TokyoTechCanon at TRECVID 2013

Nakamasa Inoue, Kotaro Mori, Zhuolin Liang,
Mengxi Lin,Koichi Shinoda
Department of Computer Science, Tokyo Institute of Technology.
{inoue, mori, zhuolin, mengxi}@ks.cs.titech.ac.jp
shinoda@cs.titech.ac.jp

Shunsuke Sato

Canon Inc.
sato.shunsuke@canon.co.jp

1 Semantic Indexing

We aim at developing a high-performance system using Gaussian-mixture-model (GMM) supervectors and tree-structured GMMs [6, 7, 8] for the semantic indexing task [1, 2, 3, 4]. GMM supervectors corresponding to six types of audio and visual features are extracted from video shots. Tree-structured GMMs reduce the computational cost of maximum a posteriori (MAP) adaptation for estimating GMM parameters while keeping accuracy at high levels.

This year, we improve our re-scoring method using video-clip scores by introducing a scaling parameter. Here, the video-clip score is defined as the maximum value of shot scores among all the shots in a video clip. Our best result was 28.4% in terms of Mean InfAP, which was ranked third among participating teams in the semantic indexing task.

1.1 Low-Level Feature Extraction

The following six types of visual and audio features are extracted from video data.

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

   Scale Invariant Feature Transform (SIFT) proposed by Lowe [9] is a local feature extraction method that is widely used for object categorization since it is invariant to image scaling and changing illumination. The Harris-Affine detector [10], which is an extension of the Harris corner detector, improves the robustness against affine transform of local regions. SIFT features are extracted from every other frame, and principal component analysis (PCA) is applied to reduce their dimensions from 128 to 32.

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

   SIFT features are extracted with the Hessian-Affine detector [10], which is complementary to the Harris-Affine detector. The combination of several different detectors can improve the robustness against noise. SIFT features are extracted from every other frame, and PCA is applied to reduce their dimensions from 128 to 32.

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

   SIFT features and 36-dimensional hue histograms [11] are combined to capture color information. SIFT+Hue features are extracted from key-frames by using dense sampling (100x100 grid with 3 scales). PCA is applied to reduce dimensions from 164 to 32.

4. HOG with dense sampling (HOG-Dense)

   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.

5. LBP with dense sampling (LBP-Dense)

   Local Binary Patterns (LBPs) [12] 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 [12] to extract LBP features. PCA is applied to reduce dimensions from 228 to 32.

6. MFCC audio features (MFCC)

   Mel-frequency cepstral coefficients (MFCCs), which describe the short-time spectral shape of audio frames, are extracted to capture audio information. MFCCs are widely used not only for speech recognition but also for generic audio classification. Δ MFCCs, $\Delta\Delta$ MFCCs, Δ log-power and $\Delta\Delta$ log-power are extracted in addition to the MFCCs. Here, "Δ" means the derivation of the feature. The dimension of the audio feature is 38, including 12-dimensional MFCCs.

1.2 Gaussian Mixture Models

Gaussian mixture models (GMMs), whose probability density function (pdf) is given by

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

are used to model video shots. Here, x is a local 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.

The GMM parameters are estimated for each shot under the maximum a posteriori (MAP) criterion. The MAP solution for GMM means, namely MAP adaptation, is given by

$$\hat{\mu}_k = \frac{\tau \hat{\mu}_k^{(\mathrm{U})} + \sum_{i=1}^{n} c_{ik} x_i}{\tau + C_k}, \quad c_{ik} = \frac{w_k g_k(x_i)}{\sum_{k=1}^{K} w_k g_k(x_i)}, \quad C_k = \sum_{i=1}^{n} c_{ik}, \quad g_k(x) = \mathcal{N}(x|\hat{\mu}_k^{(\mathrm{U})}, \hat{\Sigma}_k^{(\mathrm{U})}), \tag{2}$$

where $X_{\mathrm{F}} = \{x_i\}_{i=1}^{n}$ ($\mathrm{F} \in \{\text{SIFT-Har, SIFT-Hes, SIFTH-Dense, HOG-Dense, LBP-Dense, MFCC}\}$) 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.

1.3 Fast MAP Adaptation

The fast MAP adaptation technique [6, 7] reduces computational costs for calculating posterior probabilities c_{ik} in Eq. (9) by constructing a tree-structured GMM. The basic idea of the tree-structured GMM is to cluster Gaussian components and approximate them with a single Gaussian. Each leaf node corresponds to a Gaussian component of the UBM, and each non-leaf node has a single Gaussian that approximates its descendant Gaussian components.

The tree-structured GMM $\mathcal{T}$ is defined as

$$\mathcal{T} = (V, E, G_{\mathrm{TREE}}), \tag{3}$$

where V is a set of nodes, E is a set of edges, and G_{TREE} is a set of Gaussian components for nodes in V. Here, $g^{(v)} \in G_{\mathrm{TREE}}$ denotes a Gaussian component for $v \in V$. The Gaussian components for the UBM, $G_{\mathrm{UBM}} = \{g_k\}_{k=1}^{K}$, are assigned to leaf nodes, i.e., for each leaf $\ell \in V$, there exists $g_k \in G_{\mathrm{UBM}}$ that satisfies $g^{(\ell)} = g_k$. Gaussian components for non-leaf nodes and their children are determined by applying k-means clustering to G_{UBM} (see [6, 7] for details).

With the tree-structured GMM, posterior probabilities c_{ik} in Eq. (9) are calculated only for *active* Gaussian components. The following algorithm finds active leafs by expanding active nodes V_A from the root node to output c_{ik} quickly.

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:

$$V_A \leftarrow \bigcup_{v \in V_A} C(v), \tag{4}$$

   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. Calculate posterior probabilities $c_i^{(v)}$ for an active GMM given by

$$p(x|V_A) = \sum_{v \in V_A} \tilde{w}^{(v)} g^{(v)}(x), \quad \tilde{w}^{(v)} = \frac{w^{(v)}}{\sum_{v \in V_A} w^{(v)}}, \tag{5}$$

   i.e., calculate

$$c_i^{(v)} = \frac{\tilde{w}^{(v)} g^{(v)}(x_i)}{\sum_{v \in V_A} \tilde{w}^{(v)} g^{(v)}(x_i)} = \frac{w^{(v)} g^{(v)}(x_i)}{\sum_{v \in V_A} w^{(v)} g^{(v)}(x_i)}. \tag{6}$$

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

$$V_A \leftarrow \{v \in V_A \mid c_i^{(v)} > c_{\text{TH}}\}. \tag{7}$$

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

$$\hat{c}_{ik} = \begin{cases} c_i^{(\ell)} & (\ell \in V_A,\ g^{(\ell)} = g_k) \\ 0 & (\text{otherwise}) \end{cases}. \tag{8}$$

Otherwise, return to Step 2.

Finally, MAP adaptation is given by

$$\hat{\mu}_k = \frac{\tau \hat{\mu}_k^{(\text{U})} + \sum_{\hat{c}_{ik} \neq 0} \hat{c}_{ik} x_i}{\tau + \hat{C}_k}, \quad \hat{C}_k = \sum_{\hat{c}_{ik} \neq 0} \hat{c}_{ik}. \tag{9}$$

1.4 GMM Supervector SVM

After video shots are represented by GMMs, GMM supervectors are extracted by combining normalized mean vectors as

$$\phi(X_{\text{F}}) = \begin{pmatrix} \tilde{\mu}_1 \\ \tilde{\mu}_2 \\ \vdots \\ \tilde{\mu}_K \end{pmatrix}, \quad \tilde{\mu}_k = \sqrt{w_k^{(U)}} (\Sigma_k^{(U)})^{-\frac{1}{2}} \hat{\mu}_k. \tag{10}$$

Support vector machines (SVMs) with the following RBF-kernel are used to train discriminative models for each semantic concepts.

$$k(X_{\text{F}}, X'_{\text{F}}) = \exp\left(-\gamma \|\phi(X_{\text{F}}) - \phi(X'_{\text{F}})\|_2^2\right), \quad \gamma = \frac{1}{\tilde{d}}, \tag{11}$$

where $\tilde{d}$ is the average distance between two GMM supervectors. Here, annotations are obtained from the collaborative annotations [4]. Finally, trained discriminative functions are linearly combined as

$$f(X) = \sum_{\text{F} \in \mathcal{F}} \alpha_{\text{F}} f_{\text{F}}(X_{\text{F}}), \quad 0 \leq \alpha_{\text{F}} \leq 1, \quad \sum_{\text{F}} \alpha_{\text{F}} = 1. \tag{12}$$

where $\mathcal{F} = \{\text{SIFT-Har}, \text{SIFT-Hes}, \text{SIFTH-Dense}, \text{HOG-Dense}, \text{LBP-Dense}, \text{MFCC}\}$. Combination coefficients α_{F} are optimized on a validation set.

1.5 Video-Clip Scores

The relationship between shots are useful for detecting semantic concepts. For example, Safadi et al. [13] proposes a re-ranking method to re-evaluate scores of video shots by using shot-score distributions. In our re-ranking method, we define a video-clip score as the maximum value of shot scores among all the shots in a video clip:

$$s_{\max} = \max_i s_i \tag{13}$$

where $s_i (i = 1, 2, \cdots, n)$ are shot scores for a video-clip that consists of n shots. Our final score for ranking shots is given by

$$s'_i = (1 - p) s_i + p s_{\max} \tag{14}$$

where p is a probability of appearance of a semantic concept in a video clip given by

$$p = r \left\langle \frac{\#(\text{positive shots in a video clip})}{\#(\text{shots in a video clip})} \right\rangle. \tag{15}$$

where r is a scaling parameter. The final score s'_i gets closer to $s_{\max}$ as the concept appear more often (e.g. an anchorperson in a news video). It gets closer to the original shot score s_i for concepts appear in few times (e.g. a bus in a street video).

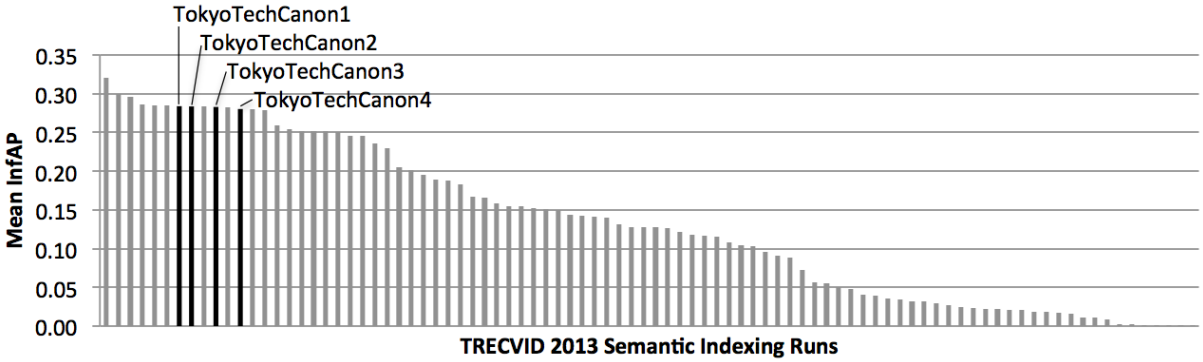

Figure 1: Overview of results of the semantic indexing task in TRECVID 2013. Our best result was Mean InfAP of 28.4%.

1.6 Experimental Conditions

Mikolajczyk's implementation [10] was used to extract SIFT-Har and SIFT-Hes features. SIFT++ [14] was used to extract SIFTH-Dense features. HTK [15] (speech recognition toolkit) was used to extract MFCC. The sum of calculation time (for PCA projection, MAP adaptation, and SVM prediction) was reduced from 2.47 sec to 1.00 sec (59.5%) by using the fast MAP adaptation technique. The calculation time was measured by using a Intel Xeon 2.93 GHz CPU.

The following four runs are submitted to the TRECVID 2013 semantic indexing [1, 2, 3].

M_A_TokyoTech_Canon_4

This run used GMM supervector SVMs with the six types of features described in Section 1.1. The number of Gaussian components was $K = 512$ for the visual features, and $K = 256$ for the audio feature. The UBMs were trained using 1,000,000 samples. Optimal tree structures for the UBMs were selected as to minimize computational costs for MAP adaptation (see [6, 7]). The hyper-parameter τ for MAP adaptation was set to 20.0. Weights α_{F} for fusion are optimized on IACC_1_B dataset. The scaling parameter r for video-clip scores is set to 1.0.

M_A_TokyoTech_Canon_3

This run added a spatial and velocity pyramid for HOG features to M_A_TokyoTech_Canon_4.

M_A_TokyoTech_Canon_2

This run used $r = 0.9$ for the scaling parameter.

M_A_TokyoTech_Canon_1

This run used $r = 0.8$ for the scaling parameter, which is the best choice on TRECVID 2011 dataset.

1.7 Results

Figure 1 shows the overview of results of the semantic indexing task. Our best result by the run of F_A_TokyoTech_Canon_1 was 28.4% in terms of Mean InfAP, which is ranked 7th among 90 runs and is ranked 3rd among participating teams. M_A_TokyoTech_Canon_2 to 4 achieved Mean InfAP of 28.4%, 28.3%, and 28.0%, respectively. There was no significant difference between them.

Figure 2 shows InfAP by semantic concepts. One of our runs achieved the best performance for "dancing", "Instrumental_Musician", and "George_Bush". For these semantic concepts, audio information captured by our MFCC features improved the detection performance.

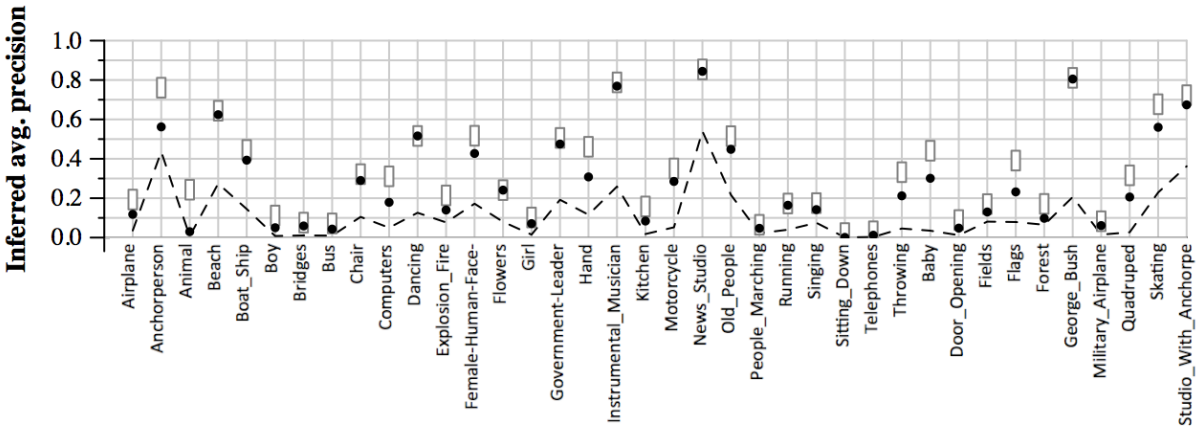

Figure 2: InfAP by semantic concept.

1.8 Conclusion

We proposed a high-performance semantic indexing system using Gaussian mixture model (GMM) supervectors with the six audio and visual features. Our best result was 28.4 % in terms of Mean InfAP, which was ranked third among participating teams in the semantic indexing task. Our future work will focus more on the spatio-temporal analysis for video indexing.

2 Multimedia Event Detection

2.1 Introduction

As the evaluation of 2011 and 2012, we applied GMM supervectors and SVMs to the MED task [5]. This year we improve the detection performance by introducing a motion feature, velocity pyramid, which is inspired by spatial pyramid. Besides, we continue using multiple kinds of features, including both static and dynamic visual features, as well as audio features. We submitted runs under the condition of Ex100, PROGAll and FullSys for both Pre-Specified task and Ad-Hoc task. For Pre-Specified task, we have an MAP of 0.245, which experienced a large degradation from last year (0.317), while the other team's MAPs are not so much different. We analyze the possible reasons in Section 2.5.

2.2 Features

The features we used consist of the following types. (1) Low-level static visual features, including HOG, SIFT-Har, SIFT-Hes and RGBSIFT. And all of them are applied spatial pyramid. (2) Motion features, including STIP, CC-DSTIP(camera motion canceled dense STIP) and HOG-VP(velocity pyramid of HOG). (3) Audio features, including MFCC.

2.2.1 Low-level static visual features

1. SIFT features with Harris-Affine detector and Hessian-Affine detector(SIFT-Har, SIFT-Hes)

   We use two kinds of detectors for local interest point detection. The first one is Harris-Affine detector [10] which is for corner detection. The second one is Hessian-Affine detector [10] which is for blob detection. The two detectors are robust for affine transformations and are expected to be complementary to each other.

   We extract Scale-Invariant Feature Transform (SIFT) [9] features from interest points detected by the two detectors, and refer to the two combinations as SIFT-Har and SIFT-Hes, respectively. Local image patches around the interest points are divided into 4×4 blocks, and an 8-bin histogram of gradient is calculated for each block. Thus a 128 dimensional vector is obtained for each key point detector. To reduce computation cost, we extract SIFT features from a frame every two seconds. After feature extraction, we apply Principal Components Analysis (PCA) and reduce the dimension of the feature vector to 32. In this way we can save computation cost during training and detection.

Feature	Mean Average Precision
HOG-original	0.217
HOG-SP	0.268
HOG-VP	0.261
HOG-SP+HOG-VP	**0.282**

Table 1: The MAP of HOG original features, HOG spatial pyramid, HOG velocity pyramid, and their combination on MED11 dataset. SP = spatial pyramid. VP = velocity pyramid.

2. HOG features with dense sampling (HOG)

   We also use histogram of oriented gradients (HOG) [21] features which is a kind of dense feature. Dense features have shown improved results than sparse interest points in several applications [26]. One reason is that sparse interest point approach may lose the information in homogeneous regions. We sample keypoints densely from an image frame at an interval of every 4 pixels. Then the region around a keypoint is divided to 2×2 blocks and an 8-bin histogram is calculated for each block, resulting a 32-dimensional vector. Like SIFT, we also extract HOG features from one frame every two seconds. PCA is then used to the HOG feature without reducing its dimension.

3. RGB-SIFT features with dense sampling (RGB-SIFT)

   Color information provides an important cue for video analysis. We capture the color information by RGB-SIFT features. An image is separated into RGB channels, and SIFT features are calculated from each of the three channels and then concatenated, resulting a 384 dimensional vector ($= 128 \times 3$). To reduce computation cost and storage requirement, we extract RGB-SIFT features from one frame every six seconds. PCA is applied to reduce its dimension to 64.

We applied spatial pyramids [23] to all the above features. The video volumes are divided into sub-volumes and feature vectors from each sub-volume are encoded and concatenated. Sub-region dividing used in spatial pyramid plays an important role. In [25], 2x2 grids gives good result, and 3x1 is believed to be able to capture layout of natural scenes. Therefore, we use three different pyramid levels: 1x1, 2x2 and 3x1. We refer to the spatial pyramid of the above features as SIFThar-SP, SIFThes-SP, HOG-SP and RGBSIFT-SP.

2.2.2 Motion features

1. STIP features with Harris 3D detector (STIP)

   We use space-time interest points (STIP) [22] as one of the motion features. Spatio-temporal interest points are detected by Harris 3D detectors, indicating regions with rapid velocity change. For descriptors we use a concatenation of 72-dimensional HOG feature with 90-dimensional histogram of optical flow (HOF) feature. The 162-dimensional vector is further reduced its dimension to 64 by applying PCA.

2. Camera motion canceled STIP features with dense sampling (CC-DSTIP)

   Since many videos in MED dataset are taken from unconstrained conditions, camera motion is unavoidable and thus degrades performance. For example, when calculating the HOF features, we observe lots of optical flow vectors are caused by camera motion. In order to separate true foreground motion from camera motion, we applied the camera motion canceled dense STIP features. We estimate camera motion of each video frame by optical flows. For each frame, a single motion vector is calculated and the frame is moved according to this vector. Then all aligned frames from one video are rewritten into a camera motion canceled video and dense STIP features are extracted from this new video. The feature vector also has a dimension of 162, and is reduced to 64 dimensions after applying PCA. The details of this method can be found in [27]. The CC-DSTIP is proved to be effective than DSTIP. Besides, STIP and CC-DSTIP are complementary to each other.

3. Velocity Pyramid of HOG (HOG-VP)

   We also applied velocity pyramid which was inspired by spatial pyramid. Instead of dividing a video volume into sub-volumes according to spatial layout, we divide the video volume based on motion information. In this way, the dynamic nature can be fully utilized.

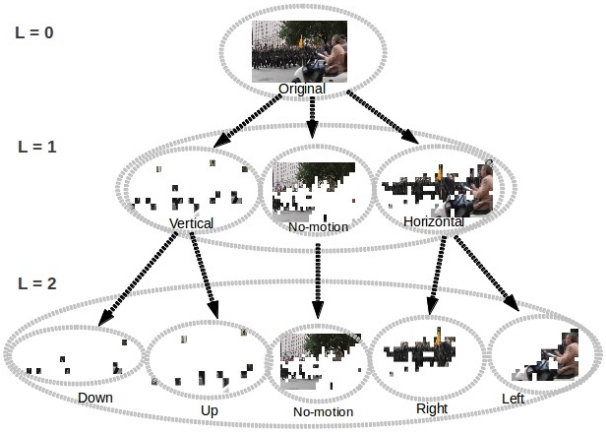

Figure 3: Velocity pyramids.

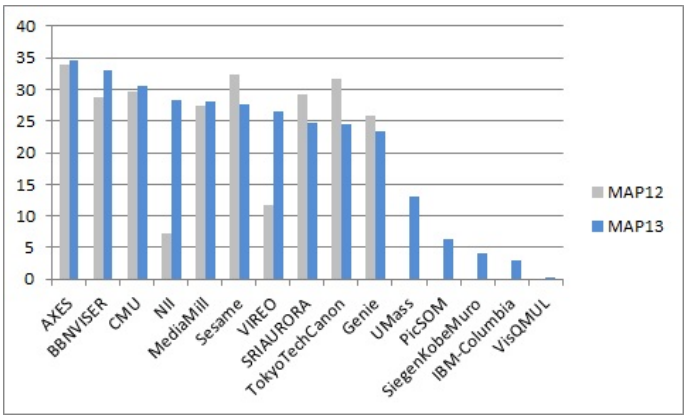

Figure 4: The comparison of MAP between 2012 and 2013 for Pre-Specified task.

For each densely sampled point from one image frame, we extract both a HOG feature and a motion vector (an optical flow). According to the orientation of the motion vectors, we categorize sample points into five groups (left, right, up, down and no-motion). For each orientation group, we accumulate features over all frames of a video clip to get an appearance feature set, and form its representation. The representations of the five orientation groups are concatenated to form a representation of the video clip.

We determine the pyramid level in velocity pyramid by the resulted number of bins of optical flow orientation quantization. Figure 3 shows different pyramid levels. In this example, a parade is passing right while a car is passing left. In L = 1, HOG features are extracted and encoded separately for vertical, horizontal and no-motion areas; and in L = 2, the same operation is done for 4 equally quantized motion bins, including down, up, right and left.

2.2.3 Audio features

1. MFCC features (MFCC)

   We use Mel Frequency Cepstral Coefficient (MFCC) as audio feature. This feature is expected to be effective for events that has distinctive audio characteristics. Besides, we also use ΔMFCC, ΔΔMFCC, Δpower, and ΔΔpower in addition to MFCC. The total dimension is 38 and PCA is applied without dimension reduction.

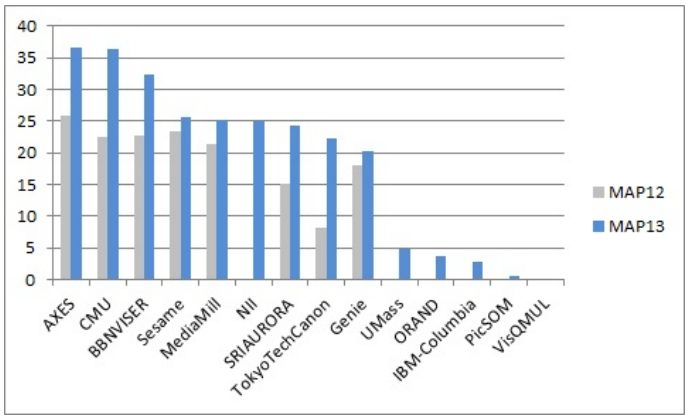

Figure 5: The comparison of MAP between 2012 and 2013 for Ad-Hoc task.

Year	Positives	Negatives
MED12	3,936	41,959
MED13	2,000	4,992

Table 2: Number of positive and negative clips used during training in MED12 and MED13.

2.3 Event detection using GMM supervectors and SVMs (GS-SVM)

We model low level features by GMM and detect events by SVMs as in the previous years. This combination has been proved effective in Semantic Indexing and Multimedia Event Detection [7, 24, 28]. For each kind of low level features (2.2), we model the features in one clip by Gaussian Mixture Model (GMM) and construct a GMM supervector using the estimated GMM parameters to form a representation of the video. The numbers of Gaussian mixtures we used are 512 for SIFT, HOG, RGB-SIFT, MFCC, and 1024 for STIP, CC-DSTIP. We train event models by support vector machines (SVMs) with the supervectors as input. Multiple features are fused in a late fusion manner, and the fusion weights are determined by two-fold cross validation during training.

2.4 Results

We submitted one run for both Pre-Specified task and Ad-Hoc task using the following condition: the system type FullSys, Event Kits 100Ex and evaluation search video set PROGAll. The MAPs are 0.245 and 0.223 respectively.

Effectiveness of velocity pyramid

We would like to know how much the newly added feature, velocity pyramid, contributes to our system. To evaluate its effectiveness, we conducted experiments on the MED11 dataset. The results are listed in Table 1. Velocity pyramid outperforms the original HOG features by 4.4 point in terms of MAP, and when combined with spatial pyramid, it has an extra gain of 1.4 point. The velocity pyramid is especially effective for such events as *Changing_a_vehicle_tire* and *Repairing_an_appliance*, in which objects move largely in the spatio-temporal space.

When integrated into the whole system, the velocity pyramid increases MAP from 0.379 to 0.391 on the MED11 dataset.

2.5 Analysis

Figures 4 and 5 compare MAP of 2012 and that of 2013 for all teams who submitted the same condition as ours. For Pre-Specified task, our MAP drops from 31.7 (2012) to 24.5 (2013). This may be due to the decrease in the size of training data.

The numbers of positive and negative samples are both reduced this year. For Pre-Specified task, since we use the same events and test dataset as last year, and use almost the same set of features except

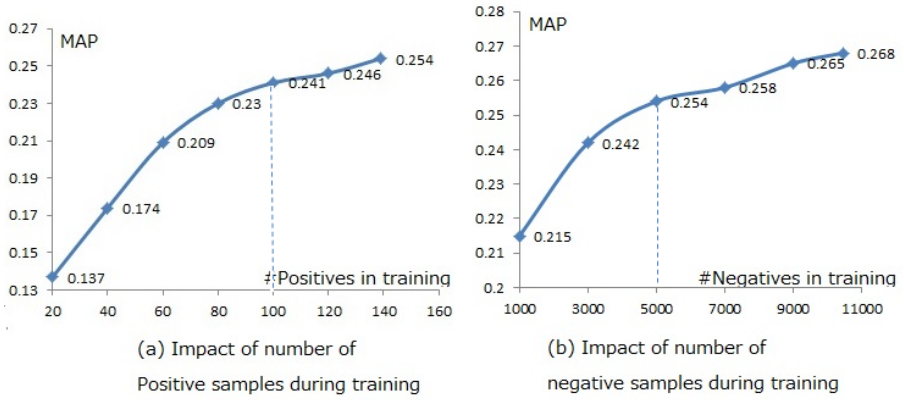

Figure 6: Influence on MAP of positive and negative samples used during training. We only use a single feature (spatial pyramid of HOG) on the MED11 dataset in the experiment. In the evaluation of positives, the number of negative clips is fixed to 5000; in the evaluation of negatives, the number of positives is fixed to the max available in event kit (#130 every event on average).

velocity pyramid for the event detection system, the only difference becomes the size of event kit and event background during training. Table 2 shows the difference.

In order to evaluate how the number of positives and negatives affects the result, we did experiments on MED11TEST dataset by changing the two conditions. The way MAP scores change subjected to the number of positive and negative examples is shown as Figure 6. It can be seen from the figure that the smaller the amount of positive or negative examples, the worse the performance will be. However, since the composition of PROGTEST is unknown, it's difficult to evaluate exactly how much these two conditions affect the final result.

2.6 Conclusion

We applied GMM supervectors and SVMs to this year's MED task. We also introduce a new feature, the velocity pyramid, and showed its effectiveness on the MED dataset. Even though we tried to find the reason for the degradation of MAP this year, there is no closed conclusion. The future work includes introducing more effective motion features.

3 Instance Search

3.1 Introduction

We propose a system that takes advantage of state-of-art feature detectors and descriptors, as well as BoW image representation, to partially cope with the difficulties due to variant appearances of an instance in different video shots. To improve the accuracy, spatial information is exploited after the initial retrieval by using RANSAC[29][30]. The technique of inverted-file indexing is also applied to make it possible to retrieve the expected video shots with no delay.

3.2 Approach

The system can be separated into two parts: the off-line processing part and the on-line query part, as shown in **Fig.7**. The video shots in the data set are processed and indexed in advance, so that in the on-line query stage the system just needs to process the query images, and is able to retrieve the video shots fast by using the pre-computed indices, making the system real-time applicable.

For the off-line part, we first try to represent each video shot with its frame(s). And then the state-of-art feature detectors and extractors are applied to the frame(s) to get some invariant descriptors for the salient regions detected. After that, a large part of descriptors are sampled and get quantized into a visual vocabulary. With that vocabulary, each video shot is capable of being represented by the visual words discovered in its corresponding sampled frame(s), thus we are able to construct a histogram counting the occurrences of the visual words for each video shot, i.e. the BoW representation. The BoW of the video

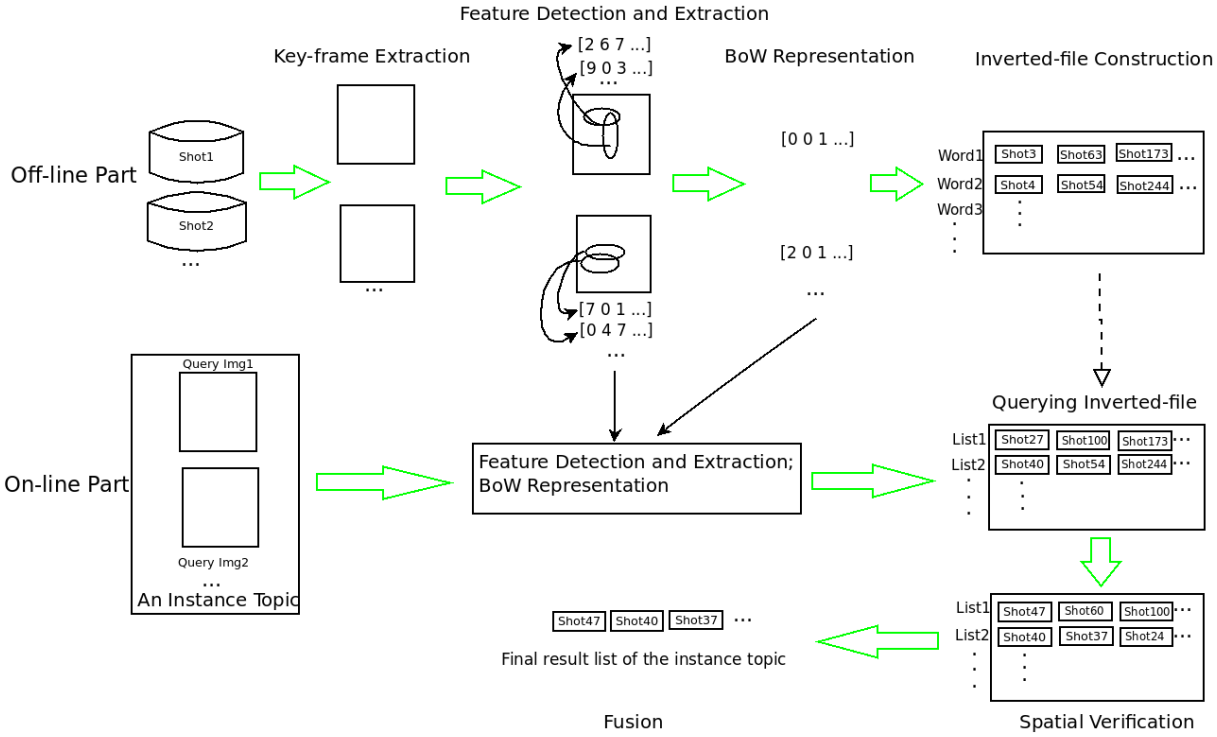

Figure 7: The Overview of the Instance Search System.

shot is generally quite sparse because the size of the visual vocabulary is often set to be large. We can utilize this property to construct the so-called inverted-file structure to index all the video shots in the data set, to make the representation of the total data set more compressed while enabling fast retrieval in the on-line query stage at the same time.

Regarding the on-line query part, the same procedures of feature detection and descriptor extraction are applied to the query images. Since we are only interested in the region denoted by the mask that corresponds to each query image, we neglect the feature points outside the mask. In the same manner, we construct a BoW for each query image and use it to retrieve the relevant video shots that are pre-indexed in the off-line stage. In order to improve the precision, spatial verification such as RANSAC[29] follows. Finally, as each instance topic is generally provided with several query images, we fuse the retrieval result lists of all the query images into one for the instance topic they belong to.

3.2.1 Key-frame Sampling

In order to reduce the computational burden, we sample some frames among each video clip for its representation. Because what we are facing are the shots that consist of continuous scenes in very short time, it's reasonable to extract the middle frame only as a key-frame. Only-middle-frame extraction keeps the processing overhead minimum while resulting in the most concise frame-based representation for video shots.

3.2.2 Feature Detection and Descriptor Extraction

For the frames extracted, we attempt to detect some affine covariant regions and represent the regions with invariant descriptors. These invariant descriptors can help us to match some visually similar patches that may be under different affinities and illumination conditions. According to [31], MSER detector outperforms the other affine region detectors in most of cases, thus it is used in our system. The MSER detector outputs ellipses to represent the affine covariant regions.

As for descriptor extraction, the SIFT proposed by Lowe in [9] is a good choice, as it's invariant against image changing in many aspects(scaling, rotation, illumination and etc.) and has been proved to be superior to others under a number of measurements. We take the center of each ellipse region detected by MSER and apply the SIFT extractor on it to construct a 128-dim vector. Different SIFT

descriptors represent different visual-variant image patches. Therefore, we are able to find out visually similar patches by measuring the distance of the descriptors.

3.2.3 Visual Vocabulary Building

In order to avoid the heavy computation of matching between descriptors of query images and all the descriptors of the data set when performing query, the descriptors should get quantized. We sample 20% of the key-frames from the data set, approximately amounting to 100K images, and quantize the descriptors extracted from them so that the descriptors wind up grouped together as clusters. The centers of the clusters would be the visual words composing the vocabulary.

The number of clusters, k, is a critical parameter. If the k is too small, the vocabulary is not sufficiently discriminant since many descriptors share the same word, resulting in visually dissimilar image patches being viewed as the same. On the other hand, when the k is set too large, some words in the vocabulary are not capable to cover all the noisy versions of the same image patch. This would usually present low recall of the retrieval even though the precision is relatively high. We have tried sizes of vocabulary of 1K, 4K, 50K, 200K and find the 50K gives the best result among the others. As for the distance measurement between descriptors, we employ Euclidean distance.

Notice that we try to cluster a rather big set of high dimensional descriptors into a big vocabulary. The well-known flatten k-means algorithm is not scalable to such big data. Hence, we apply the hierarchical k-means algorithm[32][30] instead, which has been shown much more efficient. The hierarchical k-means algorithm produces a clustering tree. It firstly partitions the descriptor space into b cells by k-means algorithm with cluster number of b(We call b the branch factor). And then for each time, the cells generated from the last iteration would be further partitioned into sub-cells using the same branch factor b. With the recursion proceeding, we end up to get the leaves of the clustering tree, namely the centroids of the lowest level clusters, as the words in the vocabulary. Empirically, we set the branch factor b to be 32.

With the construction of the vocabulary, it's able to represent a long descriptor with just a word index by assigning the descriptor to its nearest word. Thus, the key-frames could be viewed as documents with different words occurring. In the following parts, we define the similarity measurement between the "documents" and derive a method to index the "documents" for fast search.

3.2.4 BoW Representation

We use the BoW(Bag of Words) strategy to represent the video shots. BoW is actually a histogram counting the occurrences of the words in an image. Such histograms have been proven to be effective in capturing the contents of images in many applications. With BoW representation, we convert a video shot into a vector with dimension equal to the size of the visual vocabulary. To rank the video shots in the data set, it's indispensable to define a similarity measurement and calculate how close two vectors are. Here we utilize the cosine similarity defined as **Eq.(16)**.

$$sim(v_1, v_2) = \frac{v_1.v_2}{||v_1||||v_2||} \tag{16}$$

where v_1 and v_2 are the BoWs of two video shots respectively. We calculate the inner product of them and normalize with their L2-Norms, making the measurement range between 0 and 1. Cosine similarity considers only the ratio of different words. Therefore as long as the ratios of words of two BoWs are the same, they can get the largest similarity score.

Instead of using the raw histograms as described above directly, we incorporate the TF-IDF weighting scheme into the BoWs [33]. TF-IDF(term frequency and inverse document frequency) strategy is quite useful for text retrieval applications, as well as for image ones. TF(term frequency) is defined as the ratio of a specific word with respect to the total words in the document and IDF refers to taking the logarithm of ratio that the total number of documents in the data set is to the number of documents with a specific word occurring. TF-IDF is the production of these two terms. **Eq.(17)** shows the mathematical definition, in which $n_{i,j}$ is the number of word i in the document j, n_j is the total number of words in document j, N is the total number of documents in the data set and N_i is the number of documents that contain the word i. Intuitively, the TF weights words that appear many times in that document more heavily. It helps to describe the characteristics of that document better. Conversely, the IDF gives less weight to the word if it's contained frequently in many documents. In fact, the words are more discriminant if they rarely appear in the data set and thus worth more attention. TF-IDF gives the possibility to measure the importance of different words during similarity calculation.

$$TF\text{-}IDF(i,j) = \frac{n_{i,j}}{n_j} . \log \frac{N}{N_i} \tag{17}$$

Notice that BoW is essentially the term frequency. To convert the BoW to TF-IDF, we simply count the occurrences of the words in the data set and calculate the IDF for each word. Before calculating the similarity between two BoW vectors, we multiply each word in each vector with the corresponding IDF, resulting in the expected TF-IDF representation.

3.2.5 Inverted-file Indexing

Because the vocabulary size is usually large, the BoW for each video shot is generally sparse. If we store the BoWs for all video shots in the data set, it's no doubt that it requires a large amount of disk space. Also, comparing the query BoW with the ones in the data set one by one under high dimensional space is not practical for real application. The inverted-file indexing technique is well suited for such a situation[33]. In the inverted-file structure, the video shots are indexed by the words they contain. For example, if we have a vocabulary with size of k, then we construct k indices, each of which corresponds to one word. For a video shot containing a certain word, we assign it to the index of that word and record the count or frequency of that word. Therefore, a video shot would be assigned to multiple indices. Since, as we mention, the BoW is quite sparse, in this way it effectively condenses the required storing space.

In addition, inverted-file helps to accelerate the retrieval. Because we are only interested in the video shots containing the certain query words, we can just examine the ones indexed by these words. The satisfied video shots are a very small fraction of the whole data set. In this sense, it achieves a large speed-up by avoiding calculating the similarities of all the pairs between query BoW and the ones in the data set.

Specifically, in order to calculate the cosine similarities between the query BoW and the ones in the data set, we merely need to perform a one-pass scan across the words to attain the inner products and the normalization factors by traversing the records indexed by the related words. We sort the retrieved candidate video shots descending according to the similarity values they get.

3.2.6 Spatial Verification

Though using the above techniques can effectively retrieve video shots that probably contain the instance topic, the order of them is not precise enough. One reason is that the spatial information of the key-frames has been abandoned in the BoW representation, which is critical in image and video applications. We find that some of the words have been mismatched between the query image and the data set images, caused by the ambiguity of the words. Checking the spatial layout of the words is undoubtedly beneficial for discovering and eliminating the mismatched word pairs.

For that purpose, we apply the RANSAC spatial verification[29]. For each pair of query image and key-frame extracted from a candidate video shot, we choose four pairs of descriptor matches and calculate an affine transformation between the image planes. Then we check how many other pairs of descriptor matches are consistent with that affine transformation, that is, to project the query image feature points to the key-frame image plane with that affine transformation and see whether the projected points are lying near the positions where the corresponding descriptor matching points are. Taking into account the noise, we view a descriptor match as consistent, as long as the projected point to the key-frame image plane is within 3 pixels around the corresponding descriptor matching point.

We find out the most possible affine transformation by considering all the four-pair combinations of the descriptor matches between the query image and the key-frame and count the number of consistent matches with that affine transformation. For each consistent matches, we add n points to the original similarity value to make a new score. Because the spatial verification is more persuasive than the similarity value, we set the n quite large so that $n = 100$.

As the spatial verification is computationally heavy, we don't apply it to all the candidate key-frames. Instead, we just examine the first few hundred of the key-frames in the sorted list of each query image. The ones which are not examined would retain its similarity value as its score. Making the trade-off between precision and computational time, we determine the number of key-frames being examined in each list to be 300.

Run ID	Mean-AP
TokyoTech1_1	0.0053
TokyoTech2_2	0.0057

Table 3: Mean-APs of our two runs across 30 topics.

3.2.7 Fusion Scheme for Topic

By taking the steps described above, we get a list for each query image of an instance topic. A fusion scheme is necessary to combine the lists of the same instance topic to get a final list for that topic. We try to give different weights to different lists according to some criteria and re-tune the scores of video shots dependently, before combining them together. For example, if list A is given weight a and list B is given weight b, then we multiply the scores of the video shots in list A with weight a while doing the same thing to list B with weight b. After that, we put the elements of all the lists into one and sort them by their scores.

In our two runs of TRECVID task, we employ two different criteria to two runs respectively. The criteria will be discussed in **Section3.3**.

3.3 Experiment and Results

We submit two runs to the TRECVID, with two different fusion schemes applied respectively. For the first run(TokyoTech1_1), we simply put the retrieval result lists of all the query images within one instance topic together and sort the video shots based on their scores. That's equivalent to weighting all the lists with 1. For the second run(TokyoTech2_2), we count the number of feature points found in each query image and weight its corresponding retrieval list accordingly. The weighting scheme is shown as **Eq.(18)**, in which n_i is the number of feature points in the *ith* query image that is contained in the topic T. It calculates the portion of feature points of the *ith* query image among all that belong to the same topic.

$$Weight(i) = \frac{n_i}{\sum_{j \in T} n_j} \tag{18}$$

We take such weighting strategy because we think that larger number of feature points generally indicates the image is more reliable for query. **Table3** shows the mAPs of our two runs across the 30 topics. TokyoTech2_2 is better than TokyoTech1_1, though the improvement may be too small to support the effectiveness of the weighting scheme of **Eq.(18)**.

Compared with the other runs submitted by the other teams, our results are not competitive(**Fig.8**). Among 65 runs, our two runs rank 54 and 55 respectively. Therefore, each part of our pipeline is necessary to be refined to reach the best performance. We would look over the future work in **Section3.4**.

3.4 Conclusion and Future Work

We propose a pipeline for the instance task in this note. The pipeline consists of an off-line part that pre-indexes the video shots of the large data set, as well as an on-line part that incorporates some helpful techniques which make big improvement on retrieval results. However, our method still doesn't perform well compared to the other teams' ones. We should come up with some new ideas to overcome the defects of our system.

In the future, we will focus on the building and usage of the visual vocabulary, to make the vocabulary more discriminant and precise among the different words. Also, we are about to improve the representation of each key-frame, attempting to incorporate some geometric information into its vector representation. Finally, we think the utilization of more feature detectors and descriptors may be beneficial for our work, since it has been discovered that some feature detectors and descriptors are complementary[31]. The combination of distinct detectors and extractors is also a critical issue that should be thought over.

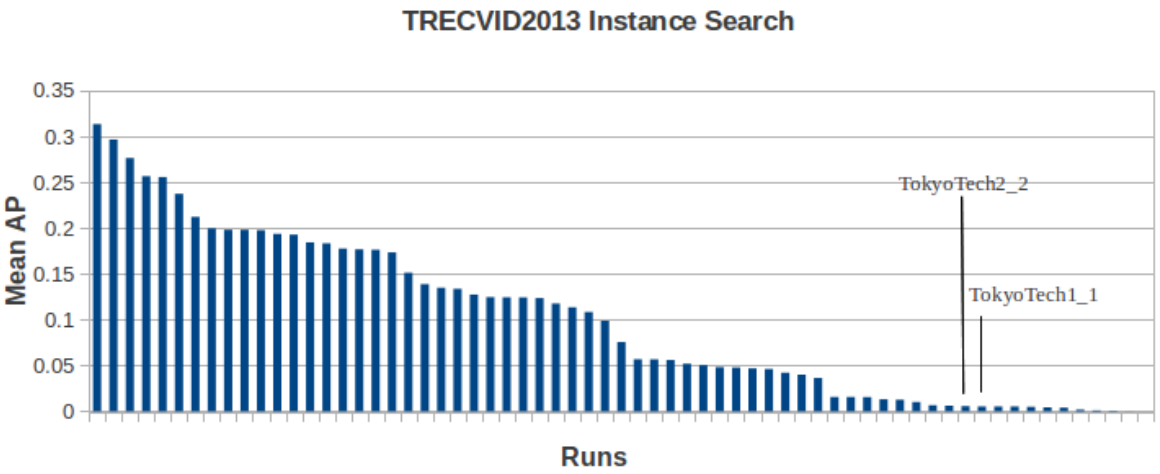

Figure 8: Overview of results of the instance search task in TRECIVD 2013

References

[1] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. F. Smeaton, and G. Quéenot. TRECVID 2013 – An Overview of the Goals, Tasks, Data, Evaluation Mechanisms and Metrics. In Proc. of *TRECVID workshop*, 2013.

[2] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVid. In Proc. of *ACM MIR workshop*, pp.321–330, 2006.

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

[4] S. Ayache, and G. Quéenot. Video Corpus Annotation using Active Learning. In Proc. of *ECIR*, pp.187–198, 2008.

[5] S. Strassel, A. Morris, J. Fiscus, C. Caruso, H. Lee, P. Over, J. Fiumara, B. Shaw, B. Antonishek, and M. Michel. Creating HAVIC: Heterogeneous Audio Visual Internet Collection. In Proc. of *LREC*, 2012.

[6] N. Inoue, and K. Shinoda. A Fast and Accurate Video Semantic-Indexing System Using Fast MAP Adaptation and GMM Supervectors. In IEEE trans. on Multimedia, vol.14, no.4, pages 1196–1205, 2012.

[7] N. Inoue, and K. Shinoda. A Fast MAP Adaptation Technique for GMM-supervector-based Video Semantic Indexing Systems. In Proc. of *ACM Multimedia* (short paper), 2011.

[8] N. Inoue, and et al. High-Level Feature Extraction using SIFT GMMs and Audio Models. In Proc. of *ICPR*, 2010.

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *In IJCV*, 2004.

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

[11] J. van de Weijer and C. Schmid. Coloring local feature extraction. In Proc. of *ECCV*, vol.2, pages 334–348, 2006.

[12] X. Wang, T. X. Han, and S.Yan. An HOG-LBP Human Detector with Partial Occlusion Handling. In Proc. of *ICCV*, pages 32–39, 2009.

[13] B. Safadi and G. Qunot. Re-ranking by Local Re-scoring for Video Indexing and Retrieval. In Proc. of *CIKM*, 2011.

[14] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms, http://www.vlfeat.org/, 2008,

[15] 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.

[16] M. Hradis, M. Kolar, J. Kral, A. Lanik, P. Zemcik, and P. Smrz. Annotating Images with Suggestions - User Study of a Tagging System. In Proc. of *ACIVS*, 2012.

[17] T.Ojala, M.Pietikainen and D.Harwood. Acomparative study of texture measures with classification based on feature distribution. *Pattern Recognition*, vol. 29, no. 1, pp. 51-59 1996.

[18] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. CVPR*, vol. 1, 2001.

[19] M. Isard and A. Blake. Condensation: Unifying low-level and high-level tracking in stochastic framework. *Proc. ECCV*, 1998

[20] B. K. P. Horn and B. G. Schunck, Determining optical flow. *Artifical Intelligence*, vol. 17, no. 1, pp. 185-203, 1981.

[21] N. Dalal and B. Triggs.B. Histograms of oriented gradients for human detection. In Proc. of *CVPR*, 2005.

[22] I. Laptev. On space-time interest points. In *IJCV*, vol.64(2), pp.107-127, 2005.

[23] S. Lazebnik, C. Schmid and J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories. In Proc. of *CVPR*, 2006.

[24] N. Inoue, Y. Kamishima, T. Wada, K. Shinoda and S. Sato, TokyoTech+Canon at TRECVID 2011. TREC Video Retrieval Evaluation (TRECVID) workshop, Dec.5, 2011.

[25] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld, Learning Realistic Human Actions from Movies. In Proc. of *CVPR*, pp. 1–8, 2008.

[26] H. Wang, M. M. Ullah, A. Klšer, I. Laptev, and C. Schmid, An Evaluation of local spatio-temporal features for action recognition. In Proc. of *BMVC*, 2009

[27] Y. Kamishima, N. Inoue, K. Shinoda, Event detection in consumer videos using GMM supervectors and SVMs. EURASIP Journal on Image and Video Processing. Vol. 2013, pp. 1-13. Sep 2013

[28] N. Inoue, Y. Kamishima, K. Mori and K. Shinoda, TokyoTechCanon at TRECVID 2012. TRECVID 2012, Nov, 2012

[29] O. Chum, J. Matas, and S. Obdr alek, Enhancing RANSAC by generalized model optimization. In Proc. *ACCV*, 2004.

[30] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman, Object retrieval with large vocabularies and fast spatial matching. In Proc. of *CVPR*, 2007.

[31] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, L. Ban Gool, A Comparison of Affine Region Detectors. International Journal of Computer Vision, 2005.

[32] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu, An Efficient k-Means Clustering Algorithm: Analysis and Implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.

[33] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos. In Proc. of *CVPR*, 2003.