NHK STRL at TRECVID 2008: High-Level Feature Extraction and Surveillance Event Detection

Yoshihiko Kawai [†‡] Masaki Takahashi [†] Masanori Sano [†] Mahito Fujii [†]
Masahiro Shibata [†] Nobuyuki Yagi [†] Noboru Babaguchi [‡]

[†]Science and Technical Research Laboratories, NHK
1–10–11 Kinuta, Setagaya–ku, Tokyo, Japan
[‡]Osaka University 2–1 Yamadaoka, Suita–shi, Osaka, Japan

1 Introduction

NHK Science and Technical Research Laboratories participated in three tasks at TRECVID 2008: the high-level feature extraction task, surveillance event detection task and rushes summarization task. For the high-level feature extraction task, we used a method based on texture features within blocks of the key frames. The method attempts to reduce the effect of differences in the size or position of objects by using several block sizes. We used an ensemble learning algorithm called random forests to classify the key frames. For surveillance event detection tasks, we targeted three events: "Person Runs," "Opposing Flow," and "Elevator No Entry". The proposed method detected person regions based on differences in the direction and length of motion vectors and computed several features for each of the detected person regions. The random forests method was used to determine whether the event occurred or not. For the rushes summarization task, we used a novel method for selecting representative frames to remove redundancy. From a broadcaster's point of view, removing redundancy and attaining a pleasant tempo/rhythm, as well as a good recall ratio, are important. The method extracted representative frames from each shot based on the motion vectors and compared the extracted frames to remove shots which contain duplicate scenes such as retakes.

This paper is organized as follows. Section **2** gives a detailed description of the high-level feature extraction method and the experimental results for it. Section **3** describes the surveillance event detection method and the experimental results for it. We conclude in Section **4**. The rushes summarization method was described in detail at the ACM multimedia workshop [1]; thus, this paper has no section about it.

2 High-level feature extraction

Figure 1 shows an overview of the method. First, the method detects shot boundaries in the input video and extracts frames at mid-point of each shot as key frames. Then, each key frame is divided into blocks, and four

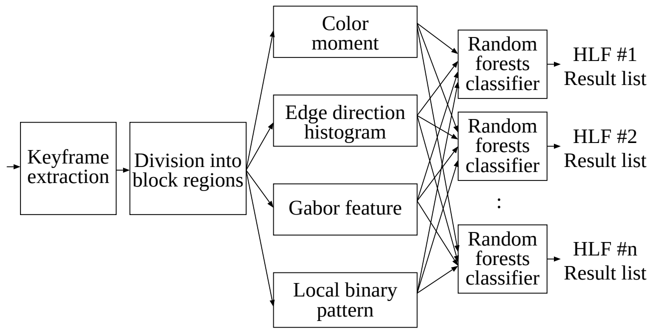

Figure 1: Overview of high-level feature extraction.

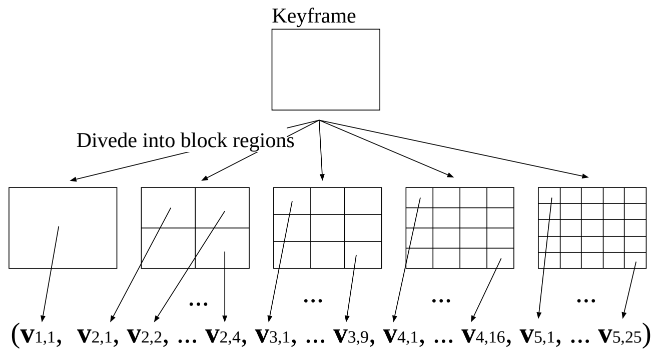
Keyframe
Divede into block regions
(v1,1, v2,1, v2,2, ... v2,4, v3,1, ... v3,9, v4,1, ... v4,16, v5,1, ... v5,25)

Figure 2: Feature vector calculation for a frame.

different texture features are computed for each block. The random forests method [2] based on the texture features is used to determine whether the frame image has a specific high-level feature or not. The random forests classifier is trained for each type of high-level feature. The output results list ranks frames in order of the index value from the decision trees of random forests.

2.1 Texture features

The method divides a key frame into block regions, and calculates texture features for each block to get a feature vector. To decrease the effect of differences in size or position of objects in the frame, a combination of block sizes from 1×1 to 5×5 are used. Figure 2 shows the method for calculating feature vector for a frame. The sign $\mathbf{v}$ represents the texture feature vector for each block, and all of them are concatenated to form the feature vector for the entire frame. Four types of texture feature that have been demonstrated to be effective in earlier research on TRECVID [3] were used: the color-moment feature, the edge-direction histogram, the Gabor feature and the local binary patterns [4].

2.1.1 Color moment feature

The color moment feature expresses the distribution of color within the region. The method transforms the input frame into the HSV color space and the Lab color space and computes the mean μ, standard deviation σ, and the cube-root of skewness s, of pixel values for each color component. Equations for these computations are given below.

$$\mu_c = \frac{1}{HW}\sum_x\sum_y f_c(x,y) \tag{1}$$

$$\sigma_c = \left\{\frac{1}{HW}\sum_x\sum_y \{f_c(x,y)-\mu_c\}^2\right\}^{1/2} \tag{2}$$

$$s_c = \left\{\frac{1}{HW}\sum_x\sum_y \{f_c(x,y)-\mu_c\}^3\right\}^{1/3} \tag{3}$$

Here, $f_c(x,y)$ represents the pixel value of the color component c ($c \in \{h,s,v,l,a,b\}$) at the coordinate position (x,y), and H and W are the height and width of the block. The values μ_c, σ_c, s_c are calculated for each component and concatenated to form the feature value.

2.1.2 Edge-direction histogram

The edge-direction histogram represents the distribution of the edge orientation in the block. Our method creates a frequency histogram with 37 bins: 36 bins for edge direction quantized at five degrees in the range of -90 to +90 degrees and one bin for non-edge points. Each element in the histogram is normalized to the number of pixels in the block. A Sobel filter is used to detect edges.

2.1.3 Gabor feature

The Gabor feature expresses the frequency and direction of gray-scale values in a small region. A Gabor filter $g_{m,n}$ with resolution m and direction n is given by Equation (4).

$$g_{m,n}(x,y) = \frac{k_m^2}{\sigma^2}\exp\left\{-\frac{k_m^2(x^2+y^2)}{2\sigma^2}\right\} \times \left[\exp\{jk_m(x\cos\theta_n + y\sin\theta_n)\} - \exp\left(-\frac{\sigma^2}{2}\right)\right] \tag{4}$$

k_m and θ_n are defined as follows.

$$\begin{cases} k_m = a^m & (0 \le m \le S-1) \\ \theta_n = \dfrac{n\pi}{K} & (0 \le n \le K-1) \end{cases} \tag{5}$$

K is the number of directions, S is the number of resolutions, and a is the magnification. The Gabor filter in Equation (4) is convolved with the block region, and the mean and standard deviation of these values is taken as the feature value. The convolution for the pixel at (x,y) is shown in Equation (6).

$$w_{m,n}(x,y) = \sum_i\sum_j f_g(i,j)\cdot g_{m,n}(i-x, j-y) \tag{6}$$

Here, $f_g(x,y)$ represents the monochrome value for the pixel at (x,y). The feature value based on the convolution result is computed using Equations (7) and (8).

$$\mu^g_{m,n} = \frac{1}{HW}\sum_x\sum_y w_{m,n}(x,y) \tag{7}$$

$$\sigma^g_{m,n} = \left\{\frac{1}{HW}\sum_x\sum_y \{w_{m,n}(x,y)-\mu^g_{m,n}\}^2\right\}^{1/2} \tag{8}$$

We use $S=4, K=6$ for a total of 24 Gabor filters as in [5], and create a feature vector by concatenating the values, $\mu^g_{m,n}$ and $\sigma^g_{m,n}$ for each filter.

2.1.4 Local binary patterns

The local binary patterns [4] express the patterns in relative brightness of surrounding pixels, relative to a target pixel. Equation (9) shows how the local binary pattern $L_{P,R}$ is computed from P pixels on a circumference of radius R around the pixel (x,y).

$$L_{P,R}(x,y) = \begin{cases} \sum_{p=0}^{P-1}\delta_{P,R}(x_p,y_p), & \mathit{if}\ U_{P,R}(x,y) \le 2 \\ P+1, & \mathit{otherwise} \end{cases} \tag{9}$$

Here, $\delta_{P,R}$ expresses the relative brightness between the target pixel (x,y) and a neighboring pixel $(x+x_p, y+y_p)$ and is calculated according to Equation (10).

$$\delta_{P,R}(x_p,y_p) = \begin{cases} 1, & f(x+x_p, y+y_p) - f(x,y) \ge 0 \\ 0, & \mathit{otherwise} \end{cases} \tag{10}$$

x_p and y_p are calculated as in Equation (11).

$$\begin{cases} x_p = R\cos\dfrac{2\pi p}{P} \\ y_p = R\sin\dfrac{2\pi p}{P} \end{cases} \quad (0 \le p \le P-1) \tag{11}$$

$U_{P,R}$ in Equation (9) gives the number of times $\delta_{P,R}$ changes from 0 to 1 or vice-versa; it is computed using Equation (12).

$$U_{P,R}(x,y) = |\delta_{P,R}(x_{P-1}, y_{P-1})| + \sum_{p=1}^{P-1} |\delta_{P,R}(x_p, y_p) - \delta_{P,R}(x_{p-1}, y_{p-1})| \quad (12)$$

$L_{P,R}(0 \leq L_{P,R} \leq P+1)$ from Equation (9) is computed for all pixels in the block region, and a frequency histogram of these values is used as the feature value. Our method computes $L_{P,R}$ frequency histograms for three parameter sets [4], $(P,R) = (8,1)$, $(16,2)$, and $(24,3)$ for scale invariance and concatenates the histograms to obtain the feature value.

2.2 Random forests method

The random forests method [2] is used to determine whether an input key frame has a specific high-level feature. Random forests is a kind of ensemble learning, and it gives highly accurate classifications by using a combination of decision trees (CART) [6]. 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 (having and not-having the high-level feature) 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 [2] attempts to resolve the problem by applying a higher weight to the smaller class. 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 devise 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 capable of a high level of generalization.

2.3 Experiments

2.3.1 Video data

The training data was about 100 hours of video data from TRECVID 2007. It was supplied by the Netherlands Institute for Sound and Vision [7]. The content is mainly documentary and educational programs and includes both color and black-and-white video. The frame size is 352×288 pixels, the frame rate is 25 fps, and the video is in MPEG-1 format.

Table 1: Settings of each run.

Run ID		Trees	Frame division
1	NHKSTRL1	500	1×1〜5×5 blocks
2	NHKSTRL2	250	1×1〜5×5 blocks
3	NHKSTRL3	1000	1×1〜5×5 blocks
4	NHKSTRL4	500	1×1 block
5	NHKSTRL5	500	3×3 blocks
6	NHKSTRL6	500	5×5 blocks

Table 2: Experimental result of each run.

Run	Recall	Precision	F-value
1	14.4%	3.2%	0.052
2	14.1%	3.2%	0.051
3	14.5%	3.3%	0.053
4	12.4%	2.9%	0.046
5	13.8%	3.0%	0.049
6	11.3%	2.7%	0.044

Table 3: Mean infAP of each run.

Run	mean InfAP
1	1.3%
2	1.2%
3	1.3%
4	0.7%
5	1.2%
6	1.1%

We manually assigned annotation data to the key frames. The system training type is classified into category B (only on the common dev. collection, not on the common annotation). We also assigned an annotation to all key frames extracted from TRECVID 2008 test data to evaluate recall. The training data consisted of approximately 33,000 key frames, while the test data was about 36,000 frames. The method from [8] was used to detect shot boundaries for extracting key frames.

2.3.2 Experimental results

The six different parameter settings shown in Table 1 were used for the experiment. The total number of decision trees used in the random forests classifier was varied in runs 1 to 3, and a single block size was used in runs 4 to 6. We evaluated the average recall and precision for the 20 types of high-level features. Table 2 lists the results. Recall was in the range of 10 to 15%, and precision was around 3%. Runs 1 to 3, which used five different block sizes, resulted in higher values for recall and precision than did runs 4 to 6. These results indicate that the effect of changes such as camera size can be reduced by using several block sizes. Runs 1 to 3 did not show any conclusive difference because of the number of decision trees, but the F-value for run 3 was slightly higher than for runs 1 and 2. The average recall in runs 4 to 6, where only a single block size was used, were 1 to 2% lower than those of runs 1 to 3.

Table 4: Results of high-level feature extraction (Run 3).

HLF	Recall	Precision
01	8% (23/ 297)	1% (23/2000)
02	7% (20/ 267)	1% (20/2000)
03	6% (5/ 80)	0% (5/2000)
04	18% (18/ 99)	1% (18/2000)
05	8% (21/ 252)	1% (21/2000)
06	13% (10/ 75)	1% (10/2000)
07	6% (196/3045)	10% (196/2000)
08	10% (7/ 70)	0% (7/2000)
09	25% (60/ 241)	3% (60/2000)
10	17% (12/ 72)	1% (12/2000)
11	29% (24/ 83)	1% (24/2000)
12	7% (37/ 509)	2% (37/2000)
13	11% (264/2460)	13% (264/2000)
14	14% (23/ 164)	1% (23/2000)
15	8% (113/1345)	6% (113/2000)
16	20% (79/ 390)	4% (79/2000)
17	29% (141/ 487)	7% (141/2000)
18	24% (84/ 351)	4% (84/2000)
19	14% (86/ 600)	4% (86/2000)
20	15% (71/ 486)	4% (71/2000)

Table 3 shows evaluation results for the mean infAP (inferred average precision) [9]. The evaluation was performed by NIST. The mean infAP values were in the range from 0.7 to 1.3%. The resulting precisions were lower than our results in Table 2, because our system was trained using the original annotation data, which were different from the annotation data used in the evaluation by NIST.

We investigated the precision and recall for 20 types of high-level features obtained in run 3, which had the largest F-value. The results are shown in Table 4, and examples of detected frames are shown in Figure 3. The recall for features "9. Driver," "11. Harbor," and "17. Nighttime" were over 25%, which was high relative to the other features. Factors contributing to the high recall could be the texture of the water surface for "11. Harbor," and the brightness for "17. Nighttime". Regarding "9. Driver," most of the frames in both the training data and the test data have similar camera angles, which could have increased the recall. Features "16. Mountain," and "18. Boat ship" also achieved recalls over 20%. Conversely, "1. Classroom," "2. Bridge," "3. Emergency vehicle," "5. Kitchen," "7. Two People," "12. Telephone," and "15. Hand," all had low recalls (under 10%). These high-level features encompass a variety of camera angles, object sizes and types, and the texture features used in our method are not adequate to identify them. It is necessary to add other features such as physical shape or feature points. Temporal features also may be useful. Another reason for the low recall is that there were too few positive examples in the training data. Note that for "7. Two people" it may be useful to consider body or face detection algorithms, but this would involve feature values tailored to a specific high-level feature, which would need to be studied in light of the generality of the algorithm.

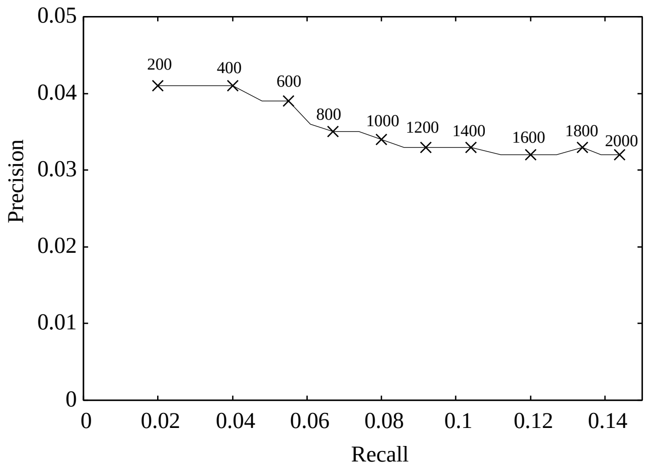

Figure 4: Recall–precision graph (Run 3).

We also studied the relationship between recall and precision. We examined the average recall and precision for all 20 high-level features while changing the number of selected top-ranked frames from 200 to 2000 frames, in increments of 200 frames. Figure 4 shows the recall–precision graph for run 3. The x-axis and the y-axis represent recall and precision respectively, and each axis is scaled to the range of the values. The recall was in the range from 1 to 14%, while the precision ranged between 3 and 4%. This confirms that increasing the number of selected frames results in an increase in recall.

3 Surveillance event detection

3.1 Overview

The surveillance event detection task involved automatically extracting sequences containing specific types of motion events within video from five surveillance cameras set in an airport.

In analyzing video for certain motions, we isolated regions containing people and then analyzed each of these regions. It can be very difficult to correctly differentiate between people within an airport because the traffic can be very busy and people frequently cross in front of one another. Color characteristics could be used to differentiate people, but people wear clothes with various colors, and two people wearing the same color would still be difficult to differentiate. Thus, we focus on differences in the direction of motion and speed of persons, by computing a motion vector from the optical flow and using it to segment the video image into regions for each person. We focus on "Person Runs", "Opposing Flow," and "Elevator No Entry," three events which seemed clearly related to a person's motion vector from the required events.

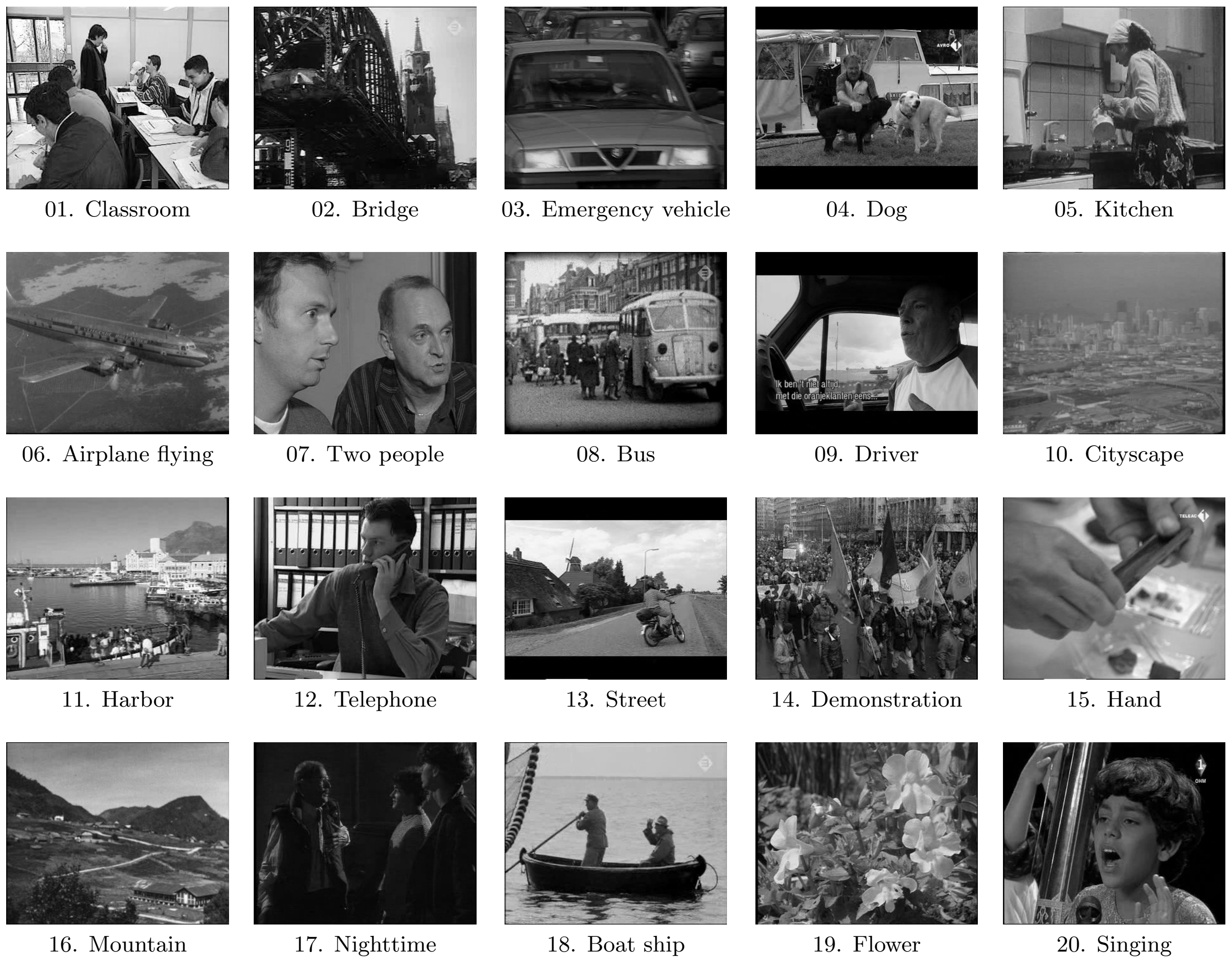

Figure 3: Example of extracted frames (Run 3).

Figure 5 shows the flowchart for our detection process. The optical flow for the development video is first computed, and this distribution is used to detect person regions. Several image features are computed for each of the detected person regions, and these feature values are used by the random forests machine learning algorithm to train classifiers for each event. To detect events, the evaluation video is first processed in the same way as the development video to extract feature values, and the results are input to each classifier to determine whether the event is present or not. If a given event is detected in more than a threshold number of frames within a fixed period of time, the event is determined to have occurred.

The details of each processing step are discussed in the following sections.

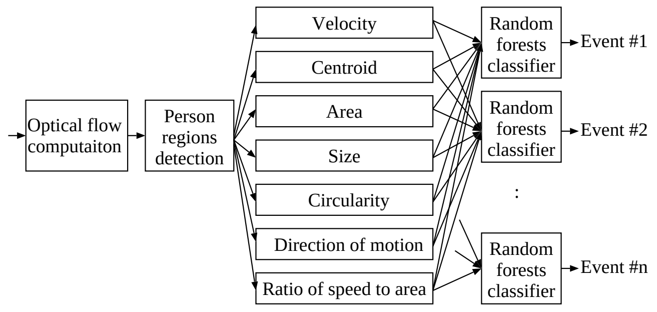

Figure 5: Overview of surveillance event detection.

3.2 Event recognition

3.2.1 Optical flow computation

The optical flow is a method for analyzing the motion of objects; it uses brightness information in the video and expresses the motion of objects as motion vectors. Optical flows appear in any region in the image containing a moving object. Methods for computing the optical flow can be broadly classified into gradient and block-matching methods. For this study, we used the gradient method of Lucas and Kanade [10], which has excellent computational efficiency.

The gradient method derives the relationship between

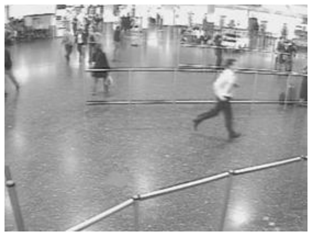
Figure 6: Video containing the "Person Runs" action.

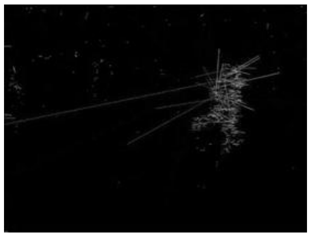
Figure 7: Optical flow.

the time-space differential and the optical flow, under the assumption that the brightness of an object does not change as it moves, and it uses this relationship to estimate the motion of objects in the video.

Let the brightness at a point (x, y) on the image at time t be $I(x, y, t)$. After a small amount of time Δt, the object will have moved by $(\Delta x, \Delta y)$. Assuming the brightness has not changed, we can write:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{13}$$

A Taylor expansion of the right side gives:

$$I(x,y,t) = I(x,y,t) + \Delta x\frac{\partial I}{\partial x} + \Delta y\frac{\partial I}{\partial y} + \Delta t\frac{\partial I}{\partial t} + e \tag{14}$$

Here, e represents the higher-order terms due to $\Delta x, \Delta y, \Delta t$, but if we ignore these, divide both sides by Δt and let $\Delta t \rightarrow 0$, we obtain the following relationship:

$$(\nabla \mathbf{I})^T \mathbf{u} + I_t = 0 \tag{15}$$

Note that $\nabla \mathbf{I} = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$, $\mathbf{u} = \left(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}\right)$ and $I_t = \frac{\partial I}{\partial t}$.

If we can assume that the optical flow $\mathbf{u}$, is the same for all points in a local region, we can estimate the $\mathbf{u}$ which best fits the above relationship in terms of the least-square error. That is, we minimize the least-square error in a local region $\mathbf{R}$:

$$E = \sum_{(x,y)\in\mathbf{R}} \left((\nabla \mathbf{I})^T \mathbf{u} + I_t\right)^2 \tag{16}$$

and this can be done by multiple linear regression.

The above process is carried out for the whole screen in order to obtain the optical flow for each frame. Figures 6 and 7 show a video sequence containing the "Person Runs" event and the optical flow computed from the sequence. The lengths of the lines in Figure 7 represent the motion vectors in the human region. It is clear that many motion vectors having the same direction and magnitude are generated in the region where the person is.

Regions containing people are extracted from fixed-camera video using differential processing against the background, but, depending on the camera, this extraction may be unreliable because changes in sunlight can create significant changes in the brightness of the background. The optical flow uses differences between adjacent frames, so changes in brightness or color over longer periods of time have no effect, and this allows person regions to be extracted more reliably.

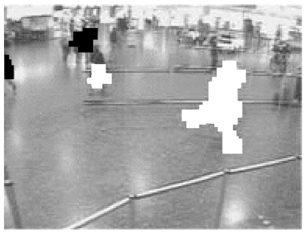
Figure 8: Detected person regions.

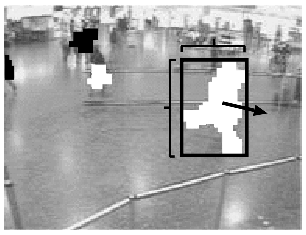
Figure 9: Samples of features in person regions.

3.2.2 Detecting person regions and computing feature values

Person regions are detected using the optical flow distribution. Neighboring pixels with similar motion vectors are aggregated and repeated to build up regions. Motion vectors tend to be different for each person, so it is possible to extract separate regions for each person in the video by using this method. The results of identifying the person regions from the optical flow in Figure 7 are shown in Figure 8. Different regions have been created for each person, and regions moving in different directions are shown in different colors.

As shown in Figure 9, feature values with the following 13 dimensions are computed for each segmented region on the basis of the image coordinates.

- Velocity $[pixel/frame]$ (horizontal v_x, vertical v_y, horizontal speed $|v_x|$, vertical speed $|v_y|$, overall speed $|v_{xy}| = \sqrt{v_x^2 + v_y^2}$)

- The position of the centroid of the region [*pixel*] (horizontal p_x, vertical p_y)
- The area [*pixel*] (total number of pixels S in the region)
- Width and height of the extracted region [*pixel*]
- Circularity, $e = \frac{4\pi S}{l^2}$ (l is the length of the perimeter of the region)
- Direction of motion [*degree*], $\theta = tan^{-1}\frac{v_x}{v_y}$
- Ratio of speed to area, $v_s = \frac{v_{xy}}{S}$

3.2.3 Creating classifiers using machine learning

The random forests classifiers were created to identify the specific actions using the 13-dimensional feature values computed as described above. The random forests method is resistant to training-data noise, and it can be used to compare feature values in terms of their significance. There are many other possible learning algorithms, such as a support vector machines (SVM) and AdaBoost, but we use the random forests method for this study because of its accuracy, speed, and significance capability.

When classifying video, each of the decision trees decides whether the event is present, and the event with the most votes, including "no event," is output. This determination is done for each frame, and if an event is detected in more than a certain number of frames within a set time period, it is determined that the event has occurred. For example, if the "Person Runs" event is detected in more than 50 frames within a ten-second period (250 frames), it is determined that a "Person Runs" event has occurred.

3.3 Results and Discussion

Each of the classifiers was applied to the evaluation video to detect each type of event. The results are listed in Table 5. There were no references to the "Elevator No Entry" event, so it could not be evaluated and is not included in the table.

The recalls were higher than the precisions for both the "Person Runs" and "Opposing Flow" events. The precision of the results were low, and this may be due to the fact that we trained with the wide range of correct data, and this resulted in more false alarms. In this study, we created a single classifier for each type of event, but we hope to reduce the incidence of false alarm in the future, by creating some classifiers for each type of event, such as for people in the foreground, the background, and for children.

Schemes for improving the recalls are also needed. Upon examination of the scenes not detected by the classifiers, we found that many of these scenes contained large numbers of people, with individuals frequently being obstructed by others. This may have resulted in incorrect feature values such as area, width, height, or circularity. In this study, we computed the optical flow image regions by using neighboring frames and then performed event-detection independently on these image regions, but in the future, we will need to consider analysis along the time dimension as well, computing transitions in parameters such as the area and shape and detecting whether an obstruction occurs.

Table 6: Comparison of classifier with single threshold detection.

	Recall	Precision
Training-based classifier	78.6%	69.0%
Single threshold detection	60.2%	53.4%

To verify the effect of training, we performed experiments comparing the learning-algorithm classifiers with single threshold decisions based on a speed feature value. Examining only video sequences containing the "Person Runs" events, we tested whether the regions containing the "Person Runs" event could be detected in sequences also containing many other people. Table 6 lists the results, with the classifiers yielding better results for both precision and recall. This confirms that the learning algorithm produced some positive effect.

Figures 10 and 11 compare the significances of the feature values, as computed by the random forests learning algorithm. For the "Person Runs" event, the feature values related to absolute speed were most important, followed by those related to area and position. This is because the speed for "Person Runs" is faster than for normal walking, but there is very little relation to the direction of motion. On the other hand, the feature values related to direction of motion and position were more important for the "Opposing Flow" events. This may have been because there were a limited number of positions and directions of motion for the applicable events.

The framework used here for extracting feature values and detecting specific events through learning can be expanded to include events besides those examined here. In the future, we plan to study this framework further with other event types.

4 Conclusion

This paper proposed a high-level feature extraction method and surveillance event detection method. For the high-level feature extraction, we calculated various texture features in block regions of key frames and classified the frames with the random forests algorithm. To reduce the effect of differences in the size or position of the object, we used several block sizes for computing the texture features. In experiments, the high-level feature extraction method was more accurate than a method using a single block size. In the future, we will attempt to increase the detection accuracy and study the use of other features such as feature points, physical shape and

Table 5: Detection results using the evaluation video.

	Reference	Correct	False	Miss	Recall	Precision
"Person Runs"	314	81	1382	233	25.80%	5.86%
"Opposing Flow"	12	1	54	11	8.33%	1.85%

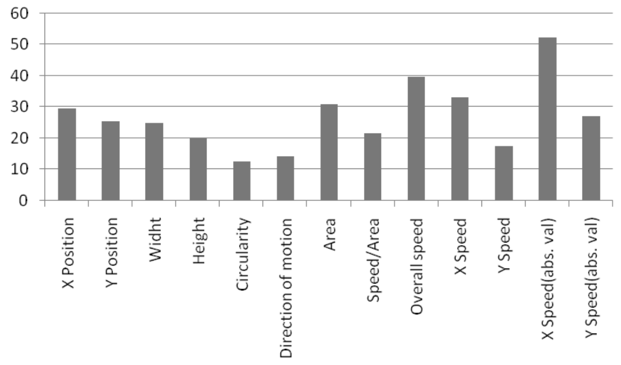

Figure 10: Significance of feature values for the "Person Runs" action.

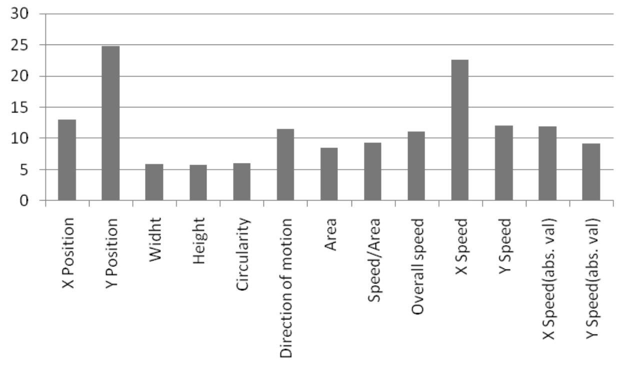

Figure 11: Significance of feature values for the "Opposing Flow" action.

positional relationships. We also plan to analyze temporal video features, such as motion vectors.

Regarding the surveillance event detection, we devised a method for automatically detecting specific events ("Person Runs," "Opposing Flow," and "Elevator No Entry") within video from fixed cameras set in an airport. We obtained motion vectors through the optical flow computations and aggregate pixels having the same vector into regions. We then decided whether these regions represent specific actions on the basis of feature values taken from the regions by using classifiers that had been trained to detect the actions. In this study, we developed classifiers for the above three actions, but our framework, consisting of the feature value extraction and learning methods, is applicable to other actions as well. We plan to expand the range of actions detected in the future.

References

[1] M. Sano, Y. Kawai, N, Yagi and S. Satoh, "Video rush summarization utilizing retake characteristics," TRECVID BBC Rushes Summarization Workshop at ACM Multimedia, 2008.

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

[3] TREC video retrieval evaluation, http://wwwnlpir.nist.gov/projects/trecvid/

[4] 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.

[5] B.S. Manjunath and W.Y. Ma, "Texture features for browsing and retrieval of image data," IEEE Trans. Pattern Analysis and Machine Intelligence, vol.18, no.8, pp.837–842, 1996.

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

[7] http://portal.beeldengeluid.nl/

[8] Y. Kawai, H. Sumiyoshi, N. Yagi, "Shot boundary detection at TRECVID 2007," In Proc. TRECVID Workshop, 2007.

[9] E. Yilmaz and J.A. Aslam,"Estimating average precision with incomplete and imperfect judgments," In Proc. ACM International Conference on Information and Knowledge Management, 2006.

[10] J-B. Shim, Y. Takeuchi, T. Mukai and N. Ohnishi, "Improving the point correspondence accuracy of Kanade-Lucas Method," ITE Technical Report, vol.26, no.30, pp 67–72, 2002 (Japanese).