NHK STRL at TRECVID 2010: Semantic Indexing and Surveillance Event Detection

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

[†]Science and Technical Research Laboratories, NHK, 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

NHK Science and Technical Research Laboratories (NHK STRL) participated in two tasks at TRECVID 2010: semantic indexing (full submission) and surveillance event detection.

For the semantic indexing task, we used a method based on the bag-of-features approach [1, 2]. This approach has been shown to be effective in general object recognition and in past work at TRECVID [3]. The proposed method aims to improve detection accuracy by combining gradient features in the local region with global features such as texture and color distribution. At TRECVID 2010, the length of video datasets and the number of target concepts increased significantly compared to the previous year, and reducing computational cost became an important issue. We use random forests [4] instead of the support vector machine (SVM) widely used in existing techniques as a classification algorithm to reduce the computational time needed for training and detection.

For the surveillance event detection task, we targeted "Pointing," "CellToEar," and "ObjectPut" events. Our proposed system identifies these events based on the bag-of-features approach. We used a fixed-dimensional feature descriptor that was extracted from a key-point trajectory as a feature. A feature weight was calculated for reducing the interference of noise trajectories in the background regions. Our system achieved a relatively strong performance with these small human behaviors.

This paper is organized as follows. In section 2, we describe the proposed method for the semantic indexing task in detail and present experimental results. In section 3, we describe the proposed method for the surveillance event detection task in detail and present the results of its evaluation. We conclude the paper in section 4.

2 Semantic indexing task

The proposed system for semantic indexing is shown in Figure 1. The basic configuration of the system is about the same as the NHK system used at TRECVID 2009, but enhancements have been made to the number of keyframes for each shot, types of global features, and classification algorithm.

First, the system extracts keyframes from each shot. It extracts a total of three frames, one each from the beginning, middle, and end of a shot, and uses these frames as representative pictures of the shot. It then calculates local features and global features from extracted keyframes and creates feature vectors by concatenating these features. For local features, the system first calculates keypoints and feature descriptors from each keyframe using two algorithms, namely, SIFT [6] and SURF [7]. Then, for the output of each algorithm, it obtains visual words by clustering the extracted features and computes feature vectors based on the frequency of appearance of visual words. Here, we use a weighting technique [8] that takes the distance between a keypoint and visual words into account. For global features, we use color moment, Haar wavelet texture, and local binary pattern [9] for each grid region. Finally, we use the random forests method to classify the feature vectors, and we determine whether the target concepts are included in the shot. The random forest classifier is trained beforehand for each concept. The following describes each process in detail.

2.1 Local feature

The proposed method uses SIFT [6] and SURF [7] to calculate local features. We expect that combining two different algorithms ensures that important points of interest will be detected and the features of an object to be captured accurately. The method calculates feature vectors for each grid region, as shown in Figure 2, so that positional information within the frame image can be taken into consideration. For the number of divisions of the grid, we decided on 2×2 and 1×3 after reviewing the experimental results of previous work [10].

To create feature vectors, we use a method that considers the distance between a feature descriptor at a detected keypoint and a visual word. This differs from the conventional method, which allocates a detected key-

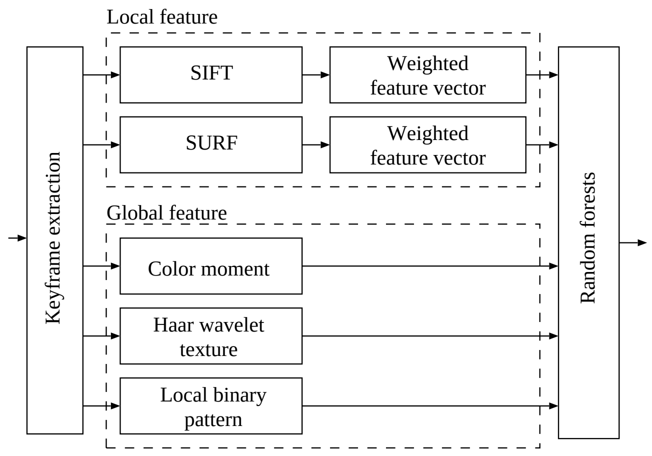

Figure 1: Overview of semantic indexing.

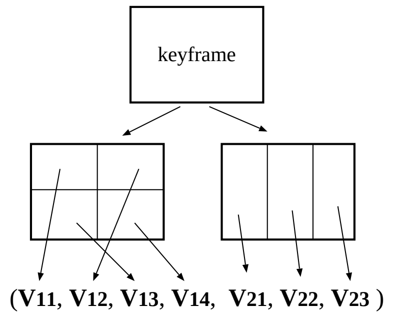

Figure 2: Calculation of local feature vector

point to one visual word, and is based on the idea that one keypoint can belong to a number of visual words. When there are K visual words, we calculate a K-dimensional feature vector $\boldsymbol{T} = (t_1, ..., t_k, ..., t_K)$. We calculate each vector element t_k by using the following equation [8]:

$$t_k = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \frac{1}{2^{i-1}} sim(p_j, v_k), \tag{1}$$

where M_i denotes the total number of keypoints at which the ith closest visual word is v_k, and $sim(p_j, v_k)$ denotes the degree of similarity between keypoint p_j and the visual word v_k. N is a constant that denotes the depth of distance considered, where N is set to 4 for the proposed method, in a similar manner to a previous study [8].

2.2 Global feature

Our method uses three kinds of global feature.

2.2.1 Color moment

The color moment feature represents 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.

2.2.2 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.

2.2.3 Local binary pattern

A local binary pattern (LBP) [9] 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:

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

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, $x_p = R\cos\frac{2\pi p}{P}$ and $y_p = R\sin\frac{2\pi p}{P}$. 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:

$$U_{P,R}(x,y) = |\delta_{P,R}(x_{P-1}, y_{P-1})| + \sum_{p=1}^{P-1} |\delta_{P,R}(x_p, y_p) - \delta_{P,R}(x_{p-1}, y_{p-1})|. \tag{3}$$

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.

2.3 Random forests method

The random forests method [4] 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) [11]. 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

Table 1: Settings of each run.

Run	System ID	Training type	# of trees
1	NHKSTRL1	A	500
2	NHKSTRL2	A	200
3	NHKSTRL3	A	100

Table 2: Mean infAP of each run.

Run	mean InfAP
1	0.0349
2	0.0344
3	0.0338

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 [4]. 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.

2.4 Experiments

2.4.1 Settings

Three settings were used in the experiments, as shown in Table 1. The training type was type A (only IACC training data were used) in all runs, and the number of decision trees built in random forests was changed for each run. The results of collaborative annotation were used as label data for training purposes.

2.4.2 Experimental results

The evaluation results for each run are summarized in Table 2. The mean inferred average precision (infAP) of Run 1 was 0.0349, the highest precision of all three runs. Run 2 had the next highest precision and Run 3 the lowest. The difference in precision among these three runs was slight, indicating that varying the number of decision trees did not lead to a significant change in precision.

Precision results for each concept in Run 1 are listed in Table 3. The concept with the highest precision was “115 Swimming” (infAP score of 0.329). This level of precision was much higher than those of other participants. We believe that features we used such as texture and color were very effective. The infAP scores for the other concepts were around 0.05. To achieve even higher

Table 3: Results of semantic indexing (Run 1).

#	Concept	infAP
4	Airplane_Flying	0.030
6	Animal	0.014
7	Asian_People	0.002
13	Bicycling	0.003
15	Boat_Ship	0.022
19	Bus	0.002
22	Car_Racing	0.003
27	Cheering	0.020
28	Cityscape	0.050
29	Classroom	0.002
38	Dancing	0.046
39	Dark-skinned_People	0.043
41	Demonstration_Or_Protest	0.031
44	Doorway	0.023
49	Explosion_Fire	0.008
52	Female-Human-Face-Closeup	0.050
53	Flowers	0.021
58	Ground_Vehicles	0.063
59	Hand	0.012
81	Mountain	0.073
84	Nighttime	0.045
86	Old_People	0.008
100	Running	0.024
105	Singing	0.014
107	Sitting_Down	0.003
115	Swimming	0.329
117	Telephones	0.003
120	Throwing	0.000
126	Vehicle	0.056
127	Walking	0.048
	Average	0.0349

Table 4: Processing time required for training (Run 1).

Processing	Processing time (hh:mm)
Feature calculation	14:21
Random forest training	24:58
Total	39:19

precision, it will be necessary to combine image features with audio features. It may also be possible to improve precision by using semantically high level features, for example, ones taken from the results of facial detection.

The proposed method builds a classifier using the random forests method, which is known for its low computational cost. This made the use of a special computer unnecessary and made the time from computing features to building a classifier relatively short. The processing time required for training in Run 1 is listed in Table 4. The computer used in the experiment had an Intel Xeon 3.4-GHz CPU and 4-GB memory. The time required to compute the feature vectors was 14 hours and 21 minutes, while the time required for training 130 kinds of classifiers corresponding to each concept was 24 hours and 58 minutes. The training time of a random forest classifier for one concept was 12 minutes on average. Total processing time was 39 hours and 19 minutes. A preliminary experiment using the SVM method revealed that it took 20 hours on average to train a classifier for one concept. The proposed method therefore greatly reduced processing time. Computational complexity is an important element in the processing of large amounts of video data. We believe it would be useful to investigate methods that take into account not just precision but also computational cost.

3 Surveillance event detection

The importance of human motion recognition technology has recently been increasing [13]. The rapid spread of surveillance cameras has increased the demand for technology that can automatically identify human behaviors.

Motion recognition techniques can be widely applied to many services, such as motion-based video searches and man-machine interfaces. This year, we targeted "Pointing," "CellToEar," and "ObjectPut" events with a focus on general versatility.

Key-point trajectories around a human region contain temporal information on the human's motion [14], so we used key-point trajectories as the feature for detecting events.

However, the time-length of a key-point trajectory is not fixed because each trajectory has a different duration. This means the bag-of-features approach cannot be used because it uses a fixed-length feature vector. A method that extracts fixed-dimensional features from a key-point trajectory has recently been proposed [15]. Our system extracts fixed-dimensional features that are related to the trajectory's direction and the speed in addition to using the method. The appearance feature is also extracted on the trajectory as a fixed 128-dimensional SURF descriptor [7]. Therefore, the proposed feature contains both motion and appearance information.

Feature weight is used in the bag-of-features approach to reduce the interference of noise feature descriptors in the background region. A descriptor is weighted based on the tf-idf approach [16], which is often used in the field of natural language processing. This feature weight technique has improved the performance of our system.

Our algorithm is described in detail in the following sections.

3.1 Overview

Our system consists of three steps, which are shown in Figure 3. Step 1 is key-point detection and tracking. The system detects foreground regions based on statistical background subtraction, and only the key-points on the foreground are extracted. These are then tracked by the Lucas-Kanade method.

Step 2 is feature extraction. The system extracts fixed-dimensional motion feature descriptors from a key-point trajectory. The SURF descriptor is also extracted at the end point of a trajectory. Both descriptors are integrated into a fixed 380-dimensional descriptor.

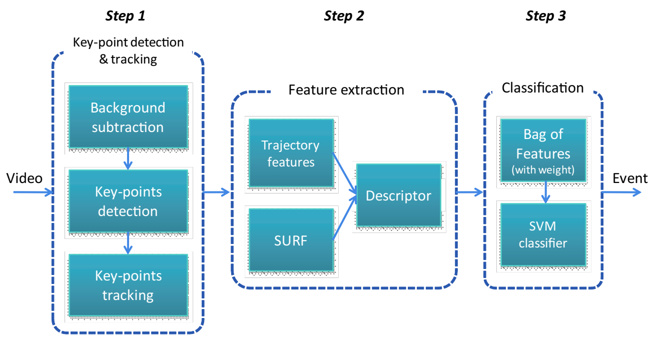

Figure 3: Overview of surveillance event detection.

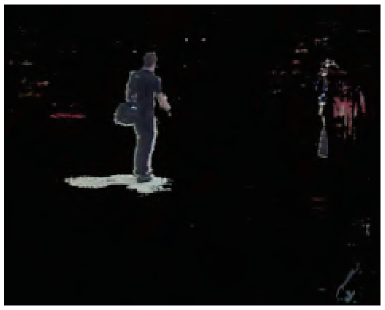

Figure 4: Extracted foreground regions by using statistical background subtraction.

Step 3 is classification. The bag-of-features approach and an SVM classifier [17] are used in the system to detect events. Feature weights are calculated based on the tf-idf method when a cluster histogram is created. We explain the processes in each step in the following subsections.

3.2 Key-point detection and tracking

3.2.1 Background subtraction

All the surveillance video used in this task was shot by fixed camera, so the 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 update 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 4.

3.2.2 Key-point tracking

A standard Kanade-Lucas-Tomasi (KLT) tracker [18] is used to track key points on the input video. The KLT tracker is an algorithm that selects and keeps track of

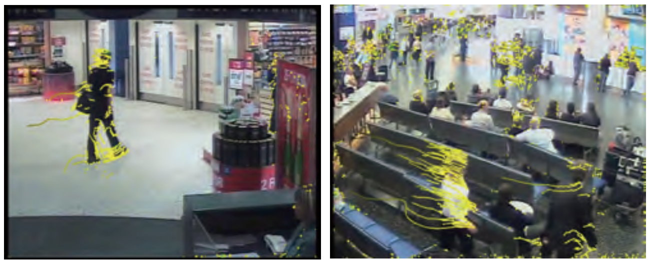

Figure 5: Key-point detection and tracking.

feature points that are optimal for tracking. It is widely used in visual feature tracking.

In the proposed system, the 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 processing. Then, only feature points in the foreground regions are detected.

The detected key-points are tracked by calculating the optical flow based on the Lucas-Kanade method. The key-points are tracked until the feature point disappears, as shown in Figure 5.

3.3 Feature extraction

3.3.1 Trajectory feature

Key-point trajectories convey critical temporal information on human behaviors. However, the feature is unsuitable for direct use in the bag-of-features approach because the 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. For this purpose, we used a method inspired by Mezaris, *et al.* [15].

To extract motion information at different time-scales, we used a hierarchic smoothing filter for each raw trajectory datum. A Haar low-pass filter was used for the smoothing. Families of trajectories were generated by applying the filter in incremental steps.

The node points in a trajectory are calculated by following equation. 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. The y-axis elements of the trajectory are calculated similarly.

$$\boldsymbol{P}_{u,q} = [\boldsymbol{p}^x_{u,q}, \boldsymbol{p}^y_{u,q}] \tag{4}$$

$$\boldsymbol{p}^x_{u,q} = [\boldsymbol{p}^{x,t_1+2^q-1}_{u,q}, \boldsymbol{p}^{x,t_1+2^q}_{u,q}, \boldsymbol{p}^{x,t_1+2^q+1}_{u,q}, ..., \boldsymbol{p}^{x,t_2}_{u,q}] \tag{5}$$

$$\boldsymbol{p}^{x,t}_{u,q} = \frac{1}{2^q}\sum_{i=0}^{2^q-1} \boldsymbol{p}^{x,t-i}_{u,0} \tag{6}$$

The histogram of motion directions at the granularity level θ is defined as a histogram of $2\pi/\theta$ bins: $[0, \theta)$, $[\theta, 2\theta)$,...,$[2\pi - \theta, 2\pi)$. The value of each bin depicts the number of elementary motions $[\boldsymbol{p}^{x,t}_{u,q}, \boldsymbol{p}^{y,t}_{u,q}]$ in the trajectory, normalized by division with the overall number of such elementary motions that belong to the examined trajectory. $\boldsymbol{a}(\boldsymbol{P}_{u,q}, \theta)$ is defined as the vector of all bin values for a given $\boldsymbol{P}_{u,q}$ and a constant θ.

Finally, different time-scale key-point trajectories can be represented as a fixed-length vector $\boldsymbol{A}_u$ as depicted in Equation (7), where R is the number of granularity levels.

$$\begin{aligned}\boldsymbol{A}_u = \Big[&\boldsymbol{a}\left(\boldsymbol{P}_{u,0}, \frac{\pi}{2}\right), \boldsymbol{a}\left(\boldsymbol{P}_{u,1}, \frac{\pi}{2}\right), ..., \boldsymbol{a}\left(\boldsymbol{P}_{u,Q-1}, \frac{\pi}{2}\right),\\ &\boldsymbol{a}\left(\boldsymbol{P}_{u,0}, \frac{\pi}{4}\right), \boldsymbol{a}\left(\boldsymbol{P}_{u,1}, \frac{\pi}{4}\right), ..., \boldsymbol{a}\left(\boldsymbol{P}_{u,Q-1}, \frac{\pi}{4}\right), ...,\\ &\boldsymbol{a}\left(\boldsymbol{P}_{u,0}, \frac{\pi}{2^R}\right), \boldsymbol{a}\left(\boldsymbol{P}_{u,1}, \frac{\pi}{2^R}\right), ..., \boldsymbol{a}\left(\boldsymbol{P}_{u,Q-1}, \frac{\pi}{2^R}\right)\Big]\end{aligned} \tag{7}$$

The above-mentioned feature is related to the trajectory's direction. The feature vector related to the speed (horizontal and vertical) is calculated in the same way. These histograms were normalized with the maximum and minimum lengths of the motion vector in each smoothing level's trajectory.

Figure 6 shows the concept of the feature vector creation from a trajectory. The line in the upper figure denotes a sample of a raw key-point trajectory ($\boldsymbol{P}_{u,0}$). The dash line denotes a one-time Haar filtered trajectory ($\boldsymbol{P}_{u,1}$) while the dotted line denotes a two-time Haar filtered trajectory ($\boldsymbol{P}_{u,2}$).

The histogram is created by counting the directions of motion vectors that are in the bins for each smoothing level's trajectory. The range of bins has several granularity levels, increasing the granularity level from 1 to R.

Finally, these histograms are combined into one. We used the numbers $Q = 3$ and $R = 3$ in this system, making the dimension of a histogram 84 (4 bins $\times$ 3 levels + 8 $\times$ 3 + 16 $\times$ 3). We also used histograms of both horizontal and vertical speeds, so the total dimension of a trajectory feature descriptor is 252 (84 $\times$ 3).

3.3.2 SURF feature

The above-mentioned histogram is a descriptor of the motion of a key-point. The system also extracts the appearance feature by calculating the SURF descriptor at the end point of the tracking [7]. The SURF is the 128-dimensional fixed-length descriptor that contains gradient information as the SIFT descriptor [6]. Figure 7 shows the track-ended key-point trajectories in a frame.

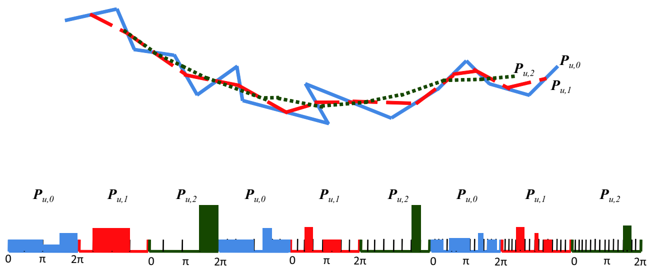

Figure 6: Concept of histogram creation of trajectory feature.

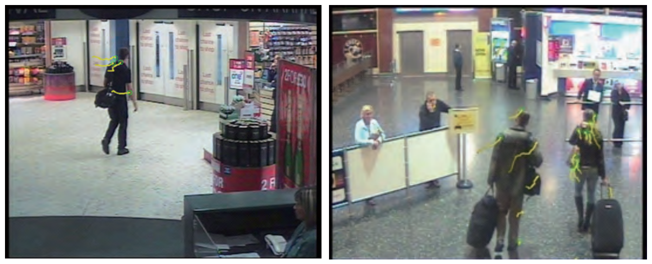
Figure 7: Key-point trajectory.

The system combines the 252-dimensional trajectory feature histogram with the 128-dimensional SURF descriptor, meaning that in total it uses a 380-dimensional motion-appearance feature.

3.4 Classification

An event is classified based on the bag-of-features approach. A certain section of video sequence is treated as the bag, and a 380-dimensional motion-appearance feature descriptor in the sequence is treated as the feature.

Each feature is labeled as the nearest cluster by referring to a code book. We made the code book beforehand based on the k-means method using a large amount of feature descriptor samples. The final feature vector as an input for the SVM classifier is represented as a cluster histogram by counting the feature descriptors that were labeled for each cluster.

When the cluster histogram is made, the system assigns a weight to each feature descriptor. The tf-idf approach is used to determine these weights.

The weight is calculated by multiplying a tf value by an idf value, as depicted in Equation (8). The idf value means the inverse sequence frequency, so the weight of a feature in a usual sequence decreases, as depicted in Equation (9). Note that N is the number of total sequences and n_x is the number of the sequence that includes cluster x.

The tf value means a feature frequency in a specified sequence, so the weight of a major feature in a sequence increases, as depicted in Equations (10) and (11). Note that oc_{xd} is the number of cluster *times* in sequence d and W is the set of features in sequence d.

Therefore, the system reduces the importance of the noise trajectory features that were constantly extracted around the background regions in the regular no-event sequences. In addition, it emphasizes the importance of the human motion trajectory features that were extracted in the event sequences.

$$FeatureWeight_{xd} = tf_{xd} \times idf_x \tag{8}$$

$$idf_x = \log \frac{N}{n_x} \tag{9}$$

$$tf_{xd} = \frac{ptf_{xd}}{\sqrt{\sum_{i \in W} ptf_{id}^2}} \tag{10}$$

$$ptf_{xd} = 0.5 + 0.5 \times \frac{oc_{xd}}{\max_{i \in W} oc_{id}} \tag{11}$$

An event classifier is created by training with the cluster histograms of each event. The SVM-supervised machine learning method [17] is used for training the classifier.

In the test phase, the system classified the events every frame by evaluating a cluster histogram that was made with trajectory features in a certain period of time.

3.5 Results

3.5.1 Parameters

We trained the system based on the following settings. We set the time-length of a sequence to one second (one sequence=25 frames). The length was determined by referencing the average duration of three target events.

We set the numbers $Q = 3$ and $R = 3$ as the smoothing and granularity levels for the parameters of the trajectory feature descriptor.

The SVM classifier was trained with cluster histograms that were made 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 three target events. Four classifiers were created for four cameras (except Cam4). The number of cluster k was set to 1,000.

The decision score for each event is determined based on a detected frequency in the duration of the detected event. If the decision score is beyond a specific threshold, the system considers the event to already have occurred.

3.5.2 Results and discussion

The results of our system are depicted in Table 5. The result of the "ObjectPut" event was better than the others, probably due to the directional and temporal simplicity of the motion. The downward 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. This result was also better than our result last year because of the

Table 5: Results for detecting events

Event	#Ref	#Sys	#Cor Det	Act. DCR	Min. DCR
ObjectPut	621	1061	39	1.113	1.002
Pointing	1063	3495	54	1.239	1.003
CellToEar	194	52	1	1.008	1.000

many key-point trajectories used as features that were suitable for detecting small motions.

The system often identified the "Pointing" event incorrectly, probably because that event has significant direction variations. For example, someone may point in the left direction and another may point in the right direction, etc. The accuracy might be improved by training the system to differentiate between "left pointing" and "right pointing."

Only one "CellToEar" event was correctly detected by the system. This event was difficult to detect because the motion was too small to extract sufficient trajectory lengths and there were not as many event samples in the development dataset as the other two events. The performance might be improved by training the classifier with more "CellToEar" event samples (ten days of samples from the development dataset should be sufficient).

4 Conclusion

We submitted methods for the semantic indexing task and the surveillance event detection task at TRECVID 2010.

The method proposed for the semantic indexing task used both gradient features in the local region around feature points and global features such as texture and color distribution. It also used a low-computational-cost method (random forests) as the classification algorithm. It obtained a mean infAP score of 0.0349 in the evaluation experiments. In future research, we plan to examine the use of audio features in addition to image features to improve precision even further and to study techniques that have both high precision and low computational cost.

For the surveillance event detection task, we developed a system of automatically detecting specific events ("Pointing," "CellToEar," and "ObjectPut") in video sequences shot by fixed cameras installed in an airport. Our system extracts fixed-dimensional features from a key-point trajectory and identifies the events based on the bag-of-features approach. The feature weights of the trajectory feature descriptors were calculated and used in the event classification. The system could robustly detect small motions, especially in the "ObjectPut" event. We plan to apply this framework to other human motion recognition systems, such as man-machine interfaces, that manage small motions.

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] "TREC Video Retrieval Evaluation Notebook Papers and Slides," http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html

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

[5] M. Takahashi, Y. Kawai, M. Fujii, M. Shibata, N. Babaguchi and S. Satoh, "NHK STRL at TRECVID 2009: surveillance event detection and high-level feature extraction," TRECVID 2009 workshop, 2009. http://www-nlpir.nist.gov/projects/tvpubs/tv9.papers/nhkstrl.pdf

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

[7] 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.

[8] Y.-G. Jiang, C.-W. Hgo and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," In Proc. ACM CIVR'07, 2007.

[9] T. Ojala M. Pietikaninen and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.24, no.7, pp.971–987, 2002.

[10] 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.

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

[12] E. Yilmaz and J.A. Aslam,"Estimating average precision with incomplete and imperfect judgments," In Proc. ACM CIKM'06, 2006.

[13] J.C. Niebles and L. Fei-Fei, "A hierarchical model of shape and appearance for human action classification," In Proc. IEEE CVPR'07, pp.1–8, 2007.

[14] P. Matikainen, M. Hebert and R. Sukthankar, "2009 Trajectons: action recognition through the motion analysis of tracked features," In Proc. ICCV Workshop on Video-Oriented Object and Event Classification, 2009.

[15] V. Mezaris, A. Dimou and I. Kompatsiaris, "Local invariant feature tracks for high-level video feature extraction," In Proc. WIAMIS'10, 2010.

[16] T. Kurita and T. Chikayama, "Classification precision of several candidates using multi-class support vector machine in generic object recognition," Technical report of IEICE, Multimedia and virtual environment vol.108, no.328, pp.251–258, 2008 [in Japanese].

[17] B. Scholkopf, J.C. Platt, J.Shawe-Taylor, A.J. Smola and R.C. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol.13, pp.1443–1471, 2001.

[18] J. Shi and C. Tomasi, "Good features to track," In Proc. IEEE CVPR'94, pp.593–600. 1994.