NHK STRL at TRECVID 2011: Surveillance Event Detection and Semantic Indexing

Masaki Takahashi[†][‡] Yoshihiko Kawai[†] Masahide Naemura[†] Mahito Fujii[†]
Shin'ichi Satoh[‡][§]

† NHK Science and Technology Research Laboratories, 1-10-11 Kinuta, Setagaya-ku, Tokyo, Japan

‡ The Graduate University for Advanced Studies, Shonan Village, Hayama, Kanagawa, Japan
[§]National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan

1 Introduction

Members of the NHK Science and Technology Research Laboratories participated in two tasks at TRECVID 2011: a surveillance event detection task and a semantic indexing task.

Surveillance event detection is the act of detecting specific human actions from surveillance videos of crowded areas such as airports. This year, we targeted "Pointing," "CellToEar," "ObjectPut," and "Embrace" actions, which our proposed system identified by using the bag-of-features model. We used a motion-appearance histogram that was calculated from a key-point trajectory as a feature. The trajectories were segmented on the basis of their detected position, and each segment functioned as a bag for the bag-of-trajectory approach. Evaluation results showed that our system has a relatively robust performance when detecting small human actions.

Semantic indexing is the task of detecting semantic concepts (such as objects or events) from a large video archive. The common method for semantic indexing is the "bag-of-visual-words" approach [1, 2] which is based on a frequency histogram of local features including SIFT [3] or SURF [4]. The effectiveness of this approach has been shown by many previous works [5]; however, the method cannot take into account the relationship between feature points. It also suffers from a loss of information when converting the feature descriptor to a visual word.

This paper proposes a novel local feature method that considers the spatial relationship between feature points and the co-occurrence frequency of feature descriptors at each feature point. The proposed method uses random forests algorithm [6] as a classification method to reduce the computational time required for training and detection. To evaluate its effectiveness, the proposed method is applied to a full task of 346 concepts.

2. Surveillance event detection

2.1 Overview

The demand for technology that can automatically identify human actions is increasing with the rapid spread of surveillance cameras. In addition to the use of surveillance, human action recognition techniques can be applied to many services, such as motion-based video searches and man-machine interfaces. In those applications, a person's full body shape can only rarely be captured from the input cameras. Therefore we targeted relatively small events that involve only part of a person's body and not motion of the entire body, such as "Pointing," "CellToEar," "ObjectPut," and "Embrace."

Key-point trajectories around a human body contain rich temporal information on the person's motion[7, 8], so we used key-point trajectories as a feature for detecting events. Our system creates trajectory histograms that are related to the motion and appearance of key points. The number of bins in the histograms is fixed irrespective of the duration of the trajectory. Thus, the trajectory histograms can be used as features in a bag-of-features approach [9].

Many techniques that extract fixed-dimensional features from a key-point trajectory by creating histograms have recently been proposed [10, 11]. Our system also creates a histogram from motion vectors that are contained in a trajectory as a feature for motion. The directions and magnitudes of motion vectors are used to create the motion histogram. In addition, the system also creates an LBP histogram from the pixels around key points on a trajectory as an appearance feature.

The size of a bag in a bag-of-features approach could be an issue. If the system regarded a whole image as one bag, trajectories belong to different persons would be mixed in the bag. Therefore, we decided to segment regions for each person and regarded each person region as one bag. The system applied the bag-of-features approach to each segmented region.

Our algorithm is described in detail in the following sections.

2.2 Proposed system

Our proposed system consists of four steps (shown in Figure 1). In step 1, the system detects key points and tracks them until they disappear. It does this by first extracting the foreground from an entire image by statistic background subtraction, which uses the average and variance of each pixel value. Next, it detects key points in the foreground with a Harris operator and tracks them by calculating the optical flow using the Lucas-Kanade method. The optical flows are then connected and stacked into key-point trajectories.

In step 2, extracted trajectories are segmented with reference to each distance. We used a dendrogram algorithm to cluster the trajectories. After this step, individual recognition processes are performed for each trajectory cluster.

In step 3, the system extracts features from each trajectory and describes them as a fixed dimensional histogram. It creates two types of histogram, one of which is related to the key point's motion and the other to its appearance. The number of bins in both histograms is fixed, so it can describe any trajectory as a fixed dimension regardless of its length.

Finally, the action is classified in step 4. We used the bag-of-features approach and a support vector machine (SVM) to classify human actions. The two histograms created in step 3 are used as a feature for the bag-of-features approach. The SVM classifier was learned in the training phase using the trajectories around a person who had performed a target action. We originally annotated correct data using the development dataset cross-referenced with NIST annotated correct data. We manually annotated bounding boxes that each contained one target person. The trajectories in the boxes were then used as correct trajectories.

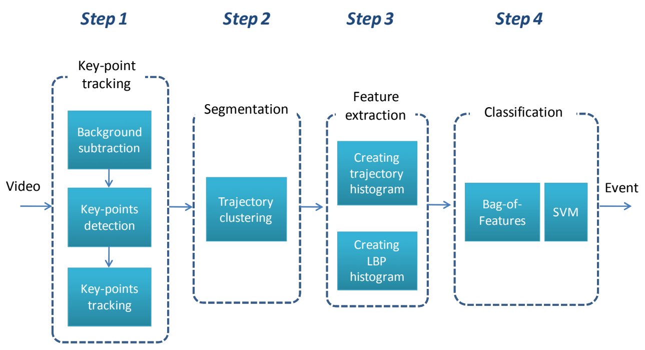

Figure 1 Flow of event detection processing.

2.2.1 Key-point detection and tracking

2.2.1.1 Background subtraction

All the surveillance video used in this task was shot with a fixed camera, so background subtraction is suitable for detecting human regions. However, at some level in the sequence the luminance changes, which renders the static background image unsuitable for robust human region detection. Thus, we updated the background image dynamically by calculating the mean value of brightness and its amplitude for every pixel. The system robustly extracts only moving regions, such as humans and their baggage, as shown in Figure 2.

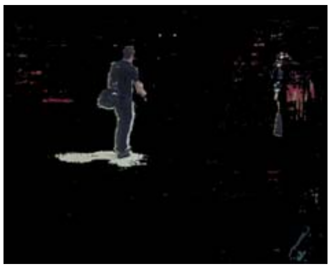

Figure 2 Foreground regions extracted by statistical background subtraction.

2.2.1.2 Key-point tracking

A standard Kanade-Lucas-Tomasi (KLT) tracker [12] is used to track key points on the input video. The KLT tracker is an algorithm that selects and keeps track of feature points that are optimal for tracking. It is widely used in visual feature tracking.

In the proposed system, a Harris operator is used for detecting feature points in the input image.

Next, feature points in the background regions are removed by referencing the mask image that was created in the above-mentioned background subtraction process. Then, only feature points in the foreground regions are detected.

The detected key points are tracked by calculating the optical flow on the basis of the Lucas-Kanade method. These key points are tracked until the feature point disappears, as shown in Figure 3.

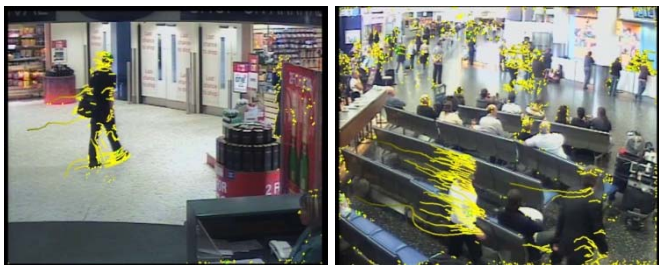

Figure 3 Key-point detection and tracking.

2.2.2 Segmentation of key-point trajectories

Many people appear in surveillance cameras sequences, and many key-point trajectories are extracted from them. The key-point trajectories should be segmented for each person because each person individually performs various actions.

We used the bag-of-features model as a classification method, so the size of the bag is a critical parameter for accurate recognition. If we set the size of a bag as an entire image, the system cannot perform robust recognition. Ideally, the bag size should be about the same size as the region of one human. Thus, we segmented detected trajectories on the basis of the distance between each of them.

The dendrogram algorithm was used to segment the key-point trajectories. Trajectory clusters were created by concatenating trajectories that were close to each other. The dendrogram algorithm needs a threshold to stop clustering, so we set the threshold in accordance with the average size of the manually annotated human regions for each camera. We achieved almost completely individual processing for each person by applying the bag-of-features model to each trajectory cluster.

A sample of the segmentation is shown in Figure 4. Rough shapes of the trajectory clusters are indicated by ellipses.

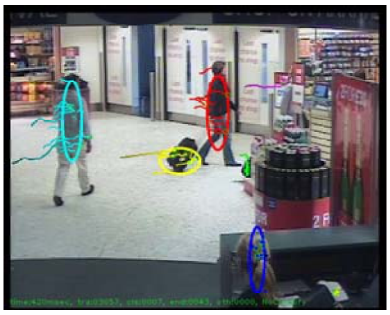

Figure 4 Sample of segmentation.

2.2.3 Feature extraction

2.2.3.1 Motion feature

Key-point trajectories convey critical temporal information on human behaviors and contain the motion vectors of each frame. These represent the movement of a particular key point, so the motion vectors are used as features in terms of motion.

However, features are unsuitable for direct use in the bag-of-features model because trajectories have variable time lengths. Therefore, each trajectory should be transformed into a fixed-length descriptor that attempts to capture the key characteristics of each motion.

The number of bins in a histogram is usually fixed. Thus, we created a histogram as a feature for the bag-of-features model. To describe a key point's motion, we extracted the direction and magnitude of motion vectors from each trajectory. The direction was divided into eight and the magnitude was divided into four, including 0.

The surveillance cameras are set on a ceiling, so the motion vectors of a person who is near the camera are large and those of a person who is far from the camera are small, despite the fact that they may have been performing the same action. Therefore, the magnitude of a motion vector should not be classified by using absolute thresholds. We set relative thresholds of magnitude for each cluster individually.

First, the average value of magnitude μ and its standard deviation σ are calculated from all motion vectors in a cluster. After that, thresholds of magnitude are defined using the μ and the σ. The ranges are described as 0, (0, μ-σ/2], (μ-σ/2, μ+σ/2], and (μ+σ/2, ∞). The direction of the motion vector is divided into eight, so a motion vector is classified into any one of 25 (= 8 × 3 + 1) labels, as shown in Figure 5.

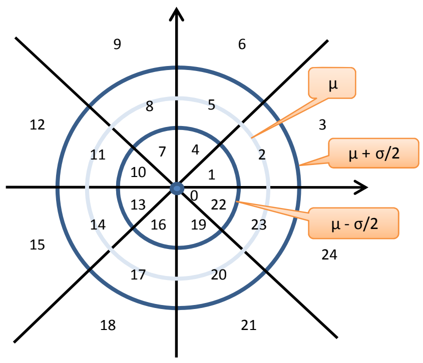

Figure 5 Label of histogram related to key point's motion.

We used a Haar low-pass smoothing filter for each raw trajectory to extract motion information at different time-scales. A smoothed trajectory was generated by applying the filter.

The node points in the smoothed trajectory are calculated by Equations (1)–(3). Let $\boldsymbol{P}_{u,q}$ be a set of node points in the u-th trajectory, $\boldsymbol{p}^{x}_{u,q}$ be the set of its x-coordinate, and t_1 and t_2 be the frame number of appeared and disappeared key points. The q means the smoothing level of a trajectory under the maximum number Q, and the level 0 (q = 0) means a non-filtered raw trajectory. Several levels of trajectories are created by increasing the number q. This time, we set the Q to 1 so that only one smoothed trajectory was created. The y-axis elements of the trajectory are calculated similarly.

A histogram for a smoothed trajectory with 25 bins was created, as well as a raw trajectory. Thus, a trajectory histogram with 50 bins is created as a motion feature.

$$\boldsymbol{P}_{u,q} = [\boldsymbol{p}^{x}_{u,q}, \boldsymbol{p}^{y}_{u,q}] \quad (1)$$

$$\boldsymbol{p}^{x}_{u,q} = [\boldsymbol{p}^{x,t_1+2^q-1}_{u,q}, \boldsymbol{p}^{x,t_1+2^q}_{u,q}, \boldsymbol{p}^{x,t_1+2^q+1}_{u,q}, \dots, \boldsymbol{p}^{x,t_2}_{u,q}] \quad (2)$$

$$\boldsymbol{p}^{x,t}_{u,q} = \frac{1}{2^q}\sum_{i=0}^{2^q-1} \boldsymbol{p}^{x,t-i}_{u,0} \quad (3)$$

2.2.3.2 Appearance feature

The above-mentioned trajectory histogram is a feature of a key point's motion. We also need a feature of a key point's appearance to ensure the accuracy of the human action recognition. SIFT and SURF are typical appearance descriptors [3, 4], but we used the LBP histogram as the appearance feature descriptor due to its superior processing speed [13].

The LBP histogram represents small and large patterns of pixel values between the target pixel and its neighbors. We used eight neighbor pixels to create the LBP. The small and large combination between a target pixel and eight neighbor pixels constructs the 8-bit information volume for a total of 256 LBP patterns.

The creation process of the LBP histogram is shown in Figure 6. First, the system extracts regions of 16 × 16 pixels around each key-point in a trajectory. Next, it creates an average image by averaging the pixel values of all pixels. Finally, it calculates the LBP for each pixel in the average image and creates the LBP frequency histogram.

Finally, a 306-dimensional feature that contains 50 bins of motion histogram and 256 bins of appearance histogram is created.

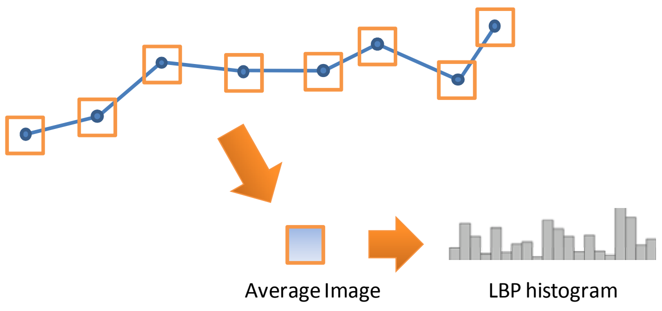

Figure 6 Creation of LBP histogram.

2.2.4 Classification

An event is classified by using the bag-of-features model. A segmented region is treated as a bag and a 306-dimensional motion-appearance feature descriptor in the region is treated as the feature.

Each feature is labeled as the nearest cluster by referring to a code book, which we created beforehand out of many feature descriptors by using the *k*-means method. The final feature vector as an input for the SVM classifier is represented as a cluster histogram by counting the feature descriptors that have been labeled for each cluster.

An event classifier is created by training with the cluster histograms of each event. The multi class SVM-supervised machine learning method is used for training the classifier.

In the test phase, the system classifies the events for every segmented region every frame by evaluating a cluster histogram that was made with trajectory features in the region. The decision score for each event is determined on the basis of the detected frequency in the duration of the detected event. If the decision score is beyond a specific threshold, the system considers the event to actually have occurred.

2.3 Results

2.3.1 Parameters

We trained the system in accordance with the following settings. The number of cluster *k* was set to 100 during the creation of the code book. The SVM classifier was trained with cluster histograms that were based on five days of sample trajectory features from the development dataset (LGW2007_1101, 1106, 1107, 1108, and 1112). The classifier recognizes "no event" as well as our four target events. Four SVM classifiers were created for four cameras (except Cam4).

2.3.2 Results and discussion

The performance results of our system are shown in Table 1. The result of the "CellToEar" event was better than the others, probably because this event is usually performed in situations in which there is no one else around the target person, making it easy to segment a human region. Moreover, the upward motion vector is likely to appear in this event, and the variation of the time-length is not as large as that in the other events. We conclude that the directional and temporal simplicity of the motion contributed to the recognition accuracy.

Our system often identified the "Pointing" event incorrectly. This is probably because the event is usually performed in situations in which neighboring people are present, making it difficult to segment a human region. Moreover, the event has significant direction and magnitude variations that differ greatly depending on the individual. For example, one person might point with a large arm-gesture while another might point with a small finger-gesture, etc. The accuracy might be improved by training the system to differentiate between "large pointing gesture" and "small pointing gesture," or between "left pointing" and "right pointing."

Table 1 Results for detecting events

Event	#Ref	#Sys	#Cor Det	Act. DCR	Min. DCR
CellToEar	194	1447	3	1.0377	1.0039
Embrace	175	3849	31	1.0865	1.0003
ObjectPut	621	9216	10	1.1649	1.0003
Pointing	1063	13974	41	1.3671	1.0003

2.4 Conclusion

We developed a system of automatically detecting specific human actions ("Pointing," "CellToEar," "ObjectPut," and "Embrace") in video sequences shot by fixed cameras installed at an airport. Our system creates a trajectory histogram that is related to a key point's motion and appearance. It then identifies the human actions by using the bag-of-features model. The regions of each person are automatically segmented and then the bag-of-features model is applied to identify human actions for each segmented region. Results showed that the system could robustly detect many small motions, especially those in the "CellToEar" action. We plan to apply this framework to other human motion recognition systems, such as man-machine interfaces, that manage small motions.

3. Semantic Indexing

3.1 Overview

An overview of the proposed system is shown in Figure 7. This system begins by extracting keyframes from each shot. A keyframe is extracted every time the cumulative sum of the frame difference exceeds a certain threshold. As a result, a small number of keyframes will be extracted from a video section with little camera motion while many keyframes will be extracted from a long shot or a video section with much camera motion. Next, the system calculates a feature vector from extracted keyframes by combining local and global features. For local features, a new feature that takes into account co-occurrence between feature points is proposed, and for global features, color moment, Haar wavelet texture, and a local binary pattern (LBP) [13] are used in the same as in the TRECVID 2010 system [14]. Finally, to determine whether the shot includes the target concepts, the system classifies the calculated feature vector by the random forests. The random forests classifier is trained for each concept.

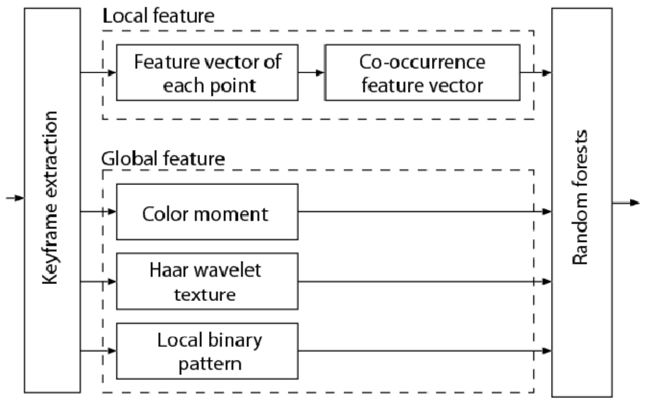

Figure 7 Overview of proposed system.

3.2 Local feature

Calculating local features, the proposed method firstly calculates edge direction and magnitude at each pixel. Next, feature points are selected by grid sampling, and an edge direction histogram is computed from the local area surrounding each feature point. This histogram is calculated for various image scales to ensure robustness to scale variation and to obtain the co-occurrence relationship between different scales. A co-occurrence feature vector is then calculated on the basis of the relationship between feature points. The co-occurrence feature vectors are averaged out for several block areas, and the resulting vectors are concatenated to give the local feature vector for the entire keyframe.

3.2.1 Feature point extraction

The previous method [1, 2] detected feature points on the basis of difference of Gaussians [3], and it suffers from a large variation in the number of detected feature points depending on image patterns. The accuracy of detecting concepts will drop if a sufficient number of feature points is not obtained. The proposed method uses grid sampling to obtain feature points at a certain pixel interval so that a fixed number of feature points can be obtained from any image.

3.2.2 Feature vector of feature point

An edge direction histogram is calculated from the local area around each feature point. This process is explained by using the example of a feature point at coordinate location (x, y). An edge direction histogram is determined on the basis of spatial weighting by using a Gauss window of dispersion σ. When quantizing edge direction $\theta(x, y)$ in n bins, each element hi of the edge direction histogram $h_\sigma = (h_1, h_2, \ldots, h_n)$ can be calculated by

$$h_i = \sum_u \sum_v G(u, v, \sigma) \cdot m(x+u, y+u) \cdot \delta_i(\theta(x+u, y+v)) \tag{4}$$

where G(u, v, σ) implies weight for edge magnitude m(x+ u, y + v) and represents a Gauss window of dispersion σ such that the weight becomes larger as the distance from point (x, y) becomes shorter. And, $\delta_i(\theta)$ is a function that returns 1 when quantized θ belongs to the ith bin of the histogram and 0 otherwise.

The histogram given by Eq. (1) is calculated for various values of Gauss-window dispersion σ. The resulting histograms are concatenated to give feature vector $g_{x,y}$, as expressed by

$$\mathbf{g}_{x,y} = (\mathbf{h}_{\sigma_1}, \mathbf{h}_{\sigma_2}, \ldots, \mathbf{h}_{\sigma_t}), \quad \sigma_i = \sigma_0 \cdot s^{i-1} \tag{5}$$

Here, σ_0 and s denote the initial value of Gauss-window dispersion and the rate of dispersion change, respectively. In our submissions, t = 3, $s = \sqrt{2}$, and σ_0 = 1.6 were used on the basis of previous experiments. Varying Gauss-window dispersion makes it possible to obtain features corresponding to various scales.

As a substitute for the edge direction histogram, other kinds of basic features can be used to calculate a feature vector. One is the frequency histogram of LBP [13]. As in the case of the edge direction histogram, a feature vector can be obtained by calculating the weighted frequency histogram of the LBP from the vicinity of the feature point.

3.2.3 Co-occurrence feature vector

The co-occurrence feature vector is calculated on the basis of the spatial co-occurrence of feature vector **g**. Co-occurrence feature matrix **U***x,y,u,v* between feature points (*x, y*) and (*x* + *u*, *y* + *v*) is defined as

$$\mathbf{U}_{x,y,u,v} = \mathbf{g}_{x,y} \cdot \mathbf{g}^{t}_{x+u,y+u} \tag{6}$$

where **U** is a square matrix of order |**g**|, which is converted to the form of a one-dimensional vector denoted as **U'** by lining up each row of **U**.

The proposed method create a co-occurrence feature vector based on the relationship between base point (*x, y*) and target point (*x*+*u, y*+*v*) lying within a certain distance from that base point. Specifically, **U'** for the 19 target points (including the base feature point) shown in Figure 8 is considered, and the 19 vectors are concatenated to give a co-occurrence feature vector with respect to feature point (*x, y*).

Co-occurrence feature vectors for each feature point are summed up within a block area in later processing, and to prevent the same combination of points being summed up more than once, points at symmetrically opposed positions about feature point (*x, y*) are therefore excluded from Figure 8.

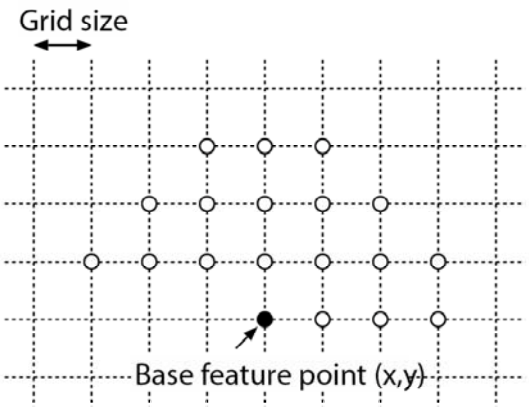

Figure 8 Calculation of co-occurrence feature vector.

3.2.4 Local feature vector

Figure 9 shows an overview of calculating the local feature vector for the entire keyframe. First, the keyframe image is divided into 2×2 and 1×3 block areas [15] so that the position where an object appears can be considered, and the co-occurrence feature vectors are averaged out for each of these areas. Finally, a feature vector for the entire keyframe is obtained by concatenating the average co-occurrence feature vectors from each block area.

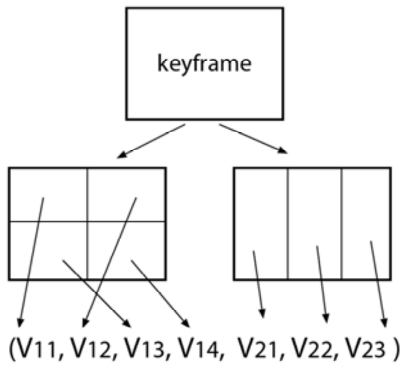

Figure 9 Calculation of local feature vector.

3.3 Global feature

Our method uses three kinds of global feature.

3.3.1 Color moment

The color moments represent the color distribution in an image. We convert the input image into the HSV color space and L*a*b* color space and calculate the average pixel value μ_c, the standard deviation σ_c, and the cube root of skew s_c for each component c ($c \in \{h, s, v, l, a, b\}$). We divide the image into a 5 × 5 grid, calculate μ_c, σ_c, and s_c for each grid region, then link them to form a feature vector.

3.3.2 Haar wavelet texture

The Haar wavelet reflects texture in an image. We divide the input image into a 3 × 3 grid and apply two dimensional Haar wavelet transforms in three stages to each grid region. We then calculate the variance of luminance values for each subband region and link them together to obtain a feature vector.

3.3.3 Local binary pattern

The LBP [13] denotes the density magnitude pattern of pixels surrounding the target pixel. The equation of the LBP $L_{P,R}$ that is calculated from P pixels around the circumference of a circle of radius R is:

$$L_{P,R}(x,y) = \begin{cases} \sum_{p=0}^{P-1} \delta_{P,R}(x_p, y_p) & \text{if} \quad U_{P,R}(x,y) \leq 2 \\ P+1 & otherwise \end{cases} \qquad (7)$$

where $\delta_{P,R}$ is a function that returns 1 if the luminance value of the surrounding pixels $(x{+}x_p,\ y{+}y_p)$ is greater than that of the target pixel (x, y), or 0 if it is smaller. Here, $x_p = \mathrm{R}\cos\frac{2\pi p}{P}$ and $y_p = \mathrm{R}\sin\frac{2\pi p}{P}$. The $U_{P,R}$ denotes the total number of times that 0 and 1 change in the $\delta_{P,R}$ sequence, which is given by:

$$U_{P,R}(x,y) = \left|\delta_{P,R}(x_{P-1}, y_{P-1})\right| + \sum_{p=1}^{P-1} \left|\delta_{P,R}(x_p, y_p) - \delta_{P,R}(x_{p-1}, y_{p-1})\right| \qquad (8)$$

To achieve scale invariance, we calculate the frequency histogram of $L_{P,R}(0 \leq L_{P,R} \leq P{+}1)$ for three combinations of (P,R) = (8, 1), (16, 2), (24, 3) and link them together to obtain a feature vector.

3.4 Random forests classifier

The random forests method [6] is used to determine whether an input keyframe has a specific concept. Random forests is a kind of ensemble learning, and it gives highly accurate classifications by using a combination of decision trees (CART) [16]. Some researchers assert that random forests is superior to methods such as bagging or boosting in certain cases. In addition, random forests can complete the learning process in a short time even for high-dimension feature vectors by searching for the best feature for the branching node in a subset of vector elements.

The random forests algorithm works well when the training data for two classes (including and not-including the concept) are roughly the same in number, but the classification error is rather unbalanced when one class is much larger than the other. The conventional method attempts to resolve the problem by applying a higher weight to the smaller class [6]. However, the bootstrap samples generated by the conventional method contain few data with high weights and many data with low weights, and this situation could cause over-training. Thus, we propose a new sampling method for creating the bootstrap samples; it ensures that each class is selected with equal probability. The data is selected with replacement and is not weighted. If the number of bootstrap samples is small relative to the amount of training data, various data are also selected from the minority class, making it possible to generate a classifier with high generalization capability.

3.5 Experiments

3.5.1 Settings

We participated a full task targeting 346 concepts and used four different settings as listed in Table 2. Each run differed in terms of the method used to calculate the local feature vector. Run 1 used the conventional bag-of-visual-words of SIFT (“siftbow”). Run 2 used the proposed method based on the co-occurrence of edge direction histograms (“coedge”). Run 3 was a combination of Run 1 and Run 2 (“siftbow + coedge”). Run 4 used a method that calculated the co-occurrence feature vector by using a frequency histogram of a LBP (“colbp”) instead of the edge direction histogram in Run 2. The training type was type A (using only IACC training data) in all runs. The methods used to calculate global features and to train the classifier were the same for each run.

Table 2 Setting of each run.

Run	System ID	Training type	Local feature type
1	NHKSTRL1	A	siftbow
2	NHKSTRL2	A	coedge
3	NHKSTRL3	A	coedge+sfitbow
4	NHKSTRL4	A	colbp

3.5.2 Experimental results

The evaluation results of the four runs are shown in Table 3. A mean inferred average precision (infAP) of about 0.08 was obtained, and the method with the highest precision was coedge, achieving a mean infAP of 0.083. Next best was the siftbow method, with a precision of 0.082. These results show that the coedge method improves slightly upon the accuracy of siftbow by considering co-occurrence between feature points. Meanwhile, the colbp method using co-occurrence of a LBP had the lowest precision of all four methods. The reason for this low precision is thought to be that, compared to the coedge method, a feature descriptor of the colbp method is excessively detailed and thereby a classifier that is overly adapted to the training data was generated. It is expected that the colbp method is effective for concepts with little variation in object shape or texture.

Table 3 Mean infAP of each run.

Local feature type	Mean infAP
siftbow	0.082
coedge	0.083
coedge+siftbow	0.081
colbp	0.080

Table 4 shows a detailed comparison between the siftbow and coedge methods giving the highest mean infAP scores. On examining the results for each concept, it can be seen that the siftbow method is more precise than the coedge method for some concepts and vice versa for others. For example, siftbow is more precise for concepts like “26.Charts,” “97.Reporter,” “113.Streets,” and “431.Skating,” while coedge has a higher detection precision for concepts like “41.Demonstration Or Protest,” “86.Old People,” “105.Singing,” and “392.Quadruped.” The coedge method is particularly more effective for concepts with little variation or objects having shapes in which co-occurrence between feature points is present.

3.6 Conclusion

This paper proposed a semantic indexing method of detecting various concepts from a large video archive. The proposed method calculated a feature vector on the basis of a novel local feature that considers the spatial relationship between feature points. To reduce a computational cost, random forest method was used for constructing classifiers. Evaluation results showed that the proposed method improves a mean average precision compared to the existing bag-of-visual-words method. Future work includes improving image features to further improve detection precision and investigating the use of audio features in combination with image features.

Table 4 Evaluation results for each concept.

Consept	InfAP	
	siftbow	coedge
2. Adult	0.059	0.048
5. Anchorperson	0.346	0.351
10. Beach	0.11	0.129
21. Car	0.066	0.051
26. Charts	0.055	0.019
27. Cheering	0.081	0.078
38. Dancing	0.006	0.011
41. Demonstration_Or_Protest	0.027	0.043
44. Doorway	0.024	0.032
49. Explosion_Fire	0.141	0.136
50. Face	0.085	0.081
51. Female_Person	0.029	0.032
52. Female-Human-Face-Closeup	0.057	0.051
53. Flowers	0.048	0.05
59. Hand	0.009	0.007
67. Indoor	0.059	0.06
75. Male_Person	0.044	0.041
81. Mountain	0.122	0.134
83. News_Studio	0.289	0.294
84. Nighttime	0.023	0.028
86. Old_People	0.017	0.022
88. Overlaid_Text	0.093	0.113
89. People_Marching	0.023	0.014
97. Reporters	0.303	0.29
100. Running	0.018	0.014
101. Scene_Text	0.011	0.011
105. Singing	0.017	0.034
107. Sitting_Down	0.009	0.001
108. Sky	0.192	0.199
111. Sports	0.087	0.094
113. Streets	0.098	0.079
123. Two_People	0.017	0.013
127. Walking	0.047	0.047
128. Walking_Running	0.037	0.042
227. Door_Opening	0.008	0.005
241. Event	0.024	0.033
251. Female_Human_Face	0.042	0.04
261. Flags	0.009	0.006
292. Head_And_Shoulder	0.065	0.058
332. Male_Human_Face	0.038	0.047
354. News	0.134	0.247
392. Quadruped	0.032	0.047
431. Skating	0.202	0.163
442. Speaking	0.091	0.073
443. Speaking_To_Camera	0.09	0.094
454. Studio_With_Anchorperson	0.382	0.368
464. Table	0.027	0.029
470. Text	0.091	0.09
478. Traffic	0.115	0.117
484. Urban_Scenes	0.087	0.078

References

[1] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," In Proc. ICCV'03, 2003.

[2] G. Csurka, C. Bray, C. Dance and L. Fan, "Visual categorization with bags of keypoints," in Proc. ECCV Workshop on Statistical Learning in Computer Vision, pp.59–74, 2004.

[3] D.G. Lowe, "Object recognition from local scaleinvariant features," In Proc. ICCV'99. vol.2. pp.1150– 1157, 1999.

[4] H. Bay, A. Ess, T. Tuytelaars and L.V. Gool, "SURF: speeded up robust features," Computer Vision and Image Understanding, vol.110, no.3, pp.346–359, 2008.

[5] "TREC video retrieval evaluation notebook papers and slides,"http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html

[6] L. Breiman, "Random forests," Machine Learning, vol.45, pp.5–32, 2001.

[7] P. Matikainen, M. Hebert, and R. Sukthankar, "Trajectons: Action recognition through the motion analysis of tracked features," Workshop on Video-Oriented Object and Event Classification (ICCV), Sep. 2009.

[8] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," In Proc of ICCV, pp.104–111, Sept. 2009.

[9] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," ECCV In Workshop on Statistical Learning in Computer Vision, pp.1–22, 2004.

[10] V Mezaris, A Dimou, and I Kompatsiaris, "Local invariant feature tracks for high-level video feature extraction," Proc. 11th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2010), Apr. 2010.

[11] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li, "Hierarchical spatio-temporal context modeling for action recognition," In Proc of CVPR, pp.2004–2011, 2009.

[12] J. Shi and C. Tomasi, "Good features to track," IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.593–600 (1994).

[13] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution grayscale and rotation invariant texture classification with local binary patterns," IEEE Trans. Pattern analysis and machine intelligence, vol.24, no.7, pp.971–987, 2002.

[14] Y. Kawai, M. Takahashi, M. Fujii, M. Naemura and S. Satoh, "NHK STRL at TRECVID 2010: semantic indexing and surveillance event detection," In Proc. TRECVID 2010 workshop, 2010. http://wwwnlpir.nist.gov/projects/tvpubs/tv10.papers/nhkstrl.pdf

[15] S.-F. Chang, J. He, Y.-G. Jiang, E.E. Khoury, C.- W. Ngo, A. Yanagawa and E. Zavesky, "Columbia university/VIREO-City/IRIT TRECVID2008 high-level feature extraction and interactive video search," In Proc. TRECVID 2008 Workshop, 2008.

[16] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, " Classification and regression trees," Wadsworth and Brooks, 1984.