Tokyo Tech at TRECVID 2008

Shanshan Hao, Yusuke Yoshizawa, Koji Yamasaki, Koichi Shinoda, and Sadaoki Furui
Department of Computer Science, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552 Japan
{shan,yoshizawa,yamasaki}@ks.cs.titech.ac.jp, {shinoda,furui}@cs.titech.ac.jp

Abstract

The Tokyo Institute of Technology team participated in the high-level feature extraction, surveillance event detection pilot and Rushes summarization tasks for TRECVID2008. In the high-level feature (HLF) extraction task, we employed a framework using a tree-structured codebook and a node selection technique last year. This year we focused on the position information of each object-related HLF. During the training phase, we applied our method not on the whole key-frame images, but on the regions in the image which contain the annotated HLF only. From the evaluation of TRECVID2008, the inferred average precisions of the three runs are all 0.011. The method we improved this year doesn't contribute to a better performance. In surveillance event detection pilot task we use optical flow features and an SVM (Support Vector Machine) to detect each surveillance event. We present our preliminary experimental results in this paper. In the rushes summarization task, we estimated the number of scenes for a summary using minimum description length. We use two low-level features, the YCbCr color histogram and optical flow.

1. High-Level Feature extraction

In the high-level feature (HLF) extraction task in TRECVID2007, we proposed a novel method, which used a tree-structured codebook and a node selection technique for HLF extraction from video data. In this method, we first construct a tree-structured codebook shared among all the HLFs by clustering SIFT descriptors [3]. Then, we select nodes to be used as visual words for each HLF. Since motion information is important for some HLFs with movement, we also employed motion words which are defined as visual words with motion activities.

Since there are often many uninterested parts in a key-frame image in which the HLFs are annotated to exist, extracting features from the whole key-frame image rather than from the exact areas where we are interested may bring a lot of noise. In order to reduce the noise impact to our system, in this year's task, we improved this framework by taking into consideration location information of each HLF.

In the annotation data, which is annotated and provided by MCG-ICT-CAS [4], the 20 HLFs of this year's task are divided into two groups: object-related HLFs and scene-related HLFs. The object-related HLFs can be located in rectangle regions, for example, bridge, dog, and two people. In contrast, the scene-related HLFs can not be located in exact regions, for example, classroom, kitchen and cityscape. In our experiment, for the object-related HLFs, we focus on the annotated regions in the key-frame images. For the scene-related HLFs which have no region annotations, we use the whole key-frame images as the location of these HLFs.

The remainder of this section is organized as follows: Subsection 1.1-1.3 will introduce our system framework and method we proposed last year. Subsection 1.5 will describe the location information used by our method. Subsection 1.6 will report our experimental results. Subsection 1.7 concludes our work of HLF extraction task.

1.1 High-level feature extraction system

Figure 1 shows our HLF extraction system. We first select key-frame images from videos. Then we extract Harris-Affine regions [5] from each key-frame image we have selected and then describe each region with a 128 dimensional SIFT descriptor (4x4-grid, 8- orientation).

We use Region Descriptor software [9] provided by Visual Geometry Group [10] with its default parameters to extract the affine-invariant regions. These SIFT descriptors are then quantized with a binary tree cluster codebook which is constructed in advance for all the HLFs from training data.

For each HLF, we select nodes from the tree-codebook and use the selected nodes as visual words. In addition, we also extract motion words which are defined as visual words with motion activities. Then for each key-frame, we count the occurrences of each visual words and motion words to build a feature vector in which each element represents the number of occurrence of each word.

At the same time, we also construct a feature vector using Hessian-Affine regions, and concatenate the aforementioned feature vector with the feature vector constructed using Harris-Affine regions for a classifier. With these feature vectors, we use a maximum entropy model (MEM) [6] to classify the presence/absence of each high-level feature. A MEM estimates the posterior distribution of a label (presence or absence) given to the features of a key-frame image. We use the implementation of MALLET [7] in our system.

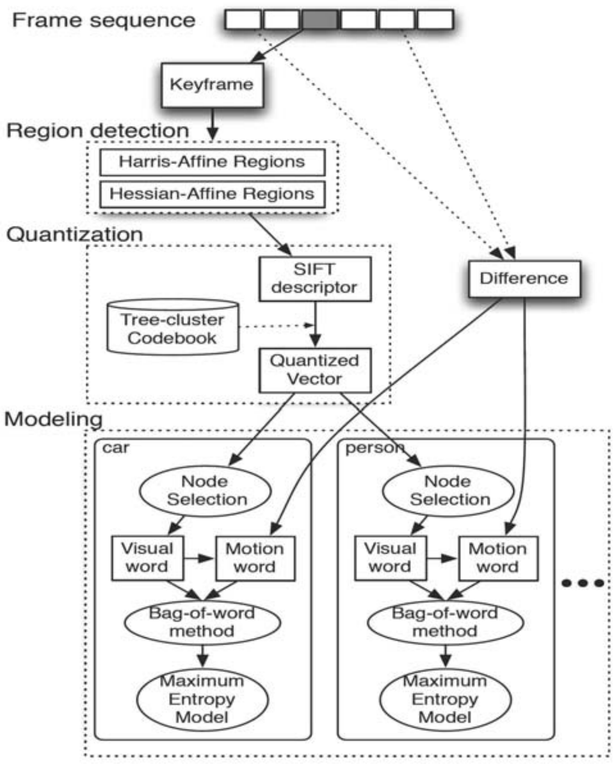

Figure 1. High-level feature extraction system

1.2 Visual word

We use a 128 dimensional SIFT descriptor to describe each region we extracted from a key-frame image in the training dataset. Then we construct a tree-structured codebook which is shared among all the HLFs by recursively clustering all the SIFT descriptors in the training data. Finally, we select nodes from the tree-structured codebook and use them as visual words for each HLF.

We will explain the construction process of tree-structured codebook and the method of node selection in detail.

1.2.1 Tree-structured codebook

We use the *K*-means clustering method to construct our binary tree cluster codebook. The Euclidean distance between vectors is used as the distance measure in the clustering process.

First, we place all the SIFT descriptors to a root node, and then we divide these descriptors into two clusters, which are defined as the child nodes of the root node. We continue dividing each child cluster into two clusters recursively until the number of all the clusters arrives to a predetermined threshold S_a. Finally, in each node we replace all the allocated descriptors with the mean of these descriptors. After this replacing step we get a binary tree structure.

1.2.2 Node selection

For each HLF, we compare descriptors of each key-frame where this HLF is present with all the nodes of the tree-structured code book. We select all the nodes that conform to certain rules which we will explain later.

The process of node selection and rules are described in detail as follows.

(0) Suppose D_i is a set of SIFT descriptors extracted from the key-frames where HLF_i is present，T is the tree structured codebook.

(1) Each SIFT descriptor of D_i is quantized using the leaf node set of T, and we set the number of occurrences to each node.

(2) Starting from leaf nodes, we add up the number of occurrences in the leaf nodes to their corresponding parent node.

(3) Starting from each of the leaf nodes, we select nodes whose occurrence number is over a predetermined threshold S.

We defined the set of selected nodes as the dictionary for HLF_i, and the number of selected nodes as the size of this dictionary. Figure 2 shows an example of node selection with different threshold S.

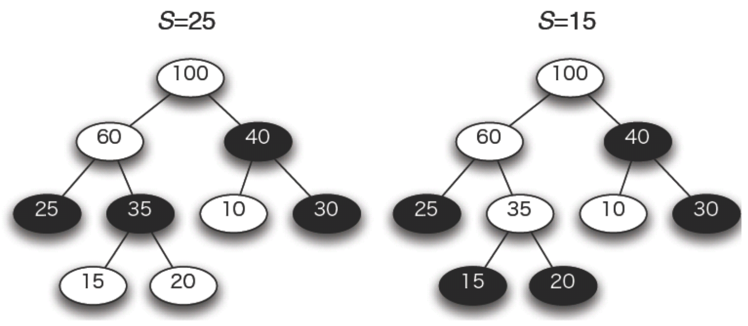

Figure 2. Examples of the node selection with different threshold *S* for HLF_i.
The number inside a node represents the occurrence number of the HLFi.
Black nodes represent the nodes that are selected.

1.3 Motion word

Motion information was very helpful for some features such as airplane_flying, bus and emergency_vehicle in our previous experiment. We pay close attention to the regions where the motion is active. The visual word of these regions is chosen as the motion word of these regions.
We determine a visual word to be moving or not according to the following process. First, the differences between two neighboring frames are calculated. Then the motion activity value of each region is obtained by taking the average of these differences. Finally, the motion activity value is compared to a predetermined threshold T_m to determine whether the motion of the region is active or not. The motion activity information we get and the region's visual word are combined to form a "motion word".

1.4 Modeling

For each key-frame, we build a feature vector in which each element represents the occurrence number of visual words and motion words. With these feature vectors, we use a maximum entropy model (MEM) to classify the presence/absence of each high-level feature. A MEM estimates the posterior distribution of a label (presence or absence) given to the features of a key-frame image:

$$P(y \mid x) = \frac{1}{Z(x)} \exp \sum_i \lambda_i f_i(x, y) \qquad (1) \ ,$$

where x is a feature vector, y is a binary variable representing presence or absence of the HLF, $Z(x)$ is a partitioning function, λ_i is a model parameter, and $f_i(x, y)$ is a feature function. We used the following feature function:

$$f_{i,y'}(x, y) = \begin{pmatrix} x_i, & \mathit{if} \ \ y = y' \\ 0, & \mathit{otherwise} \end{pmatrix} .$$

We use the Limited Memory BFGS method to estimate the parameter set $\{\lambda_i\}$.

1.5 Region information

Since there are many uninteresting areas in the key-frame images in which the HLFs are annotated to exist, extracting features from the whole key-frame images rather than from the exact areas where we are interested may bring a lot of noise. In order to reduce the noise impact on our system, we focus on the exact location areas of each HLF, rather than the whole key-frame images.

This year we used the location information of each object-related HLF which is defined by MCG-ICT-CAS [4]. The local HLF can be located by a rectangle whose coordinates are recorded and provided by MCG-ICT-CAS. We use this information in our experiment. When the region selection explained in Subsection 1.1 is performed, it is reasonable to restrict the process inside the rectangle only. We focus on these particular areas when we extract the visual words of 14 object-related HLFs (shown in Table 1). For the other six scene-related HLFs (shown in Table 1.), since no location information is available, we don't have the region restrictions. Figure 3 shows an example of location information for one HLF: hand.

We employ this location information to our system in order to examine whether it contributes to our system.

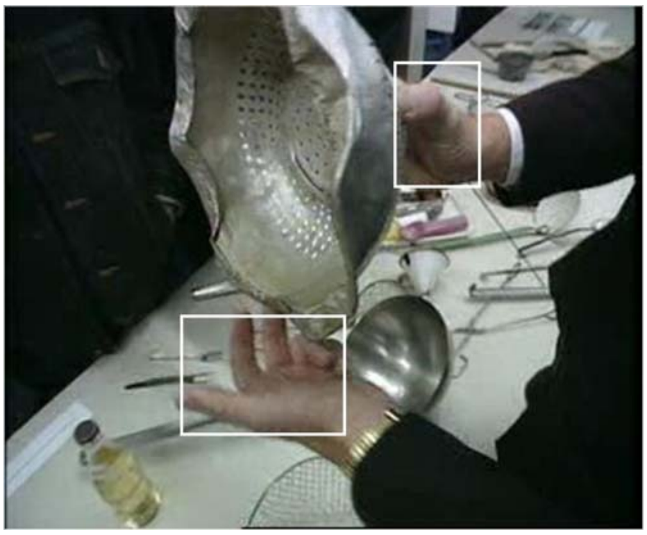

Figure 3. Example of the local information for HLF "hand".
The rectangle regions are the target areas.

1.6 Experiments

1.6.1 Experimental conditions

Our system trained on the common TRECVID2008 development collection data. We use the video information only, neglecting the audio and text information. We detect 20 HLFs (Table 1) for our evaluation with the TRECVID2008 test collection data as test data. We also use the key-frames and annotation information provided by the dataset and the location information of each of the object-related HLFs provided by MCG-ICT-CAS. Object-related HLFs are the 14 HLFs which can locate the target HLF in a rectangle range by MCG-ICT-CAS. Scene-related HLFs are the 6 HLFs which are concerned with the whole key-frame images as an object by MCG-ICT-CAS.

Considering the choice of different features, we perform three runs for three types of features, respectively:

- A_TitechV_3: We used visual words only as features and did not use the location information.
- A_TitechM_2: We used visual words and motion words as features and did not use the location information.
- A_TitechPVC_1: We used visual words only and used the location information.

The number of key-frames and the number of detected regions are shown in Table 2. From the table, we find that, although we only extract SIFT in some areas rather than in the whole images, the number of regions is still quite large.

Object-related HLFs					Scene-related HLFs	
Bridge	Emergency_vehicle	Two people	Dog	Flower	Classroom	Kitchen
Driver	Demonstration_or_protest	Telephone	Bus	Singing	Nighttime	Harbor
Montain	Airplane_flying	Boat_ship	Hand		Cityscape	Street

Table 1. The list of HLFs used in our experiment

	No. of key-frame	No. of Harris-Affine	No. of Hessian-Affine
A_TitechV_3	39674	11,983,374	10,973,117
A_TitechM_2	39674	11,983,374	10,973,117
A_TitechPVC_1	39674	11,135,741	10,334,928

Table 2. Number of key-frames and regions of the development dataset. Harris-Affine is the region detected by the Harris-Affine detector, and Hessian-Affine is the region detected by the Hessian-Affine detector

In the TRECVID2008 data, we have 39674 key-frame images in the development data set, and we use random sampling to reduce the number of key-frames images to 10000. In the clustering step, we use K-means clustering method where K=2. From our previous study of last year, we set the predetermined threshold of total number of clusters S_a to be 800. We note that, using tree-clustering, we can reduce the computational cost for quantizing each region from O(N) to O(log$N$) (N: Number of clusters).

1.6.2 Experimental results

Our submitted runs are compared based on a sample of the submission pools. Precision-recall curves based on the sample will be used as well as inferred average precision [2], which provides a good estimate of average precision-a single –valued combination of precision, recall and ranking ability [1].

Figure 4 and Figure 5 show our results comparing with the results from other groups. Most of our HLF results are below the average result of all the participants except feature 11: harbor, a scene-related object.

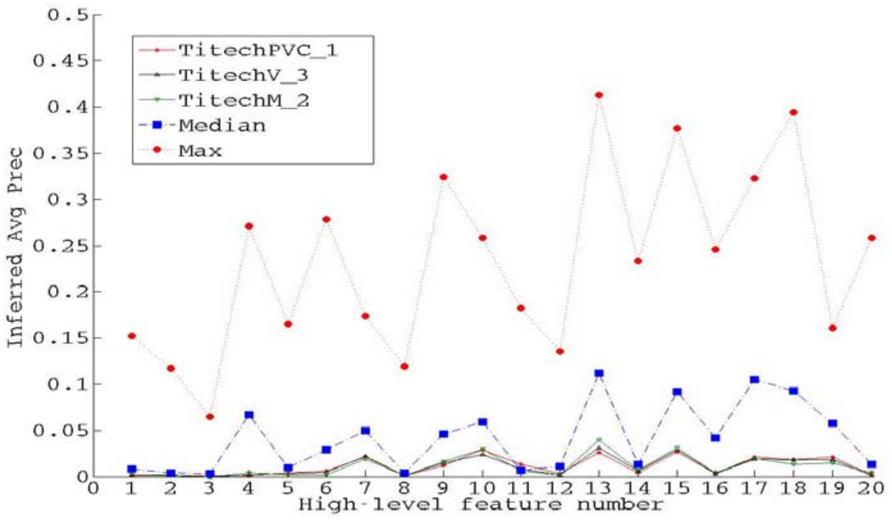

Figure 4. Inferred average precision.

It is estimated using a 50% sample. Max is the best result of the corresponding HLF.

Media is the average result of the corresponding HLF. Others are our 3 runs.

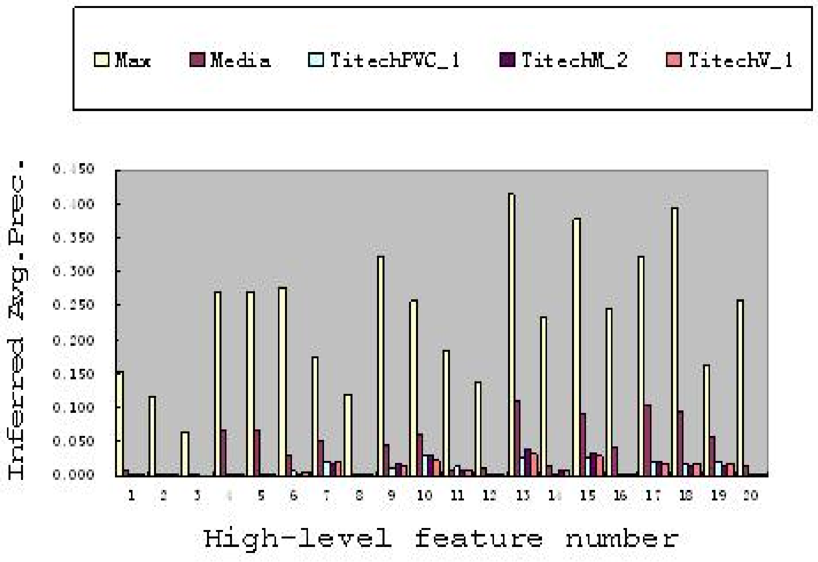

Figure 5. Inferred average precision.

It is estimated using a 50% sample. Max is the best result of the corresponding HLF.

Media is the average result of the corresponding HLF. Others are our 3 runs.

Table 3 and Figure 6 show the results of our three runs in detail.

No	Max	Media	TitechPVC_1	TitechM_2	TitechV_3
1	0.152	0.008	0.001	0.002	0.002
2	0.117	0.004	0.001	0.001	0.002
3	0.065	0.003	0.001	0.000	0.000
4	0.271	0.067	0.001	0.004	0.002
5	0.271	0.067	0.004	0.002	0.003
6	0.278	0.029	0.006	0.002	0.005
7	0.174	0.050	0.021	0.019	0.022
8	0.119	0.004	0.001	0.001	0.001
9	0.324	0.046	0.012	0.016	0.015
10	0.258	0.059	0.029	0.030	0.024
11	0.182	0.007	0.013	0.006	0.008
12	0.136	0.011	0.003	0.002	0.002
13	0.413	0.112	0.026	0.040	0.031
14	0.233	0.013	0.004	0.007	0.006
15	0.377	0.092	0.027	0.031	0.029
16	0.246	0.042	0.003	0.004	0.003
17	0.323	0.105	0.021	0.020	0.019
18	0.394	0.093	0.018	0.013	0.018
19	0.161	0.058	0.021	0.015	0.018
20	0.258	0.013	0.002	0.004	0.001

Table 3. Inferred average precision for each HLF of each run.
It is estimated using 50% sample. Estimated precision = 2×actual from sample.
No. in bold font is the scene-related HLF's number

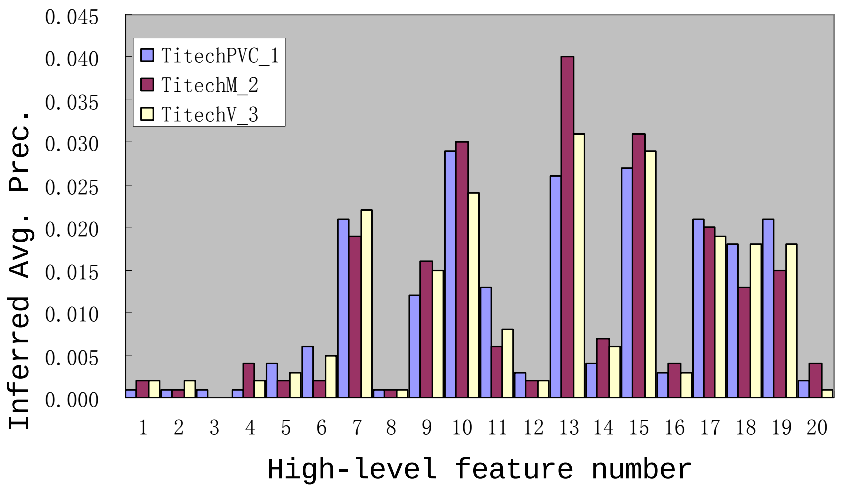

Figure 6. Inferred average precision for each HLF of each run.
It is estimated using a 50% sample.

From the figure and table we notice that there is no clear variation among our three runs' results. The mean inferred average precision of the three runs are all 0.011. In our previous experiment using the past data of TRECVID, the methods that A_TitechM_2 and A_TitechPVC_1 use have some improvements to some degree.

Comparing the result of TitechPVC_1 with the base line result of TitechV3, for the 16 object-related HLFs, five of them get better results; three of them remain the same, and the other six get worse.

The HLF numbers of the six HLFs which get worse results are 2, 4, 7, 9, 14 and 15. Most of them have some relationship with the environment outside the rectangles, such as two_people, driver, and hand. Drivers always appear with vehicles; hands always appear with human body. Maybe it's better to consider the circumstance not only the position that the target HLFs located. The HLF numbers of the five HLFs which get better results are 3, 6, 12, 19 and 20. Most of them are independent individuals, such as flower telephone and airplane. Location information seems useful for these independent individuals HLFs.

However the inferred average precisions remain the same for these two approaches. It means that our method of considering the location information for object-related HLFs did not improve the performance for all the HLFs as a result of the location information available this year.

1.7 Conclusion

Our results demonstrate that, the methods we used in A_TitechM_2 and A_TitechPVC_1 did not improve the performance all the time. The SIFT feature we extracted from the data along could not contribute to better performance on our task. It seems that other systems based on other features different from the SIFT features get better results, and it may be promising to combine the SIFT feature and other features. In the next year's task, we will consider combining other features with the SIFT feature and try to get a better result.

2. Surveillance event detection pilot

Accompanied with the increasing safety and security concerns, security surveillance systems are widely used. Continuous videos are recorded for several days, even weeks in order to be reviewed in future for some purposes. As it has become difficult for people to browse all the videos recorded from many cameras for several days, an intelligent surveillance system is strongly needed for automatically detecting suspicious events which may present a threat.

In this year's task we selected three events, person_runs, people_meet and people_split_up, out of the ten events that TRECVID2008 provided. We chose these three events for two reasons. First, these three events appear frequently in the training data set; second, these three events are related to the motion of the whole body of a person, unlike the event of taking_picture, which is only the motion of a person's hands. We use optical flow features and SVM (Support Vector Machine) in our system to detect surveillance events.

In our system, we calculate the optical flow features at first and then select the ones whose value is larger than a predetermined threshold. The selected optical flow features are clustered and the averaged optical flow of each cluster is calculated. Finally, we detect the event using SVM which has been trained using the averaged optical flow features.

2.1 Method

In our system, we have two main phases: the training phase and the detection phase.

2.1.1 Training phase

In the training phase, fist, we divide each frame of the video data into pixel blocks. We call the center pixel of each block the grid point. For each two successive video frames, we calculate an optical flow vector for each grid point we defined using OpenCV [8] (shown in Figure 7-a). The optical flow vectors whose length is out of a predetermined range R_a are excluded (shown in Figure 7-b).

Second, we cluster the optical flow features we have got. We divide the optical flow vectors into different clusters according these two rules (shown in Figure 7-c):
a) If the distance between two vectors is longer than a predetermined threshold $S_{b,}$ they should be clustered in different clusters.
b) If the number of vectors that a cluster contains is less than a predetermined threshold S_b, this cluster will be excluded.

Third, we smooth the optical flow vectors for each cluster. For each cluster, we calculate the mean of the optical flow vectors to get an averaged optical flow vector. At the same time we locate the smallest rectangle which is able to include all the grid points of the selected optical vectors. Then we replace each grid points in the rectangle with the averaged optical flow vectors (shown in Figure 7-d). The grid point which doesn't belong to any rectangular is set to zero vectors. Figure 7 shows our method of extracting feature vectors.

Finally, we train the SVM using the features we have averaged.

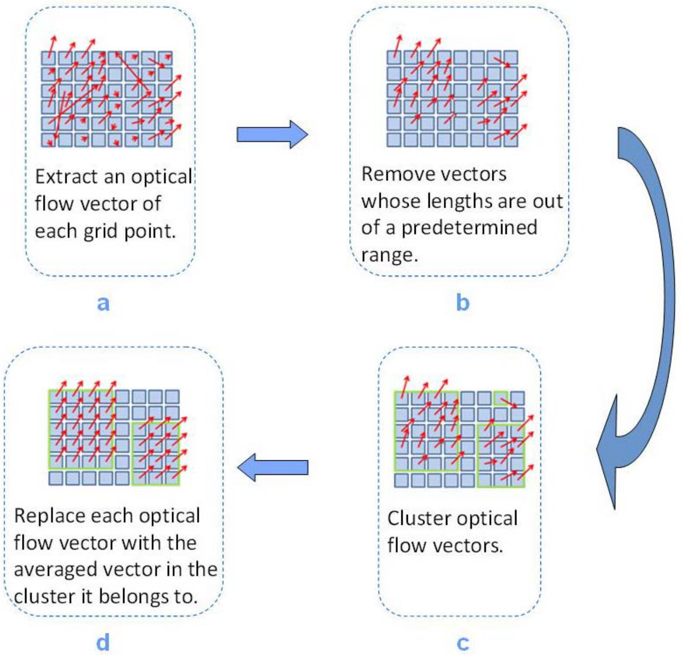
Extract an optical flow vector of each grid point.
a
Remove vectors whose lengths are out of a predetermined range.
b
Cluster optical flow vectors.
c
Replace each optical flow vector with the averaged vector in the cluster it belongs to.
d

Figure 7. Method of extracting feature vectors

2.1.2 Detection phase

In the detection phase, we do the same three steps we described in the training phase. The only difference is that we will use the SVM to detect events according to our definition that whether an event is detected. We will explain it in detail in section 2.2.1. Figure 8 shows our whole system.

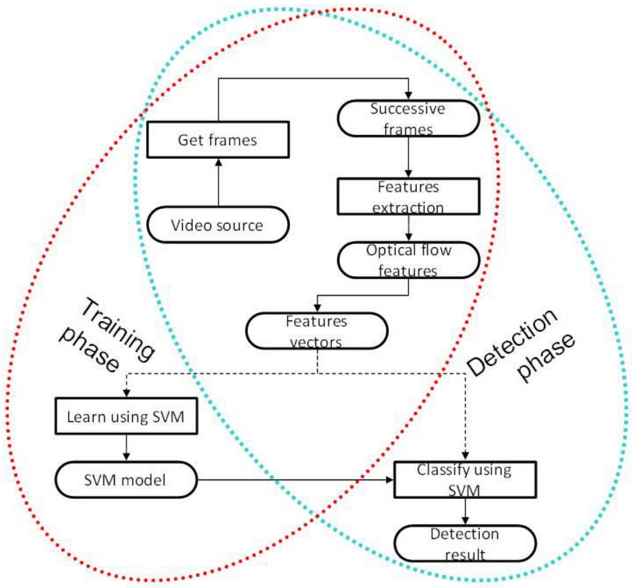

Figure 8. Surveillance event detection pilot system.
The left ellipse demonstrates the training phase.
The right ellipse demonstrates the detection phase.

2.2 Experiment

2.2.1 Experimental conditions.

In our experiment we use the development data and annotations provided by TRECVID2008. We train one SVM for each event, for each camera.

In the training phase, in order to reduce the computing time, we will not use all the frames. From the shots where the event is annotated, we randomly select 3000 pairs of frames. Each pair of frames is two successive frames (show in Figure 9). If the number of frames in one event was less than 3000 pairs, we used all the frames. We call them positive pairs of samples. At the same time, from the shots where the event is absent, we select twice as many as the positive pairs of samples which we call negative samples. We show it below:

$$x = \begin{pmatrix} 3000 & if \;\; y \geq 3000 \\ y & if \;\; y < 3000 \end{pmatrix} \qquad z = 2 \times x \qquad (2)$$

Where x is the positive pairs of samples, z is the negative samples; y is the pairs of successive two frames in one event.

Each frame of the video data is composed of 720×576 pixels. We divide each frame into 9×9 pixel blocks, and get 5120 blocks from each frame. The predetermined range R_a in our experiment is 1.5pixels to 30.0 pixels, the predetermined thresholds S_a is 3 pixels in x axis and 4 pixels in y-axis; S_b is 10.

The thresholds S_a and S_b are picked up arbitrarily. It has not been examined whether these are good thresholds or not.

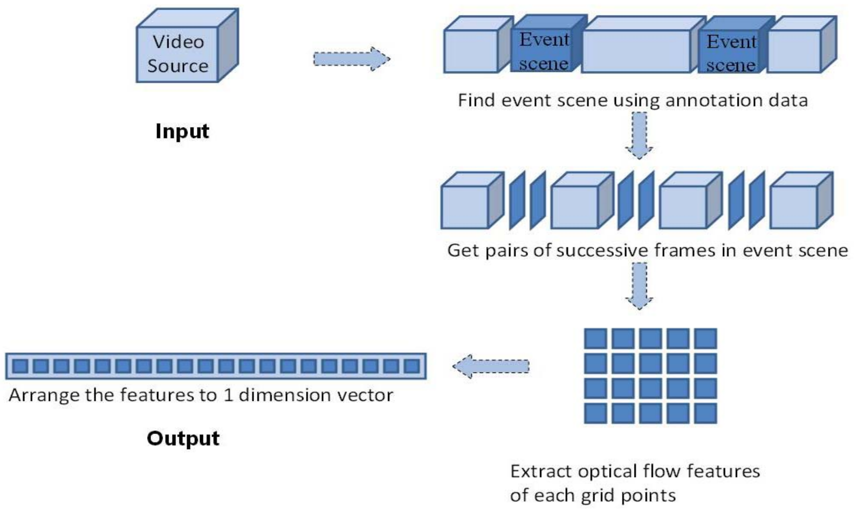
Video Source
Input
Event scene
Event scene
Find event scene using annotation data
Get pairs of successive frames in event scene
Extract optical flow features of each grid points
Arrange the features to 1 dimension vector
Output

Figure 9. Approach of preparing feature vectors for SVM

In the detection phase, we use the evaluation data of TRECVID2008's for the surveillance event detection task. We apply SVM for every 5 frames. If more than 5 detected frames are detected in any successive 25 frames we claim that the event is detected.

2.2.2 Experimental result.

We average five days data for each event, each camera. The result is evaluated by calculating precision and recall for each event, each camera. Table 4 shows our result.

Precision

	CAM 1	CAM 2	CAM 3	CAM 4	CAM 5	ALL
Runs	0.00	1.21	0.40	0.00	1.33	1.01
Meet	3.15	6.53	12.33	0.09	1.46	6.93
Split	6.73	6.29	0.00	0.48	0.61	5.19

Recall

	CAM 1	CAM 2	CAM 3	CAM 4	CAM 5	ALL
Runs	0.00	18.99	4.12	0.00	9.15	11.77
Meet	5.76	4.44	24.69	9.52	3.25	11.95
Split	3.20	11.67	0.00	6.80	0.58	4.17

Table 4 Precision and recall for the three **events and five cameras.**

2.3 Conclusion

For event detection, we chose three events, person_runs, people_meet and people_split_up. We extracted optical flow features from videos, and used SVMs as the classifiers.

3. Rushes summarization tasks

For rushes summarization task, our system uses model selection to estimate the optimal number of scenes used for a summary.

3.1 Method

First of all, each shot in a video clip is divided into segments with equal length. Our system estimates the number of scenes in two stages using Minimum Description Length (MDL) criterion as a model selection criterion. In the first stage, we estimate the number of scenes in each shot when we cluster segments into several scenes. In the second stage, we estimate the number of scenes in the full video when we bring similar scenes together. Next, we model *each shot / the full video* with a Gaussian mixture model (GMM) and assume that each mixture component corresponds to one scene. We estimate the number of mixtures using MDL criterion.

3.2 Result

Our scores evaluated in the TRECVID 2008 rushes summarization task are almost the same as the average of all the participants for the eight evaluation measures. It doesn't confirm that our system successfully estimated the appropriate number of scenes for a summary. Our system used only two low-level features YCbCr color histogram and optical flow. These two features might not capture the characteristics of scenes.

In addition, the normalization method might also be one of the causes of over-estimation. We normalized features so that the distribution of each video clip with mean 0 and variance 1, the variance between scenes is very small. Therefore, in calculation of description length, the log-likelihood became too large compared with the penalty term.

3.3 Conclusion

We proposed a video summarization system using MDL criterion to estimate the optimal number of scenes for a summary. A video clip is modeled with a GMM and we assume each component represents one scene. Finally, we estimate the optimal number of mixtures using MDL. Our system obtained scores close to the average of all the teams. Please refer to [11] for more details

References

[1] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and TRECVID. In MIR ’06, pages 321–330, 2006.

[2] Yilmaz, Emine and Aslam, Javed A. Estimating Average Precision When Judgments are Incomplete in Knowledge and Information Systems, Vol.16(2) pages 173-211, 2008.8

[3] D. Lowe. Distinctive image features from scale-invariant keypoints. In International Journal of Computer Vision, volume 20, pages 91–110, 2003.

[4] Multimedia Computing Group, Institute of Computing Technology, Chinese Academy of Sciences, Annotation of TRECVID 2008 Development Keyframes.

[5] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. In International Journal of Computer Vision, 65(1/2):43–72, 2005.

[6] A. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. In Computational Linguistics, Vol. 22(1), pages. 39-71, 1996.4

[7] McCallum, Andrew Kachites, "MALLET: A Machine Learning for Language Toolkit.", http://mallet.cs.umass.edu. 2002.

[8] Intel Open Source Computer Vision Library. http://www.intel.com/research/mrl/research/opencv/

[9] Region Descriptors http://www.robots.ox.ac.uk/~vgg/research/affine/descriptors.html#binaries

[10] Visual Geometry Group, Robotics Research Group, Department of Engineering Science, University of Oxford

[11] Koji Yamasaki, Koichi Shinoda, and Sadaoki Furui. Automatically Estimating Number of Scenes for Rushes Summarization. In TVS '08: Proceedings of the International Workshop on TRECVID Video Summarization, 2008