K)$	32.03
		χ^2	1700	$O(NK)$	40.39
q-GMM supervector	16384	linear	1251	$O(DK)$	47.09
		RBF	2261	$O(NDK)$	**49.42**
Fisher vector (mean)	16384	linear	1494	$O(DK)$	46.21
		RBF	2556	$O(NDK)$	48.58

ence for a large number of mixture components.

Figure 5.7 illustrates the influence of the hyper-parameter τ in Eq.(5.26) on the performance. We observed no significant change in performance when τ was between 10.0 to 25.0. Thus, we conclude that values between 10.0 to 25.0 are reasonable. We also confirmed that the improvement by the q-GMM is robust against the hyper-parameter τ.

TRECVID 2010

Table 5.4 shows the performance comparison of the q-GMM kernel and the GMM baseline on TRECVID 2010. The q-GMM kernel of $q = 1.05$ outperformed the GMM and achieved 7.11% in Mean AP. As shown in Figure 5.8, the q-value of 1.02 performed the best with 7.12% Mean AP on this dataset.

On the other hand, we observed that the performance is improved to 7.49% if we choose the best q-value for each of targeted semantic concept. This shows that supervised q-value optimization has potential for improving the overall performance in future work while our q-value optimization in Sec 5.4, which is based on the assignment hardness, was in an unsupervised way.

Another idea to improve the overall performance is to use the both of the GMM and the q-GMM. The simplest implementation of this idea is averaging two detection scores obtained from the GMM and the q-GMM. We additionally evaluated it and observed 7.30% Mean AP as shown in Table 5.4. On the other hand, a disadvantage of averaging scores is that it's time-consuming to compute independent scores for the GMM and the q-GMM. To reduce the computational costs, an efficient method for calculating q-Gaussian probabilities for multiple q-values is needed in future work.

Table 5.4: Average precision (AP) by semantic concepts on TRECVID 2010. Results for GMM, q-GMM ($q = 1.05$), score fusion of GMM and q-GMM ($q = 1.05$), and feature fusion of 5 types of visual and audio features for q-GMM are reported.

	Single feature			Feature fusion
Concept	GMM	q-GMM	Fusion	q-GMM
Airplane Flying	2.75	2.67	**3.01**	15.64
Animal	2.18	**2.53**	2.39	6.44
Asian People	**0.45**	0.18	0.40	3.08
Bicycling	3.10	2.90	**3.31**	5.90
Boat Ship	5.28	**5.51**	5.24	11.01
Bus	**0.80**	0.43	0.57	1.42
Car Racing	**4.30**	3.92	4.03	4.37
Cheering	3.22	**3.56**	3.38	3.81
Cityscape	9.91	10.75	**10.85**	17.43
Classroom	1.24	**1.27**	1.21	0.81
Dancing	3.12	**5.10**	4.53	8.89
Dark-skinned People	12.85	13.21	**13.77**	20.40
Demonstration Or Protest	13.63	13.54	**14.14**	17.86
Doorway	7.84	7.14	**7.95**	12.44
Explosion Fire	**4.61**	4.03	4.28	3.93
Female-Human-Face-Closeup	10.94	10.55	**11.09**	17.79
Flowers	3.57	**3.95**	3.59	3.86
Ground Vehicles	14.59	14.15	**15.46**	20.20
Hand	**4.06**	4.05	3.92	9.30
Mountain	20.14	**20.83**	20.15	20.87
Nighttime	**12.89**	9.99	12.24	15.88
Old People	**2.58**	1.98	2.37	8.20
Running	1.42	1.87	**1.90**	6.88
Singing	6.80	8.11	**8.40**	17.47
Sitting Down	**0.12**	0.08	0.09	0.56
Swimming	32.98	33.07	**33.35**	30.82
Telephones	1.03	**1.39**	**1.39**	1.88
Throwing	3.49	**5.61**	5.45	7.02
Vehicle	**14.61**	14.04	14.28	18.91
Walking	5.66	**6.98**	6.41	13.03
Mean	7.01	7.11	**7.30**	10.87

Comparison with other methods

Fig. 5.9 shows the performance comparison with the other methods in the TRECVID 2010 Semantic Indexing Task [64]. Mean AP of 7.11%, which

was obtained by using our q-GMM kernel ($q = 1.05$), ranked 10-th among 87 official runs. Fig. 5.10 shows some examples of detected video shots. We conclude the q-GMM kernel performed well since the other methods typically used more than 5 types of low-level features while we used only one type of low-level features (SIFT with hue histogram).

Furthermore, we achieved Mean AP of 10.90%, which is better than the best performance on the TRECVID 2010, by combining q-GMM kernels for 4 additional types of low-level features: SIFT with Harris-affine detector, SIFT with Hessian-affine detector, dense HOG, and MFCC audio features.

5.7 Conclusion

We proposed q-Gaussian mixture models (q-GMMs) and their application to image and video semantic indexing systems. It has been shown in our experiments that the q-GMM kernels outperform both of the BoW method and the normal GMM. The q-GMM kernel achieved 0.494 and 0.109 in Mean Average Precision on the PASCAL VOC 2010 dataset and the TRECVID 2010 Semantic Indexing dataset, respectively. The linear kernel on q-GMM supervectors was shown to be effective in terms of the scalability. Our future work will focus on optimization of q-values for each semantic concepts. An extension of the Fisher information analysis to Tsallis statistics would be interesting as a promising next step.

Chapter 6

Neighbor-To-Neighbor Search

6.1 Overview

Assigning a visual code to a low-level image descriptor, which we call *code assignment*, is the most computationally expensive part of image classification algorithms based on the bag of visual word (BoW) framework. As described in Chapter 3, a tree-structured GMM reduces computational cost by searching visual words for each input feature vectors independently. However, input vectors are often strongly depend on each other when they are extracted from the same region in an image. Their typical examples are densely-sampled image descriptors such as dense SIFT, which have been proven to be effective in image classification [23, 37].

In this chapter, we proposes a fast computation method, Neighbor-to-Neighbor (NTN) search, for this code assignment. Based on the fact that image features from an adjacent region are usually similar to each other, this algorithm effectively reduces the cost of calculating the distance between a codeword and a feature vector. This method can be applied not only to a *hard* codebook constructed by vector quantization (NTN-VQ), but also to a *soft* codebook, a Gaussian mixture model (NTN-GMM).

Figure 6.1: **Neighbor-to-neighbor (NTN) search.** NTN search assigns a code to an input vector from a neighbor vector to a neighbor vector. A typical example of a neighbor vector is a descriptor x_j adjacent to a descriptor x_{j-1} where image descriptors are densely sampled from an image. The red path on the image shows the ordering of descriptors.

6.2 Neighbor-To-Neighbor (NTN) Search for Vector Quantization

6.2.1 Outline

This section presents our neighbor-to-neighbor (NTN) search in a simple framework, hard VQ. Let X be a set of input vectors and $B(x)$ be a set of neighbor vectors for an input vector $x \in X$. The NTN search assumes that a neighbor vector in $B(x)$ is similar to x, and that the number of neighbor vectors is smaller than the codebook size. A typical example that satisfies this assumption is densely-sampled SIFT descriptors for image classification . Here, $B(x)$ is a set of the four descriptors adjacent to a descriptor x (Figure 6.1 and Figure 6.2) or a set of descriptors in the same pre-segmented region.

In NTN search, input vectors are ordered from a neighbor vector to a neighbor vector to skip distance calculations for some input vectors based on a triangle inequality. We first explain the structure of our algorithm and then explain our speeding-up idea.

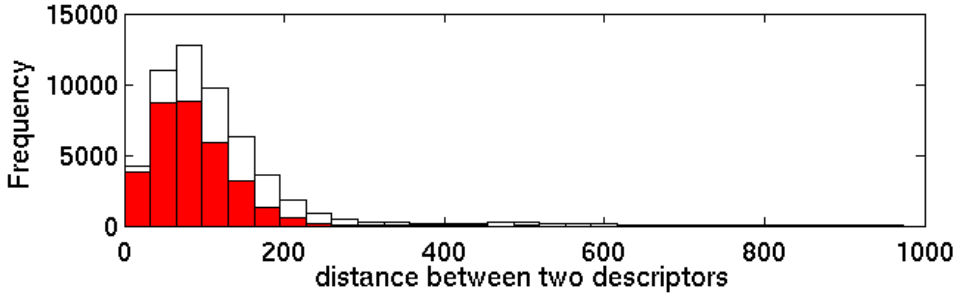

Figure 6.2: **A histogram of descriptors.** Red bars: descriptors that have the same visual word as a neighbor descriptor. White bars: all descriptors. SIFT descriptors are extracted from every 4 pixels at 5 scales on the PASCAL VOC 2007 training images. The codebook size is 512. 61.3% of two adjacent descriptors have the same visual word.

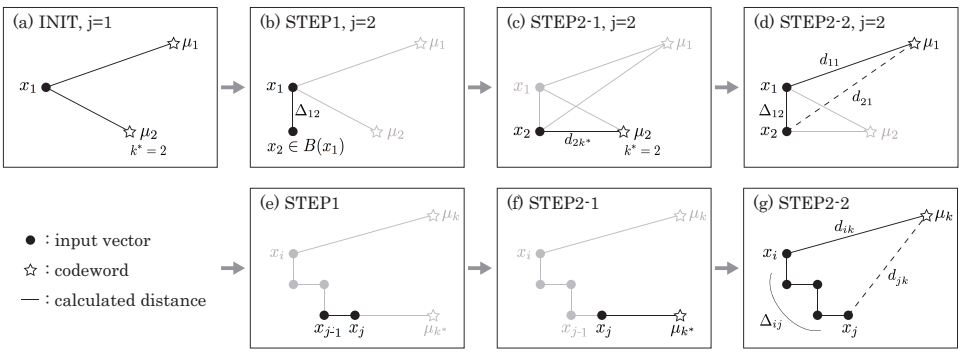

Figure 6.3: **Algorithm overview.**

6.2.2 Algorithm

Let $\{\mu_k\}_{k=1}^K$ be a codebook. In the initialization step for $j = 1$, x_j is randomly selected from X. Its code v_j is determined as

$$v_j = \underset{k}{\operatorname{argmin}}\, d_{jk}, \tag{6.1}$$

where distance

$$d_{jk} = \|x_j - \mu_k\|, \tag{6.2}$$

is calculated for each $k = 1, 2, \cdots, K$. This process is the same as the straightforward hard VQ.

For $j = 2, 3, \cdots, N$, the following three steps are iterated (Figure 6.3).

(STEP 1: Select the next input vector)

For each $x \in B(x_{j-1})$, calculate $\Delta(x) = \|x - x_{j-1}\|$, and set

$$x_j = \underset{x \in B(x_{j-1}) \cap \bar{X}}{\operatorname{argmin}} \Delta(x), \tag{6.3}$$

where $\bar{X} = X \setminus \{x_1, \cdots, x_{j-1}\}$ is a set of remaining input vectors. If $B(x_{j-1}) \cap \bar{X} = \emptyset$ then x_j is randomly picked from $\bar{X}$.

(STEP 2: Calculate distance)

Set $k^* = v_{j-1}$.

2-1) Calculate distance d_{jk^*}.

2-2) For $k = 1, 2, \cdots, k^* - 1, k^* + 1, \cdots, K$, calculate a lower bound $\underline{d}_{jk}$ for d_{jk} as follows.

$$\underline{d}_{jk} = d_{ik} - \delta \Delta_{ij}, \tag{6.4}$$

where i is the index of the input vector whose distance d_{ik} has been calculated, δ is a parameter, and Δ_{ij} is an accumulated distance from x_i to x_j. This process will be explained in detail in the next paragraph. If $\underline{d}_{jk} \geq d_{jk^*}$ then skip calculation of d_{jk}, otherwise calculate d_{jk}.

(STEP 3: Output a code)

Calculate

$$v_j = \underset{k \in E}{\operatorname{argmin}}\, d_{jk}, \tag{6.5}$$

where E is a set of indices of codewords whose distance to x_j is calculated in STEP 2.

Here we explain Eq. (6.4) in STEP 2. For a given x_j, let's go back to the previous input vector x_i $(i < j)$ whose distance d_{ik} has been calculated (Figure 6.3 (g)). Take the maximum such index i and let Δ_{ij} be an accumulated distance between x_i and x_j given by

$$\Delta_{ij} = \sum_{p=i+1}^{j} \|x_p - x_{p-1}\|. \tag{6.6}$$

The triangle inequality gives

$$d_{ik} - \Delta_{ij} \leq d_{jk} \leq d_{ik} + \Delta_{ij}. \tag{6.7}$$

It implies

$$\exists \delta^* \in [-1, 1] \text{ s.t. } d_{jk} = d_{ik} - \delta^* \Delta_{ij}. \tag{6.8}$$

Thus, for $\delta \geq \delta^*$, $\underline{d}_{jk}$ in Eq. (6.4) is a lower bound of distance d_{jk}. Note that the result of coding by this algorithm is exactly the same as that by the original hard VQ in this case.

6.2.3 The parameter δ

Our idea to improve the speed of the algorithm is to regard δ as a constant and use it as a parameter. Then, the lower bound is efficiently updated from the previous lower bound by

$$\underline{d}_{jk} = \underline{d}_{j-1,k} - \delta \|x_j - x_{j-1}\|. \tag{6.9}$$

The lower bound is obtained by only one distance calculation from x_{j-1} to x_j, which is already calculated in STEP 1. By relaxing the restriction $\delta \geq \delta^*$, we can further reduce the computational cost though the exact solution may not be obtained in such cases.

Alg. 1 summarizes the neighbor-to-neighbor (NTN) search for hard VQ which outputs assigned codes for each input vector quickly.

Algorithm 1 NTN-VQ

Input: input vectors X $(N = |X|)$,
codebook $\{\mu_k\}_{k=1}^K$, parameter δ.
Output: codes $\{v_i\}_{i=1}^N$
$x_1 \leftarrow \text{Rand}(X)$
$\underline{d}_k \leftarrow \|x_1 - \mu_k\|$ **for all** k
$v_1 \leftarrow \underset{k}{\text{argmin}}\, \underline{d}_k$; $k^* \leftarrow v_1$
for $i = 2, \cdots, N$ **do**
  $x_i \leftarrow \underset{x \in B(x_{i-1}) \cap \bar{X}}{\text{argmin}} \|x - x_{i-1}\|$
  $\underline{d}_{k^*} \leftarrow \|x_i - \mu_{k^*}\|$
  for all $k \neq k^*$ **do**
    $\underline{d}_k \leftarrow \underline{d}_k - \delta\|x_i - x_{i-1}\|$
    if $\underline{d}_{k^*} > \underline{d}_k$ **then**
      $\underline{d}_k \leftarrow \|x_i - \mu_k\|$
      if $\underline{d}_{k^*} > \underline{d}_k$ **then** $k^* \leftarrow k$ **end if**
    end if
  end for
  $v_i \leftarrow k^*$
end for

6.3 NTN Search for Gaussian Mixture Models

A Gaussian mixture model (GMM) is an extension of hard VQ to a probabilistic framework since it provides a soft assignment of codewords to an input vector. Here we extend the NTN search to a GMM framework (NTN-GMM). The algorithm structure of NTN-GMM is the same as NTN-VQ, but instead of a lower bound of distance for NTN-VQ, an upper bound of a Gaussian probability is calculated for NTN-GMM.

Let μ_k, Σ_k and w_k $(k = 1, 2, \cdots K)$ be the mean vector, the covariance matrix, and the mixture weight of the k-th mixture component (codeword) of a GMM, respectively. A code c_{jk} for an input vector $x_j (j = 1, 2, \cdots, N)$ to the k-th codeword is given by

$$c_{jk} = \frac{p_{jk}}{\sum_{k'=1}^{K} p_{jk'}}. \tag{6.10}$$

Here p_{jk} is a Gaussian probability given by

$$p_{jk} = \frac{w_k}{(2\pi)^{\frac{2}{d}}|\Sigma_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\|x_j - \mu_k\|^2_{\Sigma_k^{-1}}\right), \tag{6.11}$$

where $\|x\|_A = \sqrt{x^T A x}$. Note that NK probability calculations are required in the standard GMM.

Our empirical observation shows that the distribution of p_{jk} over all the codewords is peaky, i.e., for each input vector x_j, a few p_{jk}'s have a large value and the others do not. If x_j is similar to x_{j-1}, the "peak" is shifting gradually as j increments. Conversely, the changes in the "bottom (no-peak)" of the distribution are generally small. This observation brings us to an idea that we may ignore the change of Gaussian probabilities in the "bottom" of the distribution.

For a given x_j, let x_i $(i < j)$ be the previous input vector whose Gaussian probability p_{ik} has been calculated. The idea is to ignore the difference between p_{jk} and p_{ik} and assume $p_{jk} = p_{ik}$ for $k \in G_i^{(b)} \cap G_j^{(b)}$ to skip calculation of p_{jk}. Here, $G_j^{(b)}$ is a set of mixture components in the "bottom" of the distribution, which we call a bottom set, given by

$$G_j^{(b)} = \{k : p_{jk} < p_{\text{th}}\}. \tag{6.12}$$

where p_{th} is a threshold to categorize the mixture components into "peak" and "bottom".

A bottom set $G_j^{(b)}$ cannot be directly observed without computing p_{jk}. Thus, we introduce an upper bound $\overline{p}_{jk}$ of a probability p_{jk} given by

$$\overline{p}_{jk} = p_{ik} \exp\left(\delta_{ik}\Delta_{ij}\right). \tag{6.13}$$

Here, Δ_{ij} is the accumulated distance given by Eq. (6.6) and δ_{ik} is given by

$$\delta_{ik} = S_k \delta \|x_i - \mu_k\|_{\Sigma_k^{-1}}, \tag{6.14}$$

where $\delta \in [0, 1]$ is a parameter to control the speed of our algorithm (as Subsec. 6.2.3) and S_k is the square root of the spectral radius of Σ_k^{-1} .

The upper bound $\overline{p}_{jk}$ of the probability (Eq. (6.13)) is delivered as fol-

lows. The law of cosines gives

$$\begin{aligned}&\exists \delta^* \in [-1,1] \text{ s.t. } \|x_j-\mu_k\|^2_{\Sigma_k^{-1}} \\ &= \|x_i-\mu_k\|^2_{\Sigma_k^{-1}} + \|x_i-x_j\|^2_{\Sigma_k^{-1}} - 2\delta^*\|x_i-x_j\|_{\Sigma_k^{-1}}\|x_i-\mu_k\|_{\Sigma_k^{-1}}.\end{aligned} \tag{6.15}$$

For $\delta \geq \max(\delta^*, 0)$, it implies

$$\begin{aligned}\|x_j-\mu_k\|^2_{\Sigma_k^{-1}} &\geq \|x_i-\mu_k\|^2_{\Sigma_k^{-1}} - 2S_k\delta\|x_i-x_j\|\|x_i-\mu_k\|_{\Sigma_k^{-1}} \\ &\geq \|x_i-\mu_k\|^2_{\Sigma_k^{-1}} - 2S_k\delta\Delta_{ij}\|x_i-\mu_k\|_{\Sigma_k^{-1}},\end{aligned} \tag{6.16}$$

where S_k is the square root of the spectral radius of Σ_k^{-1} and Δ_{ij} is the accumulated distance given by Eq. (6.6). Thus, we have

$$p_{jk} = \frac{w_k}{Z_k}\exp\left(-\frac{1}{2}\|x_j-\mu_k\|^2_{\Sigma_k^{-1}}\right) \tag{6.17}$$

$$\leq \frac{w_k}{Z_k}\exp\left(-\frac{1}{2}\|x_i-\mu_k\|^2_{\Sigma_k^{-1}} + S_k\delta\Delta_{ij}\|x_i-\mu_k\|_{\Sigma_k^{-1}}\right) \tag{6.18}$$

$$= p_{ik}\exp\left(S_k\delta\Delta_{ij}\|x_i-\mu_k\|_{\Sigma_k^{-1}}\right) \tag{6.19}$$

$$= p_{ik}\exp\left(\delta_{ik}\Delta_{ij}\right) = \overline{p}_{jk}, \tag{6.20}$$

where $Z_k = (2\pi)^{\frac{d}{2}}|\Sigma_k|^{\frac{1}{2}}$ and δ_{ik} is given in Eq.(6.14).

Note that this upper bound is obtained efficiently from a previous upper bound by

$$\overline{p}_{jk} = \overline{p}_{j-1,k}\exp\left(\delta_{ik}\|x_j-x_{j-1}\|\right). \tag{6.21}$$

Finally, instead of the intersection of bottom sets $G_i^{(b)} \cap G_j^{(b)}$, its subset U_{ij} given by

$$U_{ij} = \{k : \overline{p}_{jk} < p_{\text{th}}\}, \tag{6.22}$$

is used for determining mixture components to skip calculation of p_{jk} (Figure 6.4).

The threshold p_{th} should depend on the maximum value of Gaussian probabilities at j, i.e., $\max_k p_{jk}$. However, this value also cannot be observed without computing all Gaussian probabilities at j. Since two adjacent descriptors are expected to be similar to each other, the value at the previous

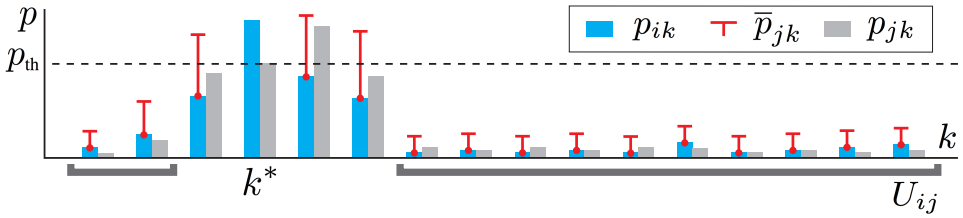

Figure 6.4: **Distribution of p_{ik} and $p_{jk}(i < j)$.** Calculation of a Gaussian probability p_{jk} is skipped for $k \in U_{ik}$.

maximum point is used to determine the threshold as

$$p_{\text{th}} = p_{jk^*}, \quad k^* = \underset{k}{\operatorname{argmax}}\, p_{j-1,k}. \tag{6.23}$$

Note that, by this thresholding, Gaussian probabilities at the previous maximum point and the current maximum point (at least) will be calculated for each input vector.

Alg. 2 summarizes the NTN search for a GMM which outputs soft codes for each input vector quickly.

To further improve the speed of NTN-GMM, avoiding the *exp* computation is effective since our observation shows that 63.0% of the computational cost in coding using a GMM is spent for it. An *exp* operator is deleted by taking a log of Gaussian probabilities and introducing log-max (LM) approximation to approximate Eq. (6.10) by

$$c_{jk} \simeq \begin{cases} 1, & \text{if } k = \underset{k}{\operatorname{argmax}} \log p_{jk}, \\ 0, & \text{otherwise.} \end{cases} \tag{6.24}$$

6.4 Experimental evaluation

6.4.1 Experimental setup

We perform image classification experiments on the PASCAL VOC 2007 classification challenge [72]. The dataset consists of 9,963 images, which are divided into a training set (5011 images) and a testing set (4952 images). We use Mean Average Precision (Mean AP) over the 20 object categories for evaluating classification accuracies. A single core of a 2.93 GHz Intel Xeon

Algorithm 2 NTN-GMM

Input: input vectors X $(N = |X|)$,
GMM $\{w_k, \mu_k, \Sigma_k\}_{k=1}^{K}$, parameter δ.
Output: soft codes $\{c_{ik}\}_{i=1}^{N}{}_{k=1}^{K}$
$x_1 \leftarrow \text{Rand}(X)$
$p_k, \overline{p}_k \leftarrow w_k \mathcal{N}(x_1|\mu_k, \Sigma_k)$ **for all** k
$\delta_k \leftarrow S_k \delta \|x_1 - \mu_k\|_{\Sigma_k^{-1}}$ **for all** k
$c_{1k} \leftarrow \frac{p_k}{\sum_{k'=1}^{K} p_{k'}}$ **for all** k; $k^* \leftarrow \underset{k}{\text{argmax}}\, p_k$
for $i = 2, \cdots, N$ **do**
  $x_i \leftarrow \underset{x \in B(x_{i-1}) \cap \bar{X}}{\text{argmin}} \|x - x_{i-1}\|$
  $p_{k^*}, \overline{p}_{k^*} \leftarrow w_{k^*} \mathcal{N}(x_i|\mu_{k^*}, \Sigma_{k^*})$
  for all $k \neq k^*$ **do**
    $\overline{p}_k \leftarrow \overline{p}_k \exp(\delta_k \|x_i - x_{i-1}\|)$
    if $p_{k^*} < \overline{p}_k$ **then**
      $p_k, \overline{p}_k \leftarrow w_k \mathcal{N}(x_i|\mu_k, \Sigma_k)$
      $\delta_k \leftarrow S_k \delta \|x_i - \mu_k\|_{\Sigma_k^{-1}}$
    end if
  end for
  $c_{ik} \leftarrow \frac{p_k}{\sum_{k'=1}^{K} p_{k'}}$ **for all** k; $k^* \leftarrow \underset{k}{\text{argmax}}\, p_k$
end for

CPU with an 8 GB memory is used for measuring computational costs.

We implement NTN-VQ (Alg. 1) on super-vector (SV) coding [23] and NTN-GMM (Alg. 2) on Fisher-vector (FV) coding [37]. We compare them with two standard methods, VQ, GMM, and two tree-based methods, ANN-VQ, Tree-GMM. The ANN-VQ uses a fast library for approximate nearest neighbor (ANN) search in [15, 38] for SV coding. The Tree-GMM is an extension of the hierarchical k-means to a GMM framework for FV coding. In addition, NTN-LM-GMM applies the LM approximation to NTN-GMM.

In all experiments, 2×2 SIFT descriptors are extracted from every 4 pixels [1] at 5 scales. We set a set of neighbor vectors $B(x)$ to a set of the four SIFT descriptors adjacent to a descriptor x. We omit Gaussian weighting for SIFT descriptors. The averaged number of descriptors per image is 49580. A codebook is trained on randomly sampled 1 million descriptors by using k-means algorithm for VQ or EM algorithm for a GMM. Covariance matrices for a GMM are assumed to be diagonal. The codebook size is set to 512. A

[1] Mean APs were 0.582, 0.582 and 0.574 for density of 3, 4, and 5, respectively.

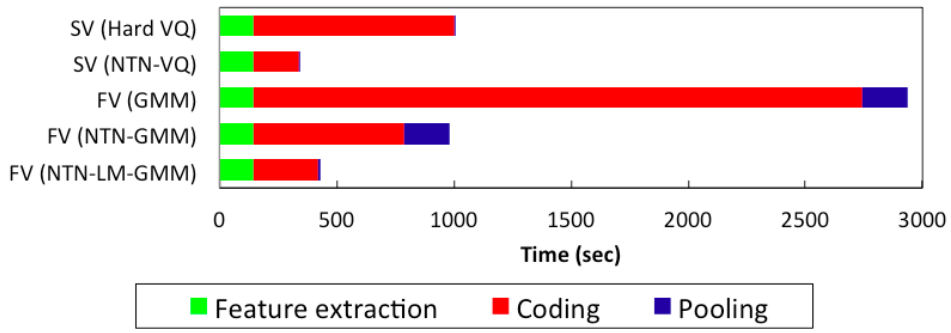

Figure 6.5: **Relative computational cost.** Computational cost for each step of super-vector (SV) coding and Fisher-vector (FV) coding is reported. The codebook size is 512. Feature extraction: SIFT descriptors are extracted from every 4 pixels at 5 scales, Coding: each descriptor is assigned to codeword(s), Pooling: an SV or FV image representation is generated. 85.3%, 56.6%, 88.4%, 65.4% and 64.2% of computational time is occupied from coding by VQ, NTN-VQ, GMM, NTN-GMM, and NTN-LM-GMM, respectively. Total computational cost is reduced by 66.0%, 66.5% and 85.3% by NTN-VQ, NTN-GMM, and NTN-LM-GMM, respectively.

one-vs-rest linear SVM is used for a classifier for each of 20 object categories, where the regularization parameter is fixed to 1.0.

6.4.2 Experimental Results

Speed of coding

In Table 6.1, we compare speed of coding at the fixed accuracy level. Overall, our NTN methods are faster than the others while keeping the classification accuracy. We observe that NTN-VQ and NTN-LM-GMM reduce the assignment cost by 77.4% and by 89.3%, respectively. Note that there are no significant differences in Mean AP on randomization test ($p < 0.05$) between methods in the same split table in Table 6.1. Here, a parameter δ is optimized on the validation set, where half of training images are used for training models and others are used for validating the models. As described in Subsec. 6.2.3, the restriction of $\delta \geq \delta^*$ is relaxed in our methods. However, more than 90% of $\underline{d}_{jk}$ gives a correct lower bound when $\delta = 0.20$ for NTN-VQ as shown in Figure 6.6.

Figure 6.7 shows the speed-accuracy trade-off for NTN methods for different values of δ. We observe that NTN-LM-GMM outperforms NTN-GMM

Method	δ	Mean AP [test/val]	$\|E\|$	Time (sec)	r (%)
VQ	-	***0.568 / 0.519**	512.0	856.2	0.0
ANN-VQ [15]	-	0.563 / 0.518	-	475.7	44.4
NTN-VQ	0.20	0.563 / 0.519	**57.8**	193.2	**77.4**
GMM	-	***0.582 / 0.532**	512.0	2595.7	0.0
Tree-GMM	-	0.582 / 0.532	295.2	1496.8	42.3
NTN-GMM	0.09	0.580 / 0.531	88.0	642.0	75.3
NTN-LM-GMM	0.09	0.579 / 0.531	**88.0**	**276.8**	**89.3**

Table 6.1: **Speed comparison at the fixed accuracy level.** VQ: standard hard vector quantization (VQ), ANN-VQ: approximate nearest neighbor search [15], NTN-VQ: our neighbor-to-neighbor (NTN) search for VQ (Alg. 1), GMM: standard Gaussian mixture model (GMM), Tree-GMM: an extension of the hierarchical k-means to a GMM framework, NTN-GMM: our NTN search for a GMM (Alg. 2), NTN-LM-GMM: NTN-GMM with log-max approximation. δ: a parameter of our NTN methods, Mean AP: image classification accuracy on the testing set and the validation set of the VOC 2007 classification challenge. $|E|$: the number of distance or probability calculations per input vector, Time: assignment time in sec, Reduction rate r: reduction rate of the assignment cost. Note that there are no statistically significant differences in Mean AP between the method marked "*" and each other method in the same split table on randomization test ($p < 0.05$).

and NTN-VQ in terms of both speed and accuracy. This is because NTN-LM-GMM has the advantages of both of NTN-VQ and NTN-GMM: it only requires distance calculations without using an *exp* operator as NTN-VQ, but it has a weight coefficient and a covariance matrix for each codeword as NTN-GMM.

Compared with tree-based methods, a disadvantage of NTN methods is that they are not very effective if neighbor vectors are not similar to each other. We confirm this in Figure 6.8: a tree-based ANN-VQ performs better than RAND-VQ which replaces a neighbor vector by a randomly sampled vector in each iteration of NTN-VQ. Notably, the average distance between two neighbor descriptors (x_{j-1} and x_j) is 132.8 and 507.7 for NTN-VQ and RAND-VQ, respectively. This confirms that the assumption that neighbor vectors are similar to each other, is necessary for NTN methods.

In addition, we confirm the necessity of the accumulated distance in Eq. (6.6) by replacing it with direct distance, i.e., $\Delta_{ij} = \|x_i - x_j\|$ in Figure 6.9. If we ignore computation time for Δ_{ij}, for example in the case where a distance matrix on input vectors is pre-computed in some way, the direct distance is better than the accumulated distance. In general, the ac-

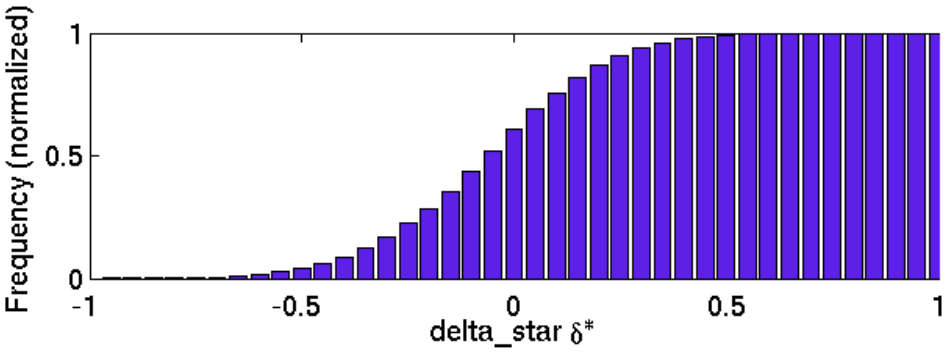

Figure 6.6: **Cumulative histogram of δ^*.** Statistics of the true δ^* in Eq. (6.8) on PASCAL VOC 2007 training images is reported for NTN-VQ.

cumulated distance is computationally effective since it derives efficient update rules of a lower/upper bound in Eq. (6.9) and (6.21).

Result examples

We examine the effectiveness of NTN-VQ for several different images in Figure 6.10. The reduction rate of the assignment cost by NTN-VQ is 84.9% for the image (a) and 66.8% for the image (d). Here, (a) and (d) are the best and the worst cases on PASCAL VOC 2007, respectively. This shows that NTN methods are more effective for images which can be segmented into several uniform regions. Notably, NTN-VQ is still better than ANN-VQ even in the worst case.

Codebook size

The simplest idea to reduce the assignment cost is to reduce the codebook size. In Figure 6.11, which shows the speed-accuracy trade-off for different codebook sizes, we confirm that using NTN methods is better than reducing the codebook size. This also confirms that NTN-LM-GMM is the best in both speed and accuracy. Note that there is no significant difference in Mean AP between a standard method and a NTN methods for each codebook size $K = 2048, 1024, \cdots, 16$.

Relative computational time in a pipeline

Figure 6.5 shows the relative cost of coding with respect to the cost of the other steps of processing pipeline for extracting SV and FV representations. As can be seen, the coding step is the majority of the whole processing

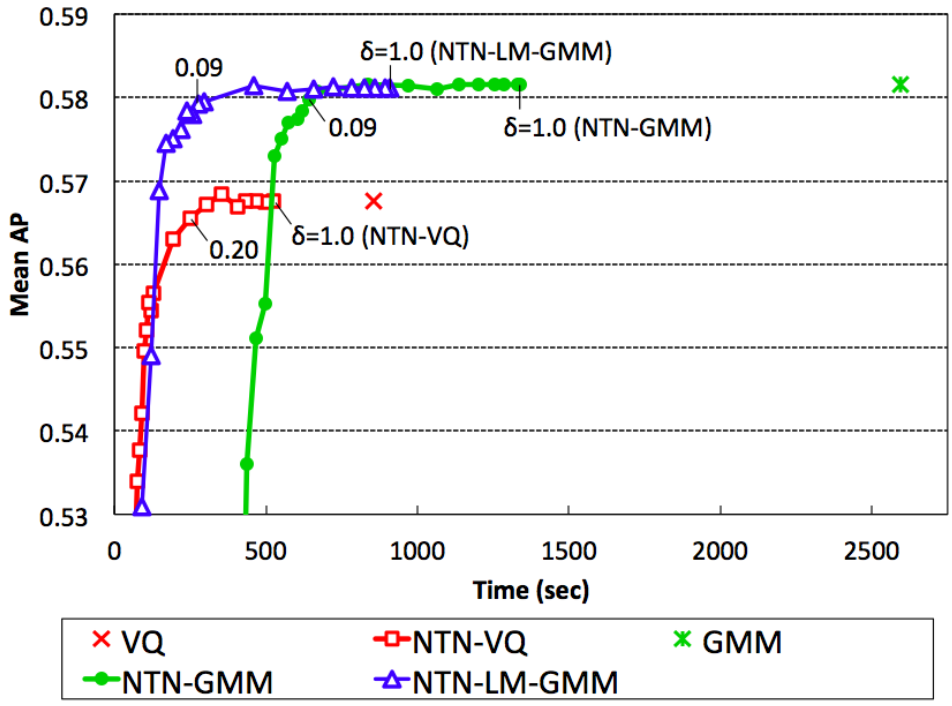

Figure 6.7: **Speed-accuracy trade-off for different values of δ.** Trade-off between assignment time and Mean AP is reported. All plots are for $\delta = 1.0, 0.9, \cdots 0.1, 0.09, \cdots, 0.01$. VQ: standard hard vector quantization (VQ), NTN-VQ: neighbor-to-neighbor (NTN) search for VQ, GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM, NTN-LM-GMM: NTN-GMM with the log-max approximation.

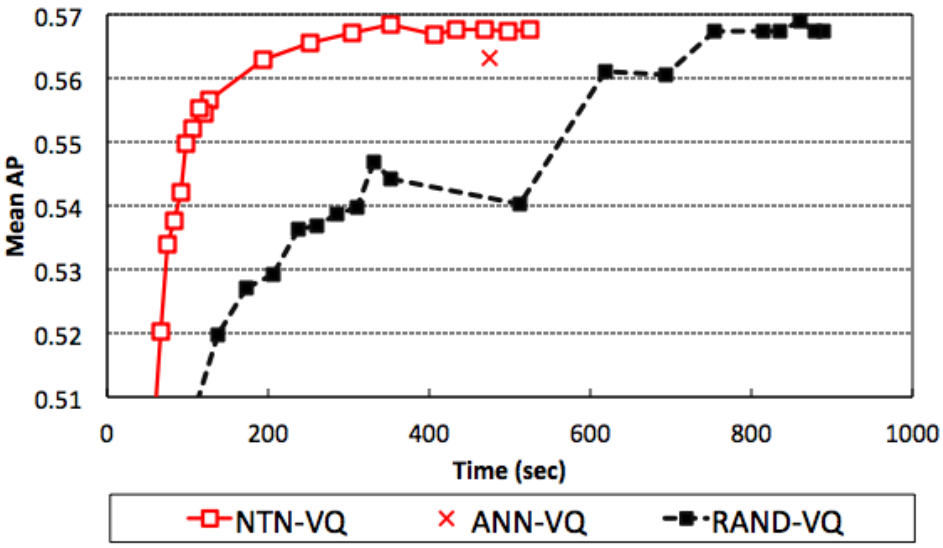

Figure 6.8: **Comparison with RAND-VQ.** Trade-off between assignment time and Mean AP is reported. NTN-VQ: neighbor-to-neighbor (NTN) search for VQ, this is the same plot as Figure 6.7, ANN-VQ: approximate nearest neighbor search [15], RAND-VQ: NTN-VQ in which a neighbor vector is replaced by a randomly sampled vector.

pipeline: 85.3% and 88.4% of computational time are occupied from it in FV coding and SV coding, respectively.

NTN-VQ, NTN-GMM and NTN-LM-GMM reduces the total computational cost by 66.0%, 66.5%, and 85.3%, respectively. Note that the cost of pooling in FV coding, which generate a final FV representation, is also reduced by the LM approximation since we can skip some summations in pooling if c_{ik} is equal to zero.

For large-scale image classification, for example on the ImageNET with more than 20,000 object categories, we should consider the SVM-classification cost, which is neg- ligible in our experiments on PASCAL VOC with 20 categories. To reduce the SVM-classification cost, applying dimension reduction techniques such as product quantiza- Speed-accuracy trade-off for different codebooktion to the final image representation can be effectively utilized.

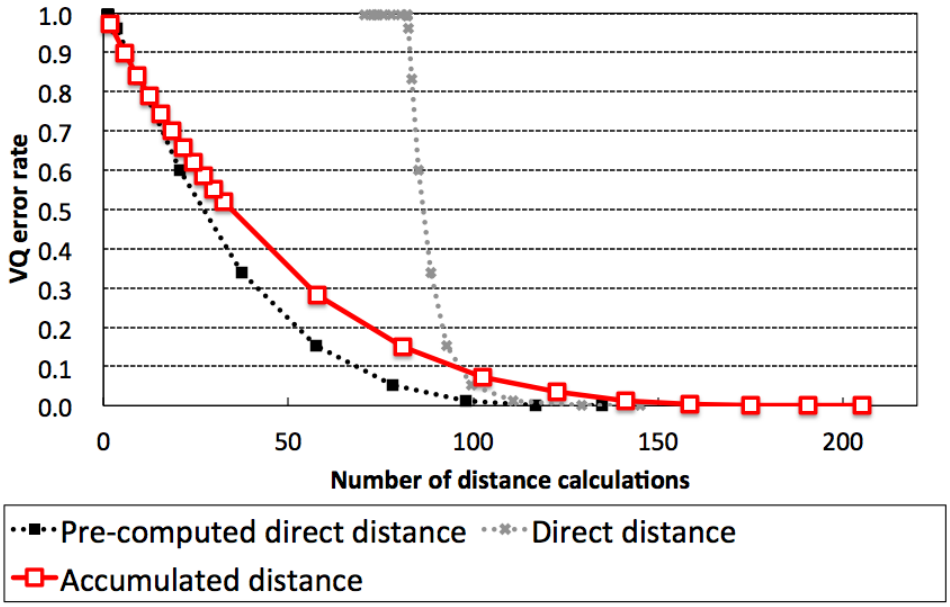

Figure 6.9: **Comparison of the accumulated distance and the direct distance.** VQ error rate in NTN-VQ for different values of δ is reported. All plots are for $\delta = 1.0, 0.9, \cdots 0.1, \ 0.09, \cdots 0.01$. Accumulated distance: Δ_{ij} is defined by Eq. (6.6). Direct distance: Δ_{ij} is replaced by the direct distance $\|x_i - x_j\|$. Pre-computed direct distance: the direct distance is used but distance calculations for it are not counted.

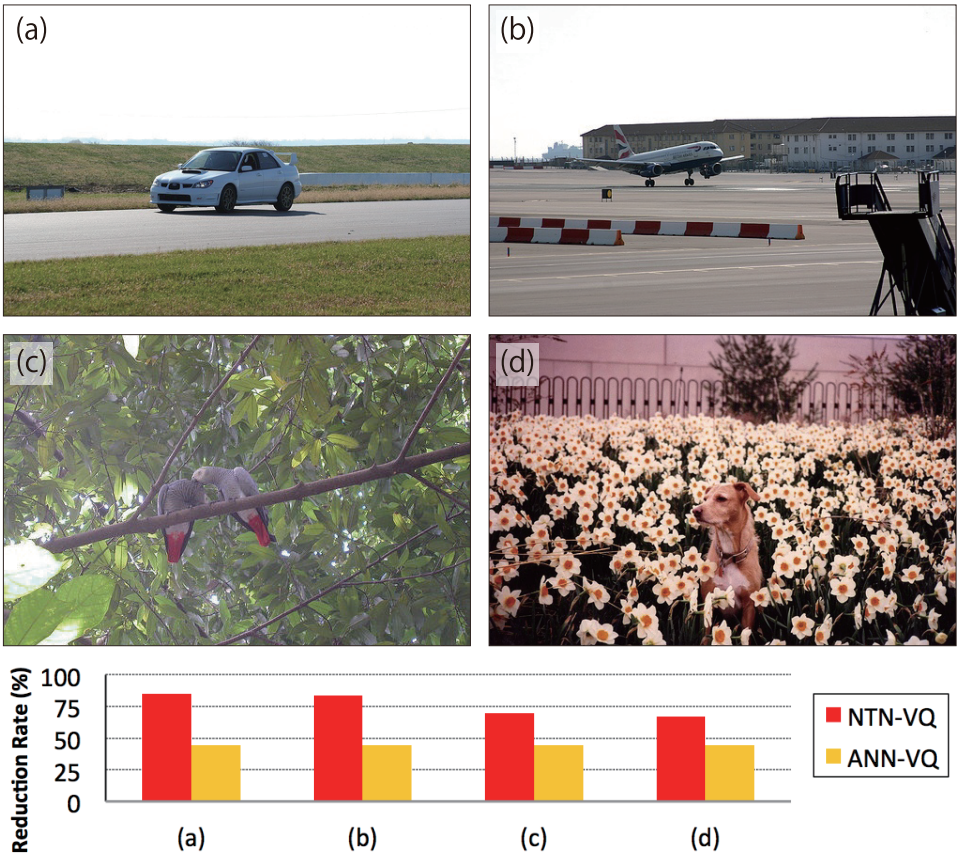

Figure 6.10: **The computational cost reduction by NTN-VQ for different images.** Four images (a), (b), (c), and (d) are from PASCAL VOC 2007. The reduction rate of the assignment cost by NTN-VQ and ANN-VQ for each image is reported.

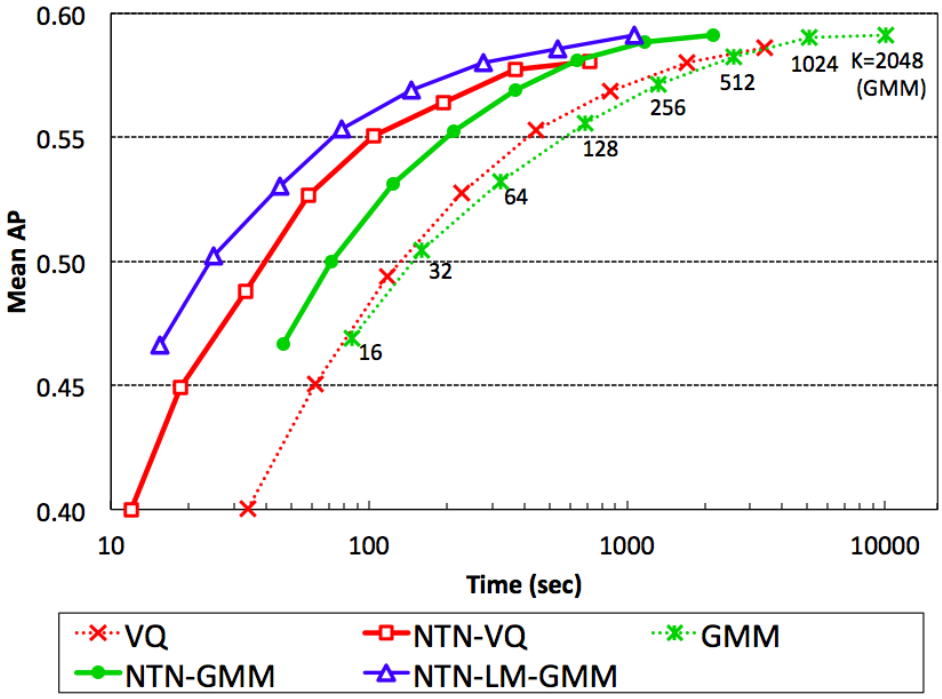

Figure 6.11: **Speed-accuracy trade-off for different codebook sizes.** Trade-off between assignment time and Mean AP for codebook sizes of $K = 2048, 1024, 512, \cdots, 16$. is reported. VQ: standard hard vector quantization (VQ), NTN-VQ: neighbor-to-neighbor (NTN) search for VQ $\delta = 0.20$, GMM: standard Gaussian mixture model (GMM), NTN-GMM: NTN search for a GMM $\delta = 0.09$, NTN-LM-GMM: NTN-GMM with the log-max approximation $\delta = 0.09$.

6.5 Conclusion

We have proposed a fast computation method for searching for the matches, neighbor-to-neighbor (NTN) search, and its applications to vector quantization (VQ) and a Gaussian mixture model (GMM). We tested NTN-VQ and NTN-GMM on super-vector coding and Fisher-vector coding, respectively. Our experiments on the PASCAL VOC 2007 classification challenge showed that NTN-VQ, NTN-GMM, and NTN-LM-GMM reduced the assignment cost by 77.4%, 75.3%, and 89.3%, respectively, without any significant degradation in the image classification performance. This result confirms the effectiveness of our proposed algorithms.

In future work, we will focus on the speeding up of the feature extraction step that we didn't discuss in this chapter. Approximation of dense sampling and SIFT descriptors would be interesting as promising next steps.

Chapter 7

Conclusion and Future Work

We have proposed an efficient and effective semantic indexing system. Our system extended the bag-of-visual-words system to a probabilistic framework in which a video segment is modeled by a GMM. By extracting audio and visual low-level features, we effectively detected semantic concepts from video data. A q-GMM, which is generalization of a GMM with a parameter q to control its tail-heaviness, further improved the modeling accuracy. In experiments on the TRECVID benchmark, we achieved Mean Average Precision (AP) of 0.178, and 0.321 on TRECVID 2011, and 2012 datasets, respectively, which is the best performance among official runs in the TRECVID workshop.

To improve the speed of the system, we have proposed fast MAP adaptation and Neighbor-to-Neighbor (NTN) search, which are complementary techniques to each other. For fast MAP adaptation, a tree-structured GMM was utilized to decrease the computational cost, where only the output probabilities of mixture components close to a feature vector are precisely calculated. The calculation time for MAP adaptation was reduced by 76.2% from the conventional method, while high detection performance was maintained on the TRECVID benchmark. The NTN search further improved the speed when densely sampled image descriptors are used in the low-level feature extraction step. Based on the fact that image features extracted from an adjacent region are usually similar to each other, the NTN algorithm effectively reduced the cost of calculating the distance between a codeword and a feature vector. In experiments, we applied the NTN search to vector quantization (NTN-VQ) and a GMM (NTN-GMM). Results on the PASCAL VOC

2007 classification challenge showed that NTN-VQ reduced the assignment cost by 77.4% in super-vector coding, and NTN-GMM reduced it by 89.3% in Fisher-vector coding, without any significant degradation in classification performance.

Our future work will focus on the relation between semantic concepts to extend our system to generate detailed meta data such as a text summarization of video. In this study, we focused on detecting *word-level* semantic concepts which is the most important part of meta-data in key-word search engines. However, what users require will probably become *sentence-level* semantic concepts such as complex events and human actions in near future. To detect the *sentence-level* semantic concepts, temporal and spatial localization of objects and motions is needed which is not discussed in this study. For example, temporal modeling using hidden Markov models (HMMs) or bayesian networks is expected to effectively capture the causality relation between semantic concepts. To improve the speed of detection, the next step would be the GPU implementation of our methods. We also would like to apply our techniques to other applications than semantic indexing such as robot serving, object tracking and video editing.

Bibliography

[1] M. Kankanhalli and Y. Rui, Application potential of multimedia information retrieval, In *Proc. IEEE MIR*, vol. 96, pp. 712-720, 2008.

[2] A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta, R. Jain Content-based image retrieval at the end of the early years, In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.22, no.12, pp.1349–1380, 2000.

[3] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *Proc. ECCV SLCV workshop*, pages 59–74, 2004.

[4] J. Yang and A. G. Hauptmann. Evaluating bag-of-visual-words representations in scene classification. In Proc. of *ACM Multimedia MIR* workshop, Augsburg, Germany, 2007.

[5] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[6] J. C. V. Gemert, J.-m. Geusebroek, C. J. Veenman, and A. W. M. Smeulders. Kernel codebooks for scene categorization. *In Proc. of ECCV*, pages 696–709, 2008.

[7] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. *Proc. CVPR*, pages 3360–3367, 2010.

[8] F. Perronnin, C. Dance, G. Csurka, and M. Bressan. Adapted vocabularies for generic visual categorization. *Proc. ECCV*, pages 464–475, 2006.

[9] C.G.M. Snoek, K.E.A. van de Sande, O. de Rooij, B. Huurnink, E. Gavves, D. Odijk, M. de Rijke, Th. Gevers, M. Worring, D.C. Koelma, and A.W.M. Smeulders The MediaMill TRECVID 2010

Semantic Video Search Engine. In Proc. of *TRECVID workshop*, Gaithersburg, MD, USA, 2010.

[10] S.-F. Chang, D. Ellis, W. Jiang, K. Lee, A. Yanagawa, A.C. Loui, and J. Luo, Large-scale Multimodal Semantic Concept Detection for Consumer Video. In Proc. of *ACM Multimedia MIR* workshop, Augsburg, Germany, 2007.

[11] W. Jiang, C. Cotton, S.-F. Chang, and D. Ellis, Short-Term Audio-Visual Atoms for Generic Video Concept Classification. In Proc. of *ACM Multimedia*, Beijing, China, 2009.

[12] M. Huijbregts, R. Ordelman, F. De Jong, Annotation of Heterogeneous Multimedia Content Using Automatic Speech Recognition, In *Proc. SAMT*, vol.4816, pp.78–90, 2007.

[13] A. F. Smeaton, P. Over, and W. Kraaij. High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements. In A. Divakaran, editor, *Multimedia Content Analysis, Theory and Applications*, pages 151–174, Springer Verlag, Berlin, 2009.

[14] C. Silpa-anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. *Proc. CVPR*, pages 1–8, 2008.

[15] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *Proc. VISAPP*, pages 331–340, 2009.

[16] M. Brown, and D.Lowe, Recognizing panoramas. In *Proc. ICCV*, pp.1218–1225, 2003.

[17] J. Matas, O. Chum, M. Urban, and T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, In Proc. BMVC, pp.384–393, 2002.

[18] P. Pritchett, and A. Zisserman, Wide baseline stereo matching, In *Proc. ICCV*, pp.754–760, 1998.

[19] T. Tuytelaars, L. V. Gool, L. Dhaene, and R. Koch, Matching of affinely invariant regions for visual servoing, In *Proc. ICRA*, 1999

[20] S. Lazebnik, C. Schmid, and J. Ponce, A sparse texture representation using affine-invariant regions, In *Proc. CVPR*, 2003.

[21] S. Lazebnik, C. Schmid, and J. Ponce, Affine-invariant local descriptors and neighborhood statistics for texture recognition, In *Proc. ICCV*, pp.649–655, 2003.

[22] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. In *IJCV*, vol. 60(1), pages 63–86, 2004.

[23] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. *Proc. ECCV*, pages 141–154, 2010.

[24] I. Laptev and T. Lindeberg, Space-time interest points In *Proc. ICCV*, pp.432–439, 2003.

[25] H. Wang, A. Kl, C. Schmid, C.-lin Liu, Action Recognition by Dense Trajectories In *Proc. CVPR*, pp.3169–3176, 2011.

[26] N. Dalal, W. Triggs, C. Schmid, S. Soatto, and C. Tomasi, Histograms of Oriented Gradients for Human Detection, In Proc. CVPR, pp.886–893, 2005

[27] X. Wang, T. X. Han, S. Yan, An HOG-LBP human detector with partial occlusion handling, In Proc. ICCV, pp.32–39, 2009.

[28] J. van de Weijer and C. Schmid. Coloring local feature extraction. In Proc. of *ECCV*, pp. 334–348, 2006.

[29] K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 32(9), pp. 1582–1596, 2010.

[30] M. Calonder, V. Lepetit and C. Strecha and P. Fua, BRIEF : Binary Robust Independent Elementary Features, In Proc. ECCV, 2010.

[31] E. Rublee, V .Rabaud, K. Konolige and G. Bradski, ORB: an efficient alternative to SIFT or SURF, In Proc. ICCV, 2011.

[32] S. N. Sinha, J. michael Frahm, M. Pollefeys, and Y. Genc. Gpu-based video feature tracking and matching. In Proc. of *EDGE* workshop, Chapel Hill, NC, USA, 2006.

[33] T. Huang. Linear spatial pyramid matching using sparse coding for image classification. *Proc. CVPR*, pages 1794–1801, 2009.

[34] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. *In Proc. of CVPR*, pages 3304–3311, 2010.

[35] T. Jaakkola, and D. Haussler, Exploiting Generative Models in Discriminative Classifiers In Proc. of *NIPS*, pp. 487–493, 1998.

[36] F. Perronnin, and C. Dance, Fisher kernels on visual vocabularies for image categorization. In Proc. of *CVPR*, pp. 1–8, 2007.

[37] F. Perronnin, S. Jorge, and T. Mensink. Improving the fisher kernel for large-scale image classification. *Proc. ECCV*, pages 143–156, 2010.

[38] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The devil is in the details: an evaluation of recent feature encoding methods. *Proc. BMVC*, pages 1–12, 2011.

[39] M. Havrda and F. Charvat. Quantification method of classification processes: concept of structural α-entropy, In *Kybernetika*, vol. 3, pp. 30–35, 1967.

[40] C. Tsallis, Possible generalization of boltzmann-gibbs statistics, In *Journal of Statistical Physics*, vol. 52, pp. 479–487, 1988.

[41] C. Tsallis, R. S. Mendes, A. R. Plastino, The role of constraints within generalized nonextensive statistics, In Physica A, vol. 261, no. 3, pp. 534-554, 1988.

[42] M. P. de Albuquerque, I. A. Esquef, and A. R. G. Mello, Image thresholding using Tsallis entropy. In Elsevier Pattern Recognition Letters, vol. 25, pp. 1059–1065, 2004.

[43] P. K. Sahoo, and G. Arora, Image thresholding using two-dimensional Tsallis-Havrda-Charvat entropy, In Elsevier Pattern Recognition Letters, vol. 27, issue. 6, pp. 520–528, 2006.

[44] Q. Lin, and C. Ou, Tsallis entropy and the long-range correlation in image thresholding, In Elsevier Signal Processing, vol. 92, pp. 2931–2939, 2012.

[45] Y. Li, X. Fan, and G. Li. Image segmentation based on Tsallis-entropy and Renyi-entropy and their comparison. In Proc. of *ICII*, pp. 943–948, 2006.

[46] R. Fabbrib, W. N. Goncalvesa, F. J. P. Lopesc, and O. M. Brunoa, Multi-q pattern analysis: A case study in image classification, In Elsevier Physica A: Statistical Mechanics and its Applications, vol. 391, issue. 19, pp. 4487–4496, 2012.

[47] D. Gerogiannis, C. Nikou, and A. Likas Robust Image Registration using Mixtures of t-distributions, In Proc. of *ICCV*, 2007.

[48] T. M. Nguyen and Q. M. J. Wu, Robust Student's-t Mixture Model With Spatial Constraints and Its Application in Medical Image Segmentation, In IEEE Trans. on Medical Imaging, vol.31, no.1, 2012.

[49] N. Johnson, S. Kotz, and N. Balakrishnan, Discrete Multivariate Distributions, New York: Wiley Interscience, 1997.

[50] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *Proc. CVPR*, pages 2161–2168, 2006.

[51] J. C. Platt, Probabilities for SV machines, In Advances in Large Margin Classifiers, pp.61–74, The MIT Press, 2000.

[52] H.-T. Lin, C.-J. Lin, and R. C. Weng, A note on Platt's probabilistic outputs for support vector machines, In Machine Learning, vol. 68, pp. 267276, 2007.

[53] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, Local features and kernels for classification of texture and object categories: A comprehensive study, In International Journal of Computer Vision, vol. 73, pp. 213-238, 2007.

[54] Y. Rubner, C. Tomasi, and L. J. Guibas, The earth mover's distance as a metric for image retrieval, International Journal of Computer Vision, vol. 40, pp. 99121, 2000.

[55] F. R. Back and G. R. G. Lanckriet. Multiple Kernel Learning, Conic Duality, and the SMO Algorithm In Proc. of *ICML*, Banff, Alberta, Canada, 2004.

[56] F. Yan, J. Kittler, K. Mikolajczyk, and A. Tahir. Non-sparse multiple kernel learning for fisher discriminant analysis. In Proc. of *ICDM*, Miami, Florida, USA, 2009.

[57] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In Proc. of *ICCV*, Rio de Janeiro, Brazil, 2007.

[58] A. P. Dempster, N. M. Laird, and D. B. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society*, vol.Series B, 39, no.1, pp.1–38, 1977.

[59] V. N. Vapnik, Statistical learning theory, Wiley, New York, 1 ed., Sept. 1998.

[60] N. Cristianini and J. Shawe-Taylor, Support Vector Machines, Cambridge University Press, Cambridge, 2000.

[61] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, *Data Min. Knowl. Discov.*, vol.2, pp.121–167, 1998.

[62] W. M. Campbell, D. E. Sturim, and D. A. Reynolds. Support vector machines using GMM supervectors for speaker verification. In *IEEE Signal Processing Letters*, vol. 13, pp. 308–311, 2006.

[63] X. Zhou, X. Zhuang, S. Yan, S.-F. Chang, M. H.-Johnson, and T. S. Huang. Sift-bag kernel for video event analysis. In Proc. of *ACM Multimedia*, Vancouver, British Columbia, Canada, 2008.

[64] A. F. Smeaton, et al. Evaluation campaigns and trecvid. In Proc. of *ACM Multimedia MIR workshop*, pp. 321–330, 2006.

[65] S. Ayache and G. Quénot. Video Corpus Annotation Using Active Learning. In Proc. of ECIR, 2008.

[66] E. Yilmaz, E. Kanoulas, and J. A. Aslam. A simple and efficient sampling method for estimating ap and ndcg. In Proc. of *ACM SIGIR*, Singapore, 2008.

[67] C. Wu. SiftGPU: A GPU implementation of sift. http://cs.unc.edu/~ccwu/siftgpu, 2007.

[68] Koen E. A. van de Sande, Theo Gevers, Cees G. M. Snoek. Empowering Visual Categorization With the GPU. In *IEEE Transactions on Multimedia*, vol. 13 (1), pages 60–70, 2011.

[69] S. J. Young, G. Evermann, M. J. F. Gales, D. Kershaw, G. Moore, J. J. Odell, D. G. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. The htk book, version 3.4, 2006.

[70] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.

[71] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/

[72] M. Everingham, A. Zisserman, C. Williams, and L. V. Gool. The pascal visual obiect classes challenge 2007 (voc2007) results. http://www.pascal-network.org/challenges/VOC/voc2007/, 2007.

[73] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

List of Publications

Journal Paper

1. Nakamasa Inoue and Koichi Shinoda, q-Gaussian Mixture Models for Image And Video Semantic Indexing, *In Elsevier Journal of Visual Communication and Image Representation*, vol.24, no.8 pp.1450-1457, 2013.
2. Yusuke Kamishima, Nakamasa Inoue and Koichi Shinoda, Event Detection in Consumer Videos Using GMM Supervectors and SVMs, *In EURASIP Journal on Image and Video Processing*, vol.2013:51, pp.1–13, 2013.
3. Nakamasa Inoue and Koichi Shinoda, A Fast and Accurate Video Semantic-Indexing System Using Fast MAP Adaptation and GMM Supervectors, *In IEEE Transactions on Multimedia*, vol.14, no.4–2, pp.1196–1205, 2012.
4. Nakamasa Inoue, Tatsuhiko Saito, Koichi Shinoda, and Sadaoki Furui, Multimodal High-Level Feature Extraction for Large-Scale Video Resources *In IEICE Transactions on Information and Systems*, vol.J93–D, no.12, pp.2633–2644, 2010, in Japanese.

International Conference Paper (Reviewed)

1. Liang Zhuolin, Nakamasa Inoue, and Koichi Shinoda, Velocity Pyramid for Multimedia Event Detection, *In Proc. of MMM*, Jan., 2014.
2. Nakamasa Inoue, and Koichi Shinoda, Neighbor-To-Neighbor Search for Fast Coding of Feature Vectors *In Proc. of ICCV*, pp.1233-1240, Dec., 2013.
3. Nakamasa Inoue and Koichi Shinoda, q-Gaussian Mixture Models Based on Non-Extensive Statistics for Image And Video Semantic Indexing, *In Proc. of Asian Conference on Computer Vision (ACCV)*, pp.1–12 (PT1–20), Korea, Nov., 2012.

4. Yusuke Kamishima, Nakamasa Inoue, Koichi Shinoda, and Shunsuke Sato, Multimedia Event Detection Using GMM Supervectors and SVMs, *In Proc. of ICIP*, USA, pp.3089–3092, Oct., 2012.
5. Nakamasa Inoue and Koichi Shinoda, A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing, *In Proc. of ACM Multimedia*, USA, pp.1357–1360, Nov., 2011.
6. Nakamasa Inoue, Tatsuhiko Saito, Koichi Shinoda, and Sadaoki Furui, High-Level Feature Extraction using SIFT GMMs and Audio Models, *In Proc. of ICPR*, Turkey, pp.3220–3223, Aug., 2010.

International Workshop Paper

1. Nakamasa Inoue, Kotaro Mori, Liang Zhuolin, and Koichi Shinoda, Semantic Indexing Using GMM Supervectors and Video-Clip Scores (TokyoTechCanon at TRECVID 2013), *In Proc. of TREC Video Retrieval Evaluation* (*TRECVID*) *workshop*, USA, Nov., 2013.
2. Nakamasa Inoue, Yusuke Kamishima, Kotaro Mori, and Koichi Shinoda, Multimedia Event Detection Using GMM Supervectors and Camera Motion Cancelled Features (TokyoTechCanon at TRECVID 2012), *In Proc. of TREC Video Retrieval Evaluation* (*TRECVID*) *workshop*, USA, Nov., 2012.
3. Nakamasa Inoue and Koichi Shinoda, Video Semantic Indexing Using GMM-Supervectors, Greater Tokyo Area Multimedia/Vision Workshop, Japan, Aug, 2012.
4. Nakamasa Inoue, Toshiya Wada, Yusuke Kamishima, Koichi Shinoda, and Shunsuke Sato, Semantic Indexing Using GMM Supervectors and Tree-structured GMMs (TokyoTech+Canon at TRECVID 2011), *In Proc. of TREC Video Retrieval Evaluation* (*TRECVID*) *workshop*, USA, Dec., 2011.
5. Nakamasa Inoue, Toshiya Wada, Yusuke Kamishima, Koichi Shinoda, Ilseo Kim, Byungki Byun, and Chin-Hui Lee, Semantic Indexing Using GMM Supervectors with MFCCs and SIFT features (TT+GT at TRECVID 2010 Workshop), *In Proc. of TREC Video Retrieval Evaluation* (*TRECVID*) *workshop*, USA, Nov., 2010.
6. Nakamasa Inoue, Shanshan Hao, Tatsuhiko Saito, Koichi Shinoda, Ilseo Kim and Chin-Hui Lee, High-Level Feature Extraction Using SIFT

GMMs, Audio Models, and MFoM (TITGT at TRECVID 2009 Workshop), *In Proc. of TREC Video Retrieval Evaluation (TRECVID) workshop*, USA, pp.373–380, Nov., 2009.

Domestic Conference Paper

1. 井上 中順, 篠田 浩一, GMM Supervector とビデオクリップスコアを用いた映像のセマンティックインデクシング, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.112, no.441, pp.173–178, Feb., 2013.
2. 上嶋 勇祐, 井上 中順, 篠田 浩一, カメラの動き補正に基づく時空間特徴量と GMM supervector を用いた映像からのイベント検出, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.112, no.441, pp.185–190, Feb., 2013.
3. 井上 中順, 篠田 浩一, 映像のセマンティックインデクシングのための q-混合ガウス分布, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.112, no.197, pp.31–36, Sep., 2012.
4. 井上 中順, 篠田 浩一, GMM-Supervector を用いた高速な映像のセマンティック検索システム, 画像の認識・理解シンポジウム (MIRU), pp.DS–09, Aug., 2012.
5. 井上 中順, 篠田 浩一, GMM-supervector を用いた映像の高速セマンティック検索システム〜映像の意味に基づいた検索を実現〜, 画像センシングシンポジウム (SSII), pp.DS2–08, Jul., 2012.
6. 上嶋 勇祐, 井上 中順, 篠田 浩一, GMM-Supervector と SVM を用いた映像からのイベント検出, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.111, no.430, pp.195–200, Feb., 2012.
7. 井上 中順, 篠田 浩一, 木構造 GMM を用いたセマンティックインデクシングの高速化, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.111, no.77, pp.105–110, Jun., 2011.
8. 井上 中順, 上嶋 勇祐, 篠田 浩一, マルチモーダル・マルチフレームな手法を用いた TRECVID セマンティックインデクシング, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.110, no.414, pp.25–30, Feb., 2011.
9. 井上 中順, 上嶋 勇祐, 篠田 浩一, SIFT 混合ガウス分布を用いた一般物体認識のためのマルチカーネル学習, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.110, no.187, pp.7–12, Sep., 2010.

10. 斉藤 辰彦, 井上 中順, 篠田 浩一, 古井 貞熙, 音響特徴を用いた映像からのイベント検出の研究, 日本音響学会 2010 年 春季研究発表会, 日本音響学会講演論文集, pp.201-202, Mar, 2010.
11. 井上 中順, 斉藤 辰彦, 篠田 浩一, 古井 貞熙, SIFT 混合ガウス分布と音響特徴を用いた映像からの高次特徴検出, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.109, no.306, pp.97–102, Nov., 2009.

Invited Talk

1. 井上 中順, 篠田浩一, ［特別講演］映像の高性能なセマンティックインデクシングを目指して, 電子情報通信学会 パターン認識・メディア理解研究会 (PRMU), vol.111, no.353, pp.89–94, Dec., 2011.

Article

1. Koichi Shinoda and Nakamasa Inoue, Reusing Speech Techniques for Video Semantic Indexing, *In IEEE Signal Processing Magazine*, vol.30, no.2, pp.118–122, 2013.

Award

2011 年 パターン認識・メディア理解研究会 研究奨励賞