Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2010, Article ID 801252, 14 pages
doi:10.1155/2010/801252

Research Article

Robust Recognition of Specific Human Behaviors in Crowded Surveillance Video Sequences

Masaki Takahashi,[1,2] Mahito Fujii,[1] Masahiro Shibata,[1] and Shin'ichi Satoh[2,3]

[1] *Human & Information Science Research Devision, Science and Technology Research Laboratories, Japan Broadcasting Corporation, Tokyo 157-8510, Japan*
[2] *Department of Informatics, The Graduate University for Advanced Studies, Kanagawa 240-0193, Japan*
[3] *Multimedia Information Research Devision, National Institute of Informatics, Tokyo 101-8430, Japan*

Correspondence should be addressed to Masaki Takahashi, takahashi.m-iu@nhk.or.jp

Received 12 November 2009; Accepted 16 March 2010

Academic Editor: ChangIck Kim

We describe a method that can detect specific human behaviors even in crowded surveillance video scenes. Our developed system recognizes specific behaviors based on the trajectories created by detecting and tracking people in a video. It detects people using an HOG descriptor and SVM classifier, and it tracks the regions by calculating the two-dimensional color histograms. Our system identifies several specific human behaviors, such as running and meeting, by analyzing the similarities to the reference trajectory of each behavior. Verification techniques such as backward tracking and calculating optical flows contributed to robust recognition. Comparative experiments showed that our system could track people more robustly than a baseline tracking algorithm even in crowded scenes. Our system precisely identified specific behaviors and achieved first place for detecting running people in the TRECVID 2009 Surveillance Event Detection Task.

1. Introduction

There have been many studies of human motion recognition through video content analysis. This work has gained widespread interest for both academic and industrial purposes [1]. These techniques can be applied to motion-based video searches or to man-machine interfaces that use human gestures, for example.

To achieve these goals, some fundamental technologies have to be established. For recognizing human motions, the capability to detect and track humans in video is essential. A human detector and tracker based on feature points as represented by the Kanade-Lucas-Tomasi Feature Tracker (KLT tracker) has been studied [2, 3]. Moreover, gradient-based features as represented by histograms of oriented gradient (HOG) are currently being used for human detection [4, 5].

In addition, the rapid spread of surveillance cameras has increased demand for cameras that can not only track people but can also automatically identify their specific motions. A number of technologies identify specific motions by detecting a particular image feature value that is unlike the other major features. For example, Shiraki et al. have developed a technology to detect a specific motion from a video sequence using cubic higher-order local auto correlation (CHLAC) image features [6]. However, many of these technologies assume relatively simple videos in which it is relatively easy to detect and track people. A great deal of work has been done analyzing human behavior in simpler datasets (KTH [7], Weizmann [8]) where the motions are performed in controlled situations [9–12]. The features used in these algorithms are corner points, optical flows, and shape. These are not enough available due to the occlusion, different lighting conditions, or varying object sizes. Practical algorithms that can be applied to complicated sequences such as surveillance video from train stations or airports are required [13, 14].

Although some studies have targeted crowded surveillance video sequences, they have been limited to tracking human objects or detecting the overall motion of a great

number of people [15–17]. No technology has been established that can robustly detect specific behaviors within crowded sequences in real videos.

We describe a method that can detect specific human behaviors, such as running and meeting, even within crowded sequences. Although tracking all human objects in a crowded scene is a difficult problem, it could be possible to detect specific human behaviors by searching for a certain unique feature value from major normal feature values. The trajectory of a moving person contains rich information about the person's behaviors, such as velocity or travel distance, so we used this trajectory for recognizing human behaviors. No previous technology has recognized human behaviors based on their trajectory in crowded scenes.

For tracking people in complicated sequences, we used a HOG and a support vector machine-(SVM-) based human detection algorithm [18, 19] that is known to be relatively robust. In addition, we used a Kalman filter based tracking algorithm that contributes to robust tracking, even with occlusion, by predicting the position of the person.

Our system recognizes specific behaviors by analyzing the similarities to the reference trajectory of each specific behavior in the trajectory feature space. The feature space was generated from ten-dimensional features that were extracted from a trajectory based on principal component analysis (PCA) [20]. Our system can sensitively detect particular behaviors from minimal evidence using the feature space. Though it sometimes incorrectly identifies nontargeted behaviors as the targeted behaviors, most of these so-called false detections are rejected by the verification process. This verification technique is one of the unique features of our system.

We compared our method with a baseline tracking algorithm using two different datasets: the KTH dataset and the TRECVID dataset [21]. The results of these comparative experiments showed that our method more robustly tracked people even in crowded scenes. In addition, the TRECVID 2009 surveillance event detection task showed that our system recognized several specific behaviors precisely and that it was highly effective.

We introduce conventional techniques in Section 2, describe our motion recognition method in Section 3, show results of several experiments in Section 4, and conclude in Section 5.

2. Conventional Techniques

Many researchers have studied human appearance and motion recognition in the field of computer vision [1]. In particular, studies detecting specific motions using surveillance cameras have increased with the number of crimes and instances of terrorism.

Most conventional techniques that recognize particular objects or human motions follow a two-step process: (1) cut out objects or human shapes precisely or calculate low-level image features from a video, (2) apply the cutout objects to detailed shape or motion models prepared beforehand [22–25]. These methods can track people or detect human motions with a low error rate and can precisely recognize even small motions, such as hand waving and jumping, by considering kinematic models or spatio-temporal images. To cut out human shapes, advanced background subtraction methods and contour definition methods have also been developed for precise segmentation of the human shape [26, 27].

However, most of this work has assumed simple situations in which large images of people appeared in front of a smooth background as in the KTH dataset [7] and the Weizmann dataset [8], as shown in Figures 1 and 2. Therefore, these techniques are limited to the target video sequences, and it is difficult to apply them to actual video sequences. An autosurveillance system that can detect specific behaviors in practice has not yet been developed.

The KTH motion sequences have been frequently used in motion recognition papers. The dataset consists of 2391 low resolution videos (160×120, 25 fps) showing six types of human motions each performed 4 times by 25 persons. The motions are walking, jogging, running, boxing, hand waving, and hand clapping. Only one person appears in each sequence, which is shot by an almost-fixed camera in front of a smooth background, as shown in Figure 1.

The Weizmann dataset is 90 low-resolution video sequences (180×144, 25 fps) showing nine different people, each performing 10 natural motions such as running, walking, skipping, jumping, and hand waving. Though this set contains somewhat more complicated situations such as occlusions, the sequences are controlled, with only one person per sequence, and they were shot by a fixed camera as shown in Figure 2.

In contrast, the TRECVID dataset contains video sequences of real situations. The dataset for the surveillance event detection task was shot in a crowded airport with five different angled surveillance cameras as shown in Figure 3. The video is in PAL format (720×576, 25 fps). It consists of 100 hours of video sequences for development and 44 hours of video sequences for evaluation.

The TRECVID is a workshop for evaluating information retrieval technologies using a common video corpora. It is organized by the US National Institute of Standards and Technology (NIST) [28]. Once a year, participants, such as research groups from universities and companies, are invited to the evaluation. The participants compete at tasks specified by the TRECVID. The submitted results are evaluated and compared by the TRECVID organizers. The principal tasks are high-level feature extraction, copy detection, and surveillance event detection.

The surveillance event detection is a task to detect sequences of specific human motions. The ten required specific motions are called PersonRuns, CellToEar, ObjectPut, PeopleMeet, PeopleSplitUp, Embrace, Pointing, ElevatorNoEntry, OpposingFlow, and TakePicture. Participants should output detection results for any three events from the required set. NIST annotated the videos with the correct human motion data.

We used complicated videos for detecting specific human behaviors using the TRECVID dataset. We devised a sensitive method for detecting human behaviors by evaluating the

Figure 1: Samples of KTH dataset.

Figure 2: Samples of Weizmann dataset.

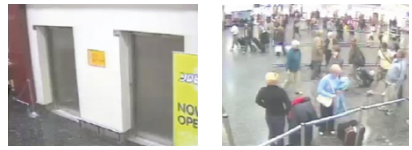

Figure 3: Samples of TRECVID dataset.

trajectory of a person. Though this method also misidentifies many nontargeted behaviors, most of these are then rejected during the verification process. This approach has not been studied in the past.

3. Proposed System

3.1. Overview. We propose a system that can recognize several human behaviors even in a crowded scene. We mainly focused on human behaviors associated with traveling, such as PersonRuns and PeopleMeet. Figure 4 shows the system configuration. We use the term *human region* to refer to any region in an image that contains one person.

In the training phase, the functions of human region detection and human behavior recognition are created from human region images and video sequences from fixed cameras. We describe human region detection in Section 3.2 and human behavior recognition in Section 3.4. In the operation phase, the system automatically detects specific behaviors from a video sequence shot by the same camera as the training phase. As input, we assume digitized video files that can be played repeatedly. The system outputs a file that has a time sequence of detected specific behaviors. The beginning time and ending time of the specific behaviors are written in the file. Though our system needed to be trained in advance, it could be applied to any surveillance video sequences without any camera calibration.

The method for recognizing specific human behaviors consists of four steps, as shown in Figure 5. Step 1 is human region detection. The system detects the regions that contain a human from the input image using HOG descriptors and an SVM classifier. Step 2 is human region tracking. The system tracks a region by evaluating a two-dimensional color histogram in the region. It robustly tracks human regions using a Kalman Filter. Step 3 is human behavior recognition. The system judges behaviors on the basis of the trajectory of the human region. It recognizes behaviors by analyzing the similarities to a reference trajectory of each specific behavior. Step 4 is human behavior verification. The system verifies the detected human behaviors by analyzing the followed trajectory or optical flows. In the following subsections, we describe the processing of each step in detail.

3.2. Human Region Detection

3.2.1. Processing Flow. The system follows the flow in Figure 6 to detect human regions. The preprocessing and postprocessing are performed before and after this human detection processing. The preprocessing (changed area detection) contributes to reducing the total processing cost and noise objects by limiting the area searched. The postprocessing (clustering) contributes to stabilizing the human region tracking.

3.2.2. Changed Area Detection. If the system searched for human regions in all ranges of the input image, the processing cost would be extremely high. In addition, the noise objects interrupt robust detection of human regions. Therefore, it searches for human regions only where there are frame differences or luminance differences from the background. Figure 7 shows a sample of background subtraction. The system can detect even a standing person by referring to the image. Frame differences are used for detecting the changed areas when a background subtraction image cannot be used because of the illumination change.

3.2.3. Human Detection Processing. Our human detector searches for human regions by calculating image features around the changed region. We used a human detector that combines the histograms of oriented gradient (HOG) feature descriptors and a support vector machine (SVM).

HOG descriptors are used for object detection. Many studies have reported that the HOG is suitable for detecting human regions because it is robust to a wide range of variations of poses [4, 5]. The technique counts occurrences of gradient orientation in localized portions of an image. It computes on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved performance.

We normalized all target regions at 25×50 pixels. The HOG descriptor is calculated by shifting a block by one cell. We defined a cell as 5×5 pixels and a block as 3×3 cells, so 81 (9 directions $\times$ 9 cells) dimensional histograms of gradient directions are obtained in a block. The cells and

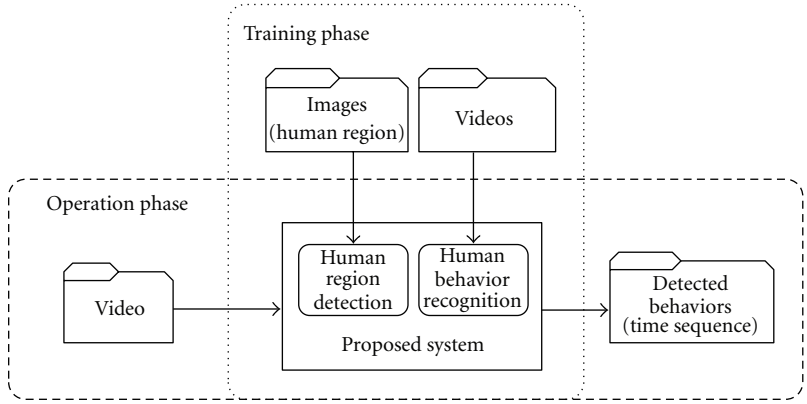

Figure 4: System configuration.

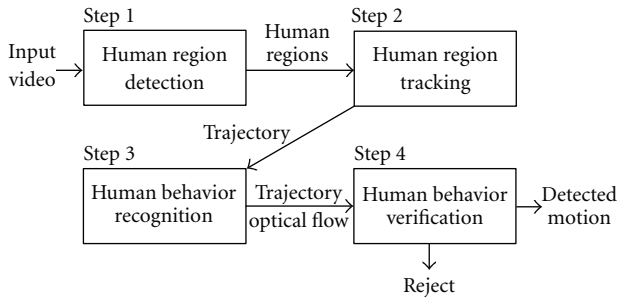

Figure 5: Flow for recognizing specific human behaviors.

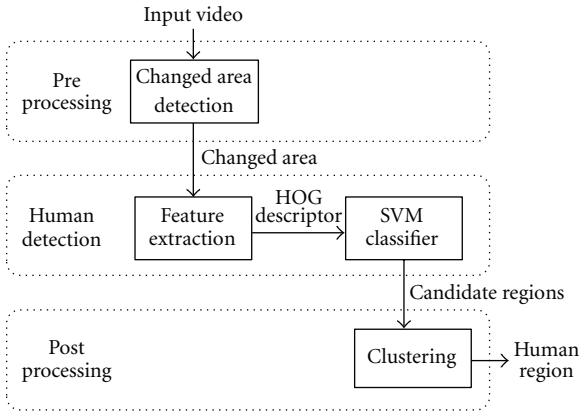

Figure 6: Flow for human region detection.

Figure 7: Detection of changed area.

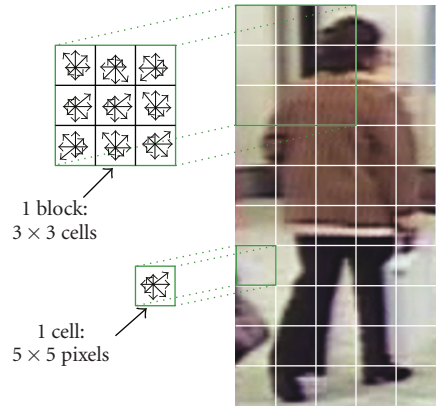

Figure 8: Cells and blocks.

blocks concept is shown in Figure 8. The normalization is performed for 24 blocks (3 horizontal × 8 vertical blocks) so that a 1,944-dimensional descriptor (24 blocks × 81 dimensions) is obtained for each target region.

The human detector (SVM classifier) uses the HOG descriptor to determine whether a target region contains a person. We trained the classifier with about 1,000 HOG descriptors in human regions (positive data) and about 2,000 HOG descriptors in other regions (negative data) for every camera of the TRECVID dataset. All sample data were extracted manually from the development datasets. The detected regions were treated as candidate human regions. A scheme for detecting candidate human regions is shown in Figure 9.

3.2.4. Clustering Human Regions. More than one candidate human region is usually detected around one person. The system should detect one human region for each person in order to track robustly and recognize human behaviors accurately. Therefore, the system clusters the candidate human regions and determines a representative human region for one person from each of the candidate human regions in each cluster. This function stabilizes the tracking of human regions.

The similarity of candidate human regions is evaluated by calculating two distances: (1) the simple Euclidean

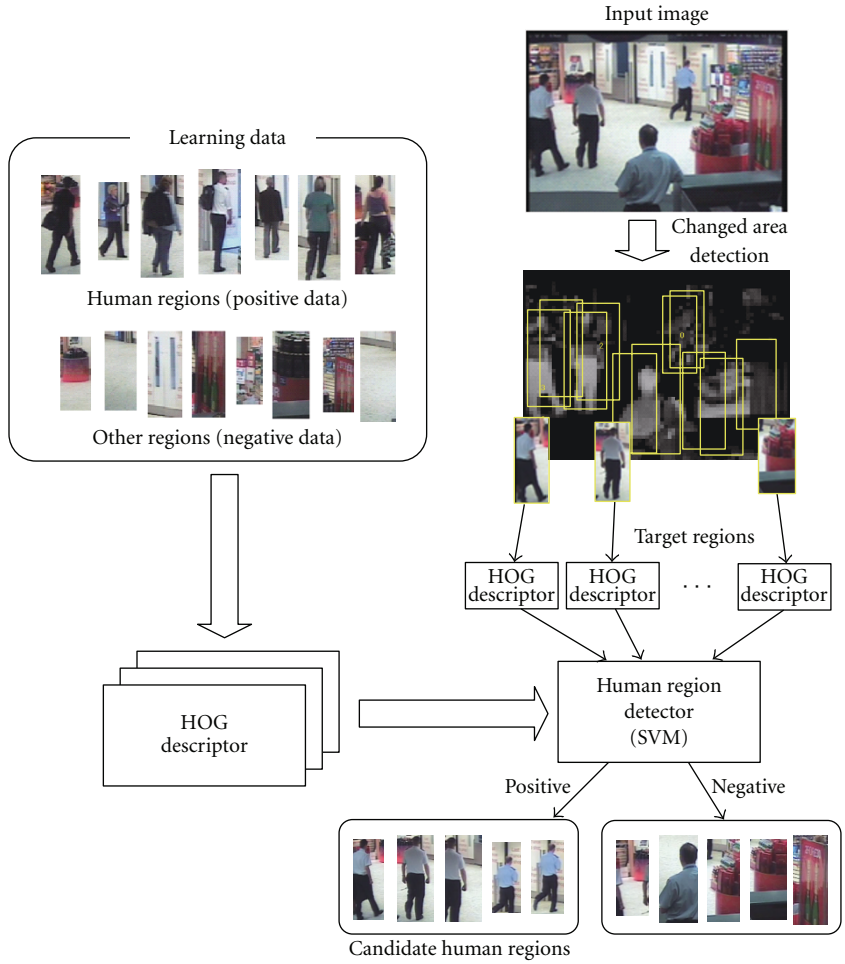

Figure 9: Detecting candidate human regions.

distance between the centers of gravity of two human regions and (2) the Bhattacharyya distance between two color histograms of candidate human regions [29]. First, the system calculates a simple distance d by (1). (x_1, y_1) and (x_2, y_2) represent the centers of gravity of two human regions in (1). After that, the distance of the color histogram is calculated only for nearby candidate human regions

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \quad (1)$$

We used HSV color space for creating color histograms. HSV stands for hue, saturation, and value. The hue and saturation are intimately related to the way human eye perceives color, so they are suitable as components of the histogram. The value is the brightness of the color. A histogram that is influenced by luminance should not be used because it changes depending on the positions of people. We used two-dimensional (2D) color histograms of HSV color space (H, S). The sum of a histogram is normalized to deal with the difference of the size of target regions.

Figure 10 shows samples of visualized 2D color histograms corresponding to the left target region. The horizontal axis shows hue, and the vertical axis shows saturation. The figure indicates that 2D color histograms differ according to colors in a target region.

The similarity of the 2D color histograms is calculated by Bhattacharyya distance, which is normally used to measure the separability of classes in classification. The distance is calculated from a Bhattacharyya coefficient that shows the similarity of two normalized histograms (3). The coefficient is calculated by (2) from normalized color histograms p and q. The ρ, m, and D_B denote the Bhattacharyya coefficient, the number of color components, and the Bhattacharyya distance

$$\rho = \sum_{x=0}^{m} \sqrt{p(x)q(x)}, \quad (2)$$

$$D_B = \sqrt{1 - \rho}. \quad (3)$$

A candidate human region located in the center of its cluster is selected as the representative human region of the cluster.

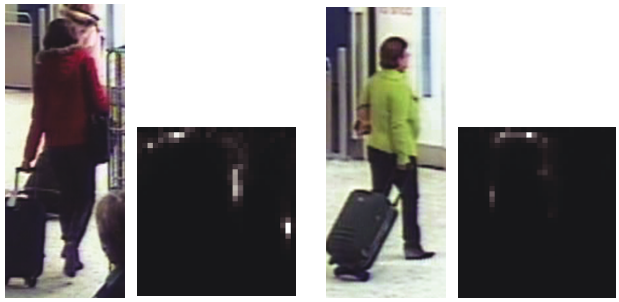

Figure 10: Target regions and 2D color histograms.

3.3. Human Region Tracking

3.3.1. Processing Flow. A human region is tracked following the flow of Figure 11. The system tracks human regions by searching for a similar human region in previous image frames. The similarity is evaluated by a simple Euclidean distance and a Bhattacharyya distance of 2D color histograms in the same way as the clustering process in human region detection.

If a similar human region is detected, the same ID as the past frame is set to the current human region. Otherwise, a new ID number is set. By connecting the center positions of the same ID region from past frames to the current frame, we can obtain the trajectory of the human region. Figure 12 shows sample trajectories of human regions.

It is difficult for a motion vector-based tracker to track all human regions in a crowded area because large numbers of the same motion vectors are detected. In our system, each human region can be tracked relatively robustly, even in a crowded scene, because of the 2D color histograms.

3.3.2. Prediction Processing. That approach, however, does not necessarily give the position of the human region in every frame. Detections can fail when an occlusion occurs in a crowded scene. Therefore, we supplemented the system with a prediction-based retrack function. The prediction continues even after a detection failure, so it is possible to pick up the human region again after it passes through an area where extraction is difficult.

A Kalman filter [30, 31], which assumes uniform straight-line motion, is used for prediction. It estimates the state of a linear dynamic system from a sequence of measurements that contain noise. It predicts the vectors **X**, **Y**, and **P** for each frame using

$$\begin{aligned} \mathbf{X}_t &= \mathbf{F}\mathbf{X}_{t-1} + \mathbf{v}_t, \\ \mathbf{Y}_t &= \mathbf{H}\mathbf{X}_t + \mathbf{w}_t, \\ \mathbf{P}_t &= \mathbf{F}\mathbf{P}_{t-1}\mathbf{F}^T + \mathbf{Q}_t. \end{aligned} \tag{4}$$

Here t is a frame number, and matrix $\mathbf{F}^T$ expresses the transposition of matrix **F**. $\mathbf{X} = [x, y, x', y']^T$ is the state estimation vector that consists of the position coordinates and speed, and **P** is the covariance matrix of the prediction error. **F** is the state propagator matrix whose elements express uniform straight-line motion for the linear prediction. **v** is a covariance matrix of the process noise ($p(\mathbf{v}) \sim \mathcal{N}(0, \mathbf{Q})$), and **w** is a covariance matrix of the observation noise ($p(\mathbf{w}) \sim \mathcal{N}(0, \mathbf{R})$).

Furthermore, if the system extracts a region, the process updates **X**, **P**, and the Kalman gain **K** using

$$\begin{aligned} \mathbf{X}' &= \mathbf{X} + \mathbf{K}(\mathbf{Y} - \mathbf{H}\mathbf{X}), \\ \mathbf{P}' &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}, \\ \mathbf{K} &= \mathbf{P}\mathbf{H}^T\left(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{R}\right)^{-1}. \end{aligned} \tag{5}$$

where $\mathbf{Y} = [p_x, p_y]^T$ is the measurement state (detected position coordinates), and $\mathbf{X}'$ and $\mathbf{P}'$ are updated values of **X** and **P**. **H** is a matrix that converts a state vector into a measurement vector. **R** is the covariance matrix of the measurement error. When the tracking goes wrong, the diagonal elements of **P** become large values, and when it succeeds, the elements become small. Therefore, the process controls the size of the image area searched for people on the basis of the values of **P**.

There usually exist many missing trajectories due to occlusion in crowded scenes. When a trajectory disappears, the system estimates the speed and moving direction of the missing person and predicts the future position of the person by calculating **X** frame by frame. The system matches the 2D color histogram information of the missing human region with any new human regions around the predicted position. If a match succeeds, the system connects the new detected position with the missed trajectory. This prediction processing contributes to robust tracking of human regions. Figure 13 shows a sample of retracking. Though two people crossed and an occlusion occurred at the center of the image, both people were extracted again and retracked after they separated.

3.4. Human Behavior Recognition

3.4.1. Normalization of the Length of Motion Vector. A trajectory of detected human regions contains rich information about human behavior, such as moving speed and travel distance. Our system recognizes specific human behaviors from their trajectories. However, the length of the motion vector differs depending on the detected position in the image coordinates. For example, the motion vectors of people who are detected at positions near the camera are large, and the motion vectors of people who are detected at a distance are small. To address this, we devised an average velocity map to normalize the motion vectors.

The average velocity map is created by calculating the average velocities of the small blocks in the image coordinates using (6). Motion vectors (v_x, v_y) at the center of gravity of a person about 170 cm tall were gathered automatically and the average velocity V_{ij} in each block was calculated. The variable n is the number of motion vectors gathered in block i, j. The concept of generating an average velocity map is shown in Figure 14. The lengths of motion vectors are accumulated

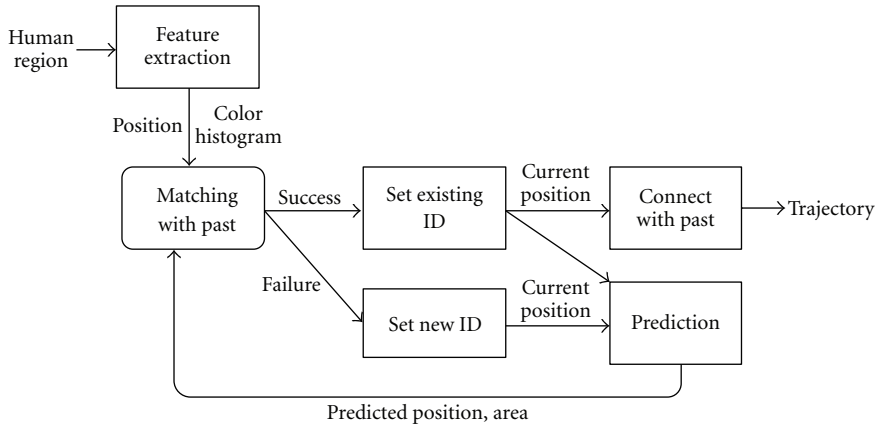

Figure 11: Flow for human region tracking.

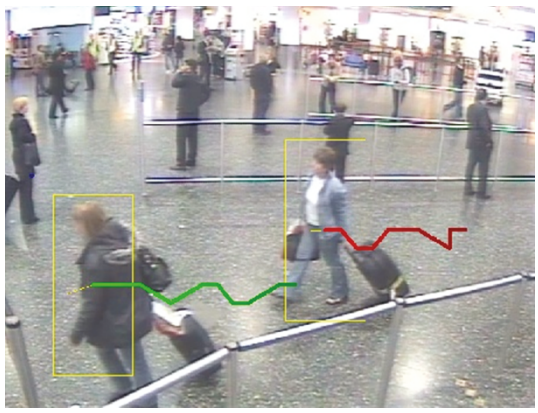

Figure 12: Samples of human region trajectories.

Figure 13: Retracking after occlusion.

in blocks, filled in dark in the figure. The average velocity for each block is calculated by averaging the length of all accumulated motion vectors during a certain period of time

$$V_{ij} = \frac{1}{n}\sum_{p=1}^{n}\sqrt{v_{x_p}^2 + v_{y_p}^2}. \quad (6)$$

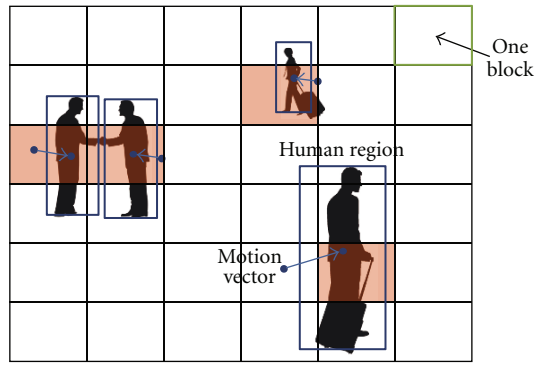

Figure 14: Concept of calculating an average velocity map.

We created average velocity maps from four hours of video sequences for each camera of the TRECVID dataset. A sample of an average velocity map is shown in Figure 15. In the figure, each number indicates the average velocity in the block. The values of the average velocities at the bottom are large because the motion vectors in the blocks were large. The values become smaller toward the top. The system found no human regions in any of the top row blocks, so the average velocities in these blocks are zero.

By referring to this average velocity map, the system can normalize an input motion vector, so it can treat all trajectories equally regardless of their detected position in the image. This technique does not need any calibration or positional information of the camera, so it can be applied to every video sequence shot by a fixed camera.

The system can normalize the motion vectors more accurately when the blocks are set to be small. However, the suitable block size depends on the resolution of the video, the distance from the camera to people, and the number of motion vectors used for training. If the resolution was low, the camera was near people, and the training sample vectors were poor, block size should be large, because the average velocity might be influenced by unusual motion vectors.

Figure 15: Sample of average velocity map.

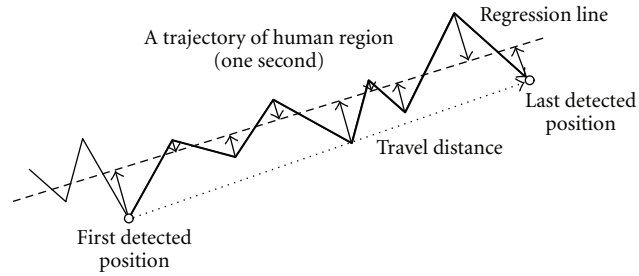

Figure 16: Features of a trajectory.

3.4.2. Feature Extraction from a Trajectory. We extracted the following ten-dimensional features from a trajectory.

(i) first detected position (x, y),

(ii) last detected position (x, y),

(iii) total vector (x, y),

(iv) travel distance,

(v) average velocity,

(vi) acceleration,

(vii) linearity.

The total vector is calculated by accumulation of the motion vectors of each frame in a trajectory. The feature indicates the direction of the trajectory. The system calculates the vector horizontally (x) and vertically (y). The travel distance means the distance from the first detected position to the last detected position. The distance is normalized in the same way as the motion vectors. The average velocity is the average length of motion vectors in a trajectory. The acceleration is a shift in velocity. We can calculate this feature by differentiating velocities. The system can detect a person who stopped or started suddenly with this feature. The linearity is the average distance from each detected position to a regression line. If a person has moved straight ahead, the linearity is close to zero.

If the system extracts these features from the full length of a trajectory, the features fade away by averaging them. Therefore, we divided the trajectories of an ID by every one second. Figure 16 shows the relations between a detected human trajectory and its features.

Generally, it is difficult to precisely estimate the moving speed and direction. However, the system can calculate these features as suitable for recognizing the specific behaviors from a human region trajectory that has been normalized with the average velocity map.

3.4.3. Detecting Behaviors from the Trajectory Feature Space. To cluster trajectories, we projected each trajectory onto a trajectory feature space that was generated using principal component analysis (PCA) [20]. We calculated eigenvalues and eigen vectors from the ten-dimensional features of each trajectory. We narrowed down the dimensions from ten to five. The first principal component was strongly influenced by the average velocity and the travel distance, the second one was influenced by the vertical position and the total vector, and the third one was influenced by the horizontal position, the total vector, and the acceleration.

Figure 17 shows a scatter diagram of trajectories in the trajectory feature space of the axis of primary components one to three. Each point denotes one trajectory in the video sequences for training. Almost all of the normal trajectories gathered at the original point in the feature space and specific trajectories tended to be plotted far from the original point.

In addition, trajectories of the same behaviors tended to be plotted near each other. So, we created classes called PersonRuns, PeopleMeet, and ObjectPut by calculating their average positions $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ in the feature space. Ellipses denote classes of specific behaviors, and their radii denote standard deviations. A large bias is seen at the trajectories of the specific behavior PersonRuns. This showed that the PersonRuns behavior could be reliably detected from this feature space. On the contrary, the ObjectPut class was positioned near the original point and its variance was large. This showed that it is difficult to detect the ObjectPut behavior from only features of a trajectory.

The system detects specific behaviors by calculating the Mahalanobis distance from the position of an input trajectory $\mathbf{x}$ to each behavior class [32]. The Mahalanobis distance $D_M(\mathbf{x})$ is calculated with the following:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu})}, \tag{7}$$

where $\boldsymbol{\mu}$ is an average position of a behavior class in the trajectory feature space. Σ is a covariance matrix of each class.

The system sets decision scores according to the closeness for each behavior class. If a decision score exceeds the particular threshold for the motion, the system recognizes that the specific behavior might occur in the ID that contains the trajectory. The threshold is experimentally decided for each camera during the learning process. However, the system does not identify the behavior immediately. It verifies whether the behavior truly occurred by a verification process that we describe in the next section.

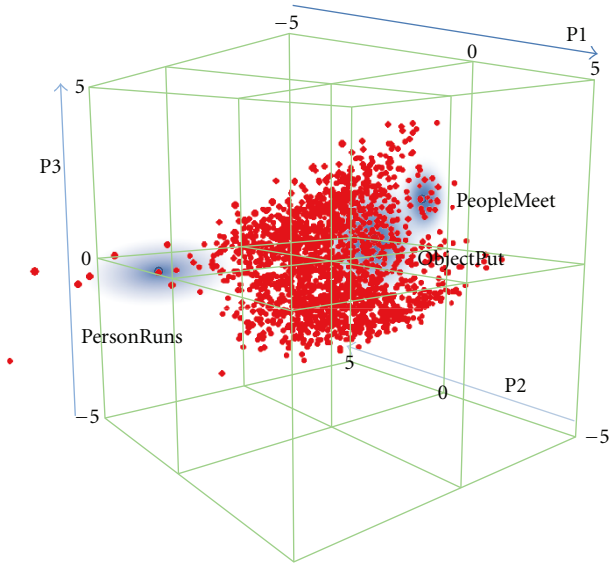

Figure 17: Scatter diagram of the trajectory feature space.

3.5. Human Behavior Verification

3.5.1. Verification of Fast Motion by Backward Tracking. People move in various ways in the surveillance video sequences. The movements can be roughly classified into three types: (1) fast motions that have large motion vectors such as PersonRuns, (2) big motions associated with traveling such as PeopleMeet and PeopleSplitUp, and (3) small motions not associated with traveling such as ObjectPut, TakePicture, and CellToEar. Our proposed system uses several methods to verify behavior depending on the motion type.

The system sometimes makes a mistracking, especially when the motion is fast, because of the large motion vectors. In addition, false detections for PersonRuns frequently occurred when the system misextracted a distant region because the motion vectors also tended to be large at greater distances. Thus, the system verifies the trajectory by searching backward after the PersonRuns behavior is identified in forward tracking. Tracking results between forward tracking and backward tracking are different in a crowded scene because the predicted positions of persons in each frame are different for forward and backward tracking, so the system tends to extract different regions.

Figure 18 shows a timeline of backward tracking. Once a PersonRuns event is detected in forward tracking (t_{f1}), the system starts backward tracking from the frame when the forward tracking finished (t_{f2}). The system tracks the person in the same way as in forward tracking. If the PersonRuns behavior is detected in backward tracking as well, the system identifies a PersonRuns event (t_{b1}). If the behavior could not be detected over the time when forward tracking was started (t_{f0}), it rejects the event by decreasing the decision score (t_{b2}). After that, the system restarts the processing from the time the tracking finished in forward tracking (t_{f2}).

Figure 19 shows sample images of the backward tracking. A PersonRuns event was detected by both forward and backward tracking. This technique reduces the number of false detections.

3.5.2. Verification of Big Motion by Following Trajectories. Continuing to follow a trajectory after a behavior is detected can be used to verify big motions, such as PeopleMeet and PeopleSplitUp. Our system does not stop tracking a person even after detecting a big motion with the human behavior recognition process. If a followed trajectory does not match one or more of the criteria decided experimentally in the training phase, our system rejects the occurrence of the behavior by decreasing the decision score. For example, if a trajectory with a travel distance greater than T_d was extracted just after detecting a PeopleMeet event as (8), our system rejects the behavior because most people do not walk far away immediately after meeting someone. T_d is the threshold decided in the training phase, (x_p, y_p) is the position of the detected PeopleMeet event, and (x_c, y_c) is the current position of the person in (8). This technique contributes to reducing false detections in the same way as backward tracking

$$\sqrt{\left(x_c - x_p\right)^2 + \left(y_c - y_p\right)^2} > T_d. \tag{8}$$

3.5.3. Verification of Small Motion by Optical Flows. It is difficult to detect small motions from only a trajectory, because a trajectory contains little information about small motions. Thus, it is hard to distinguish the trajectory from other normal trajectories in the trajectory feature space. Therefore, the system also uses optical flows [33] in the human region as a local feature for detecting small motions.

We tried to detect the ObjectPut behavior using optical flow. After a person stops moving (the motion can be detected from a trajectory), the system calculates optical flows within the detected human region. If the average of the vertical flows v_y was large and more downward than a threshold T_p that we set experimentally as (9), the system detected the ObjectPut behavior. The variable n in (9) denotes the number of optical flows in the human region. Figure 20 shows a sample of the optical flows in an ObjectPut scene. Though we can also apply this method to other small motions, particular criteria have to be set for each behavior

$$\frac{1}{n}\sum_{i=1}^{n} v_{y_i} < T_p. \tag{9}$$

4. Experiments

4.1. Comparison of Tracking Accuracy. We compared the tracking accuracy of our system with a baseline method that was created based on the KLT tracker. 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 and the method can be used with the OpenCV video library [34]. The KLT tracker detects motion vectors around moving objects; motion vectors that have the same length and direction tend to be detected around one person. Thus,

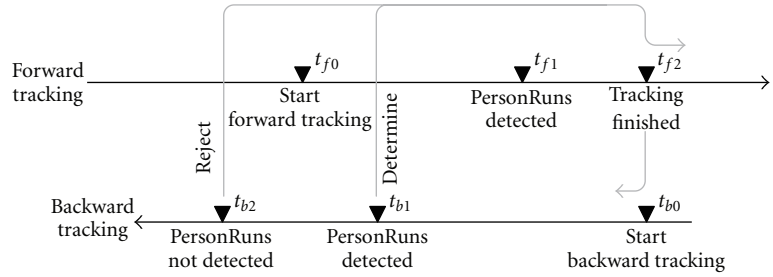

Figure 18: Timeline of backward tracking.

Figure 19: Sample backward tracking trajectory (right).

Figure 20: Optical flow in ObjectPut sequence.

Table 1: Success rate of two methods with different datasets.

	KTH	TRECVID
Baseline method	84.03%	33.9%
Proposed method	86.21%	47.62%

the baseline method uses motion vectors to cluster human regions.

We used two different datasets for the experiment; the KTH dataset and the TRECVID event detection dataset. The KTH motion sequences have been used in many motion recognition studies. The dataset contains sequences of a person walking in front of a smooth background, and only one person appears in each sequence. We used 100 sequences of walking people for the experiment. The TRECVID event detection dataset is a series of video sequences from five surveillance cameras set at different angles in an airport. A large number of people appear in each sequence and many occlusions occur. We used 14 sequences of the TRECVID dataset in which 120 people appeared.

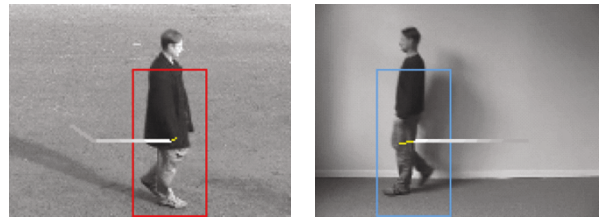

Figure 21: Tracking samples of baseline method using KTH dataset.

Table 1 lists the tracking accuracy results of the two methods using the two different datasets. The success rate S was calculated by

$$S = \frac{N_{\text{success}}}{N_{\text{detect}}}. \tag{10}$$

N_{success} is the number of trajectories that were successfully tracked, and N_{detect} is the total number of trajectories that the system detected. People in both datasets appeared for 3–5 seconds on average. Thus, we considered it a success when the system tracked a person correctly for more than 75 frames (3 seconds).

The baseline method was slightly worse than the proposed method for the KTH dataset. This is because the KTH dataset was not shot by completely fixed cameras, so motion vectors appeared even in the background as well as around people. However, both success rates of the two methods in the KTH dataset were quite high, indicating that both methods are accurate enough to track human regions in simple video sequences. Figure 21 shows samples of tracking results with the baseline method using the KTH dataset.

We can see the differences in the two methods in the results of the TRECVID dataset. Our method was 13.72% better than the baseline method. This indicates that our method can track human regions relatively robustly even in crowded scenes.

Both methods were sensitive to detect human regions so there were few false negative errors. Thus, the overwhelming majority of errors were false positive. To clarify the factors of the false positive error, we conducted another experiment using the TRECVID dataset. We defined two types of errors.

Table 2: Comparison of two methods.

	Error		Processing speed
	Mistracking	False alarm	
Baseline method	48.31%	17.80%	219 msec/frame
Proposed method	38.10%	14.29%	429 msec/frame

(1) *Mistracking*: tracking a human region for less than 75 frames (includes tracks jumping from one person to another).

(2) *False alarm*: tracking a noise region that does not contain a human for more than 75 frames.

Table 2 shows that mistracking occurred more frequently than false alarm. The baseline method tracks human regions based on motion vectors. It identifies objects based on the lengths and the directions of the motion vectors. Because many people walk in the same direction at approximately the same speed in a crowded scene, the tracking method tended to mistrack people because of the difficulty of distinguishing between adjacent or overlapping human regions. To overcome this, our method clusters human regions considering color histograms so that it can track human regions relatively robustly even in crowded scenes. In addition, our prediction process helped prevent a tracking region from shifting from one person to another. Consequently, our system had 10.21% fewer tracking errors than the baseline method.

The baseline method also had many false alarms. The method was very sensitive not only to human regions but also to other objects, such as shadows or baggage, because it does not distinguish people from other objects. On the other hand, our classifier can identify whether a region contains people or not because it was trained using supervised machine learning on examples of regions containing people (positive) and examples of regions not containing people (negative). This is one reason that our error rate of false alarm was 3.51% better than the baseline method.

At the same time, the processing time of our method could be a disadvantage. Table 2 shows that the proposed method takes about twice as long as the baseline method. The data is calculated by averaging the processing time for about 2 hours in the TRECVID dataset. Although the proposed method is slower due to the complexity of the processing, the performance improvement outweighs the disadvantage.

In addition, the baseline method is easier to apply to a new video sequence than our method because it does not need to learn human regions. The proposed method needs much prior knowledge such as background information and an average velocity map. However, most of the data is calculated automatically in the training phase, so the proposed system does not require much preparation.

4.2. Effectiveness of Verification Process. Our system has fewer false detections of specific behaviors than other systems because the verification process reduces this type of error. To confirm this fact, we conducted an experiment. Table 3 lists the number of false detections with and without the verification process for three behaviors. The false detections were extracted from about 16 hours of video sequences in the TRECVID dataset. The table also lists the reduction rate with the verification process.

Table 3: Occurrence of false detections.

	The number of false detections without verification process	The number of false detections with verification process	Reduction rate (%)
PersonRuns	1572	440	72.01
PeopleMeet	2198	1124	48.86
ObjectPut	37275	754	97.98

The verification process for the PersonRuns behavior reduced false detections to less than one third, from 1572 to 440. This indicates that backward tracking avoided two of three false detections. The process was effective for the PeopleMeet behavior as well. The false detections were reduced from 2198 to 1124.

These results show the effectiveness of the verification process. Even though the process restricts our system to not being able to be applied in real time, the accuracy rather than the processing speed should be emphasized.

Though false detections were reduced to only about 2% of the first decision for the ObjectPut behavior, the detecting accuracy of the behavior was low. We should also search for effective features for identifying small motions.

4.3. Recognition Accuracy of Specific Behaviors. We evaluated the recognition accuracy of our system for specific human behaviors in the TRECVID 2009 surveillance event detection task. We trained our detecting, tracking, and recognition algorithm using 100 hours of a development dataset. NIST evaluated our algorithm from submission data that was made using 44 hours of evaluation dataset.

We tried to detect three specific behaviors: PersonRuns, PeopleMeet, and ObjectPut using fast motion, big motion, and small motion. Table 4 shows the results. Detection cost rate (DCR) was used to evaluate the system performance. It was calculated using missed detection probability P_{Miss} and false alarm rate R_{FA} as shown in (11). The $\text{Cost}_{\text{Miss}}$, the Cost_{FA}, and the R_{Target} are constants with values set by NIST

$$\begin{aligned} \text{DCR}(\theta) &= P_{\text{Miss}}(\theta) + \beta * R_{\text{FA}}(\theta), \\ P_{\text{Miss}}(\theta) &= \frac{N_{\text{Miss}}(\theta)}{N_{\text{Targ}}}, \qquad R_{\text{FA}}(\theta) = \frac{N_{\text{FA}}(\theta)}{T_{\text{Source}}}, \\ \beta &= \frac{\text{Cost}_{\text{FA}}}{\text{Cost}_{\text{Miss}} * R_{\text{Target}}}, \\ \text{Cost}_{\text{Miss}} &= 10, \qquad \text{Cost}_{\text{FA}} = 1, \qquad R_{\text{Target}} = 20, \end{aligned} \tag{11}$$

where $N_{\text{Miss}}(\theta)$ is the number of missed detections at decision score θ, N_{Targ} is the number of event observations, $N_{\text{FA}}(\theta)$ is the number of false alarms at decision score θ, and T_{Source} is the total duration of the video segments in hours.

The results indicate that many false detections and missed detections occurred. However, our system performed

Table 4: Results in TRECVID 2009 event detection task.

Event	Reference	Our system	Correct	FalseAlarm	Miss	R_{FA}	P_{Miss}	DCR
PersonRuns	107	354	15	339	92	22.234	0.860	0.971
PeopleMeet	449	960	55	905	394	59.355	0.877	1.174
ObjectPut	621	488	19	469	602	30.760	0.969	1.123

Table 5: Comparison with other systems.

	Average of all DCR	Standard deviation of all DCR	Our DCR	Our rank
PersonRuns	2.651	2.700	0.971	1/13
PeopleMeet	2.349	1.705	1.174	4/8
ObjectPut	1.186	0.292	1.123	6/8

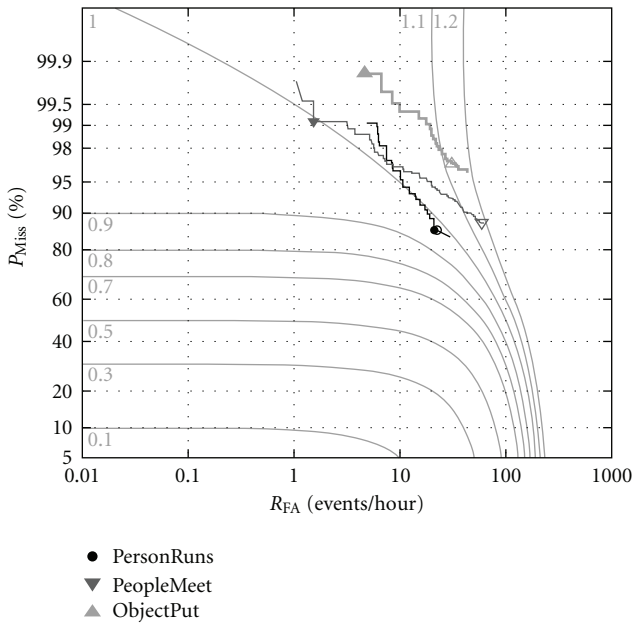

Figure 22: Results of our system in DET curve.

better than other systems developed by participants for the TRECVID 2009 surveillance event detection task.

Figure 22 shows the detection error tradeoff (DET) curve for each behavior. The DET curve plots a series of event-averaged missed detection probabilities and false alarm rates that are a function of a detection threshold, θ. This θ is applied to the system's detection scores, meaning the system observations with scores. The θ is declared to be the set of detected observations. If the P_{Miss} and the R_{FA} are low, the line is drawn around lower left in the figure. Filled points on the lines show the DCR.

Table 5 compares our results to systems developed by other participants. Our system was first place out of thirteen systems for the PersonRuns behavior. Our DCR (0.971) was much better than the average DCR of all systems (2.651). This shows that our system is reliable and effective.

An important technique for detecting the PersonRuns behavior is robust tracking of fast moving objects. The Kalman filter based prediction process accomplished this. The process set the search area for a tracked person depending on the speed of the person and the tracking situation so that our system could track running people robustly even if an extraction failure occurred in a few frames. The recognition process also contributed to good results. We could confirm a large bias on the axis of the first principal component in the trajectory feature space. The first principal component weighed heavily on the features of the average velocity and the travel distance. These features could be sufficiently extracted from a trajectory. Thus, our system was able to detect the PersonRuns behavior. In addition to the above two processes, our verification process was also effective in helping reduce false detections.

Our system could not detect the PersonRuns behavior when a person ran temporally (for less than one second) because the averaged velocity tended to be low in that case. It was particularly difficult to detect temporal dashing by children. However, the time length of trajectories can be configured freely so that we can set an appropriate length depending on the target person.

For the PeopleMeet behavior, our result was fourth place out of eight systems. The average DCR of all systems was 2.349. Our DCR was 1.174, so our system detected the PeopleMeet behavior relatively accurately compared to the other systems.

The places where one person can meet another are limited, because people do not stop where traffic density is high. Before meeting someone, a person tends to slow down his walking speed before stopping. A slight bias in the third principal component was confirmed in the trajectory feature space. The third principal component weighed heavily on the features of position and acceleration. So, we could divide the trajectory of a suspected PeopleMeet behavior from people who are moving without stopping. However, the distance from the center of major normal trajectories to this behavior class was relatively small in the trajectory feature space, causing many false detections. Even though the verification process reduced the number, the recognition accuracy was worse than for the PersonRuns behavior.

For the ObjectPut behavior, our result was sixth place out of eight systems. Our DCR was worse than the average DCR of all the systems. Our system is not reliable for detecting the ObjectPut behavior because that behavior is small compared to the other two behaviors, and it was difficult to recognize it from only a trajectory. In addition, there are many potential actions that fall under the category of ObjectPut. The behavior includes putting heavy baggage on the floor, picking up money at a register, and leaning on

a luggage cart. Though the system accounts for optical flows in the human region, it considered only downward motion. There is still room for improvement in the detection of small motions. In future work, we should set assumptions more strictly or extract more effective features for detecting small motions.

5. Conclusions

We proposed a method that can detect specific human behaviors even in crowded surveillance video sequences, and we developed a system that detects the NIST-defined behaviors PersonRuns, PeopleMeet, and ObjectPut. The system recognizes these behaviors by identifying a human region trajectory, which is created by detecting and tracking areas that contain people, which we call human regions, in video sequences. The system detects these human regions using HOG descriptors and an SVM classifier and uses a Kalman filter to track them robustly. The similarity of two human regions is evaluated by the simple Euclidean distance between them and the Bhattacharyya distance of 2D color histograms.

The system determines an occurrence of a specific behavior based on the distance from a trajectory to each class of specific behaviors in the trajectory feature space created using PCA. The system also has functions to verify detected behaviors. For example, it uses backward tracking to verify fast motions, and it calculates optical flows to verify small motions. These functions contribute to robust recognition of specific types of behavior.

We evaluated the tracking accuracy of our system by comparing it with a baseline tracking method. Our system was able to track human regions more robustly than the baseline method even for crowded scenes. We also showed that the verification process could reduce false detections effectively. In addition, the results of the TRECVID 2009 surveillance event detection task showed that our system could recognize human behaviors robustly; our recognition capability won first place for detecting the PersonRuns behavior.

The system is suitable for detecting fast motion and big motions because it recognizes behaviors based on motion trajectories, which contain rich information about these motions. We plan to analyze local features, such as optical flows, in detail to expand the range of human behaviors that can be recognized.

References

[1] G. Bradski and J. Davis, "Modeling people: vision-based understanding of a person's shape, appearance, movement, and behavior," *Computer Vision and Image Understanding*, vol. 104, pp. 87–89, 2006.

[2] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS, pp. 91–132, Carnegie Mellon University, 1991.

[3] D. Moore, *A real-world system for human motion detection and tracking*, Undergraduate Thesis, California Institute of Technology, May 2003.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pp. 886–893, San Diego, Calif, USA, June 2005.

[5] F. Han, Y. Shan, and R. Cekander, "A two-stage approach to people and vehicle detection with HOG-based SVM," *PerMIS*, pp. 133–140, 2006.

[6] T. Shiraki, H. Saito, Y. Kamoshida, et al., "Real-time motion recognition using CHLAC features and cluster," in *Proceedings of IFIP International Conference on Network and Parallel Computing (NPC '06)*, pp. 50–56, 2006.

[7] C. Schüldt, I. Laptev, and B. Caputo, "Recognizing human actions: a local SVM approach," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR '04)*, vol. 3, pp. 32–36, Cambridge, UK, August 2004.

[8] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, pp. 1395–1402, October 2005.

[9] A. Fathi and G. Mori, "Action recognition by learning mid-level motion features," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, Anchorage, Alaska, USA, June 2008.

[10] X. Sun, M. Chen, and A. Hauptmann, "Action recognition via local descriptors and holistic features," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pp. 58–65, Miami, Fla, USA, June 2009.

[11] Z. Li, Y. Fu, T. S. Huang, and S. Yan, "Real-time human motion recognition by luminance field trajectory analysis," in *Proceedings of ACM International Conference on Multimedia*, pp. 671–676, Vancouver, Canada, October 2008.

[12] K. Mikolajczyk and H. Uemura, "Action recognition with motion-appearance vocabulary forest," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, Anchorage, Alaska, USA, June 2008.

[13] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Proceedings of the 9th IEEE International Conference on Computer Vision*, vol. 2, pp. 726–733, Nice, France, October 2003.

[14] J. C. Niebles and L. Fei-Fei, "A hierarchical model of shape and appearance for human motion classification," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pp. 1–8, June 2007.

[15] K.-C. Lien and C.-L. Huang, "Multiview-based cooperative tracking of multiple human objects," *Eurasip Journal on Image and Video Processing*, vol. 2008, Article ID 253039, 2008.

[16] Y.-T. Tsai, H.-C. Shih, and C.-L. Huang, "Multiple human objects tracking in crowded scenes," in *Proceedings of the 18th International Conference on Pattern Recognition (ICPR '06)*, vol. 3, pp. 51–54, Hong Kong, August 2006.

[17] M. Hu, S. Ali, and M. Shah, "Learning motion patterns in crowded scenes using motion flow field," in *Proceedings of the 19th International Conference on Pattern Recognition (ICPR '08)*, pp. 1–5, Tampa, Fla, USA, December 2008.

[18] B. Schölkopf, 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, no. 7, pp. 1443–1471, 2001.

[19] P.-H. Chen, C.-J. Lin, and B. Schölkopf, "A tutorial on v-support vector machines," *Applied Stochastic Models in Business and Industry*, vol. 21, no. 2, pp. 111–136, 2005.

[20] P. Besse and J. O. Ramsay, "Principal components analysis of sampled functions," *Psychometrika*, vol. 51, no. 2, pp. 285–311, 1986.

[21] "TREC Video Retrieval Evaluation," http://www-nlpir.nist.gov/projects/trecvid/.

[22] G. R. Bradski and J. W. Davis, "Motion segmentation and pose recognition with motion history gradients," *Machine Vision and Applications*, vol. 13, no. 3, pp. 174–184, 2002.

[23] Y. Dedeoğlu, B. Uğur Töreyin, U. Güdükbay, and A. Enis Çetin, "Silhouette-based method for object classification and human motion recognition in video," in *Proceedings of the Computer Vision in Human-Computer Interaction*, vol. 3979, pp. 64–77, Graz, Austria, May 2006.

[24] Y. Ke, R. Sukthankar, and M. Hebert, "Event detection in crowded videos," in *Proceedings of the 11th IEEE International Conference on Computer Vision (ICCV '07)*, Rio de Janeiro, Brazil, October 2007.

[25] V. Pavlovic, J. M. Rehg, T.-J. Cham, and K. P. Murphy, "A dynamic Bayesian network approach to figure tracking using learned dynamic models," in *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV '99)*, vol. 1, pp. 94–101, Kerkyra, Greece, September 1999.

[26] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '99)*, vol. 2, pp. 246–252, Collins, Colo, USA, June 1999.

[27] Z. Garrett and H. Saito, "Live video object tracking and segmentation using graph cuts," in *Proceedings of International Conference on Image Processing (ICIP '08)*, pp. 1576–1579, San Diego, Calif, USA, October 2008.

[28] "National Institute of Standards and Technology," http://www.nist.gov/index.html.

[29] G. Xuan, P. Chai, and M. Wu, "Bhattacharyya distance feature selection," in *Proceedings of the International Conference on Pattern Recognition (ICPR '96)*, vol. 2, pp. 195–199, 1996.

[30] M. J. Grimble, *Robust Industrial Control: Optimal Design Approach for Polynomial Systems*, Prentice-Hall, Upper Saddle River, NJ, USA, 1994.

[31] X. Yu, C. Xu, Q. Tian, and H. W. Leong, "A ball tracking framework for broadcast soccer video," in *Proceedings of IEEE International Conference on Multimedia & Expo*, vol. 2, pp. 273–276, 2003.

[32] P. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 2, pp. 49–55, 1936.

[33] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185–203, 1981.

[34] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94)*, pp. 593–600, Seattle, Wash, USA, June 1994.