

Detecting and tracking of multiple pedestrians using motion, color information and the AdaBoost algorithm

JongSeok Lim · WookHyun Kim

Published online: 16 June 2012
© Springer Science+Business Media, LLC 2012

Abstract Robust detection and tracking of pedestrians in image sequences are essential for many vision applications. In this paper, we propose a method to detect and track multiple pedestrians using motion, color information and the AdaBoost algorithm. Our approach detects pedestrians in a walking pose from a single camera on a mobile or stationary system. In the case of mobile systems, ego-motion of the camera is compensated for by corresponding feature sets. The region of interest is calculated by the difference image between two consecutive images using the compensated image. Pedestrian detector is learned by boosting a number of weak classifiers which are based on Histogram of Oriented Gradient (HOG) features. Pedestrians are tracked by block matching method using color information. Our tracking system can track pedestrians with possibly partial occlusions and without misses using information stored in advance even after occlusion is ended. The proposed approach has been tested on a number of image sequences, and was shown to detect and track multiple pedestrians very well.

Keywords Pedestrian detection · Pedestrian tracking · Histogram of oriented gradient · AdaBoost

1 Introduction

Detection and tracking of pedestrians in image sequences has attracted much attention in computer vision research over the past couple of years [16, 29, 30, 32]. This has many practical applications such as video surveillance systems, and mobile robot systems. To perform this task, we need to detect all pedestrians in the image first, and then track them across other frames, while maintaining the correct identities. In particular, in the case of a stationary system, detection of moving objects is very simple, since the camera does not move. But, in the case of a mobile system, it is difficult to detect moving objects, since there

J. Lim · W. Kim (✉)
Department of Computer Engineering, Yeungnam University, 280 Daehak-Ro, Gyeongsan, Gyeongbuk,
South Korea
e-mail: whkim@yumail.ac.kr

J. Lim
e-mail: robertlim@yumail.ac.kr

is motion by both the camera itself, and moving objects in the environment. Those two motions are mixed together in the image sequences. Therefore, in order to robustly detect pedestrians, it should be able to analyze these two motions.

A number of approaches to stabilize camera motions have been proposed by vision researchers [3, 9, 26]. This paper estimates global motion using estimation of the transformation between two image coordinate systems. Once motion has been identified, pedestrian detection is not difficult if pedestrians are well apart from each other. However, if pedestrians are adjacent to each other, it is very difficult to segment them accurately. Many recent papers have used machine-learning methods to overcome this problem [4, 19, 27, 28]. But these methods have a relatively lower computational time than our method, which searches for a specific area of an image.

There are additional difficulties in tracking pedestrians after initial detection. The appearances of pedestrians change from time to time while walking, and also occlusion is caused by intersecting pedestrians. Thus it is hard to maintain the identities of objects during tracking in this case. A number of pedestrian tracking methods have been developed in recent years [1, 7, 11–13, 31].

We describe a method to track multiple pedestrians. Our tracking method is based on a block matching algorithm, using color information. Pedestrians are tracked based on complete detection where possible. To track possible partially occluded pedestrians, we first store the information of objects. In the presence of occlusion, only some parts of an object can be seen; in such cases, our system tracks the visible parts only. After occlusion, we can keep track of them using the information stored in advance.

Our approach has been tested on a number of image sequences with complicated backgrounds. We present quantitative evaluation results on both standard data sets, and data sets we have collected from the Internet. The results show that our approach is superior to other methods, for both detection and tracking.

The remainder of this paper is organized as follows: Section 2 introduces some related work; Section 3 gives an outline of our approach; Section 4 describes the proposed detection system; Section 5 presents the proposed pedestrian tracking method; Section 6 provides the experimental results; and the final section contains the conclusion and ideas for future work.

2 Related work

Many approaches to detect multiple pedestrians have been proposed. Most work has utilized a learning-based approach using a component or shape of a person, or motion and appearance information.

Mohan et al. [18] proposed an example-based object detection system in static images by components. This system is structured with four distinct example-based detectors trained to separately find four components of the human body: the head, legs, left arm, and right arm. After ensuring that these components are present in the proper geometric configuration, a second example-based classifier combines the results of the component detectors, to classify a pattern as either person or nonperson. Although this algorithm is more robust than the full-body person detection method and capable of locating partially occluded views of people, it is computationally too intensive for real-time application. Viola and Jones [28] described an efficient variant of the sliding window technique, which involves a detector trained (using AdaBoost) to detect a walking person, by taking advantage of both motion and appearance information. The appearance information was extracted from the first input images using rectangle filters, and the motion information was extracted from sequences of images using

rectangle filters. This method detects only pedestrians at very small scales. Zhao and Nevatia [30] presented the tracking of multiple humans in crowded environments. Humans are modeled with shape parameters represented by a number of ellipsoids, appearance parameters are represented by color histograms, and a modified Gaussian distribution is used to model the background. The optimization problem is calculated by a Markov Chain Monte Carlo (MCMC).

Another approach for detecting pedestrians involves using an infrared image, and Distance Transform or wavelet transform. Gavrilu [6] proposed a shape exemplar-based pedestrian detection system. This system detects candidate solutions using a shape matching approach based on the Distance Transform (DT), and verifies them using a pattern classification approach based on Radial Basis Function (RBF). Broggi et al. [2] employed vertical symmetry features for pedestrian detection in infrared images. This system is based on a multi-resolution localization of warm symmetrical objects with specific size and aspect ratio to detect both close and far pedestrians. Papageorgiou and Poggio [19] proposed a more robust person detection system based on Haar wavelet transform in combination with a SVM. This approach is invariant to changes in color and texture, and is used to robustly define a rich and complex class of objects such as persons. However, it is not able to detect a partially occluded or adjacent person.

On the other hand, the method of detecting a person using a moving camera has been performed by Jung and Sukhatme [14]. They have utilized camera ego-motion compensation to estimate camera motion, and an adaptive particle filter and Expectation-Maximization (EM) algorithm to estimate the positions of moving objects. This method detects only one person, not multiple persons.

Part-based model methods have been utilized. Shashua et al. [22] divide the human body into nine regions, for each of which a classifier is learned based on features of orientation histograms. Mikolajczyk et al. [17] divide the human body into seven parts, face/head for frontal view, face/head for profile view, head-shoulder for frontal and rear view, head-shoulder for profile view, and legs. For each part, a detector is learned by following the Viola-Jones approach applied to orientation features. These methods do not use part-based tracking nor consider occlusions.

A variety of methods to track multiple pedestrians have been used. Isard and MacCormick [10], Smith et al. [25], and Peter et al. [20] use multiple object models to explain foreground or motion objects. These methods handle occlusions by computing joint image likelihood of multiple objects. To perform optimization for joint hypotheses spaces of high dimension, a particle filter [10], Markov Chain Monte Carlo [25] or Expectation Maximization algorithm [20] is used. All of these methods have shown experiments with a fixed camera only, where background subtraction provides relatively robust object motion models. They assume all moving objects are from pedestrians. Although this is true in some environments, it is not suitable in general situations.

A pose tracking method has been used to track humans by Lee and Nevatia [15], Ramanan et al. [21], and Sigal et al. [24]. However the objectives and methodologies of pose tracking methods and multiple human tracking methods are distinctly different. These pose tracking methods consider only one human. Although they can work with temporary or slight partial occlusions, because of the use of part representation and temporal consistency, they do not work well with persistent and significant occlusions, as they do not model occlusions explicitly, and the part models used are not very discriminative. The automatic initialization and termination algorithms in the existing pose tracking methods are not general. In [21], a human track is started only when a side view walking pose human is detected, and no termination algorithm is mentioned.

3 Outline of our approach

A schematic diagram of our system is shown in Fig. 1.

Pedestrian detection is done frame by frame. We use motion information to find the region of interest where pedestrians are likely to exist. In the case of a mobile system, we use frame difference by ego-motion compensation from two consecutive image frames to find moving objects. In the case of a stationary system, we search the region of interest using frame difference only. Then we detect pedestrians using a detector learned from Histogram of Oriented Gradient (HOG) features [5]. These features are suitable for pedestrian detection as they are relatively invariant to illumination differences. We have the detector learn by a boosting approach proposed by Viola and Jones [27].

Our tracking method is based on a block matching algorithm using color information. Detection responses from the pedestrian detector are taken as inputs for the tracker. We track pedestrians by matching the object hypotheses with the detection responses, whenever corresponding detection responses can be found. We search for the best matching area by comparing the detection response window of the current frame with the corresponding one of the previous frame. If an occlusion occurs while tracking, a pedestrian who had been obscured cannot be tracked because of loss of information. In order to keep track of pedestrians after occlusion, we store the information of detected pedestrians. Thus our tracking system can robustly track pedestrians without misses, even if occlusions occur.

4 Detection of multiple pedestrians

We detect pedestrians by applying the region of interest to the pedestrian detector that is learned from HOG features.

4.1 Moving objects detection

Typically, the straightforward and fast method to detect moving objects is to use the frame difference between two consecutive image frames when the input camera is static. However, if the camera moves as when it is mounted on a mobile system, this method is not applicable because objects in the background region are also extracted by the moving camera. In general, there are two independent motions involved in the moving camera environment: motions of moving objects and the camera ego-motion. These two motions are blended into a single image. Therefore, the ego-motion of the camera should be eliminated so that the motion of moving objects can be effectively detected. The detection of moving objects is performed according to frame difference, but the ego-motion of the camera in the previous image is compensated for, before comparing it with the current image.

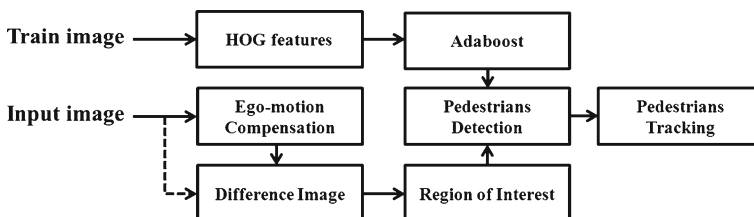


Fig. 1 A schematic diagram of our system

In the real world environment, perfect ego-motion compensation is rarely achievable because of various noise sources. Even assuming that the ego-motion compensation is perfect, the difference image would still contain structured noise on the boundaries of objects because of the characteristic of monocular images. We remove these noise terms using the area elimination method.

4.1.1 Ego-motion compensation

The ego-motion of the camera can be estimated by feature tracking between images [3]. When the camera moves, two consecutive images, I^t (the current image) and I^{t-1} (the previous image) cannot be directly compared since these coordinate systems are different. Thus, it is necessary to compensation for ego-motion, which is a transformation from the image coordinate of I^{t-1} to that of I^t . Transformation can be estimated using both a set of features in I^t , and a set of corresponding features in I^{t-1} . We adopt the Harris corner detector [8] for feature set selection. The feature selection algorithm generates features (f^{t-1}) by running on image (I^{t-1}). The feature-tracking algorithm by Shi-Tomasi [23] is applied to find the corresponding set of features (f^t) in the subsequent image (I^t).

Once correspondence is generated, the ego-motion of the camera can be estimated using a transformation model. We used a bilinear model among the various transformation models because it is a nonlinear transformation model that can estimate most ego-motion of the camera regardless of the length of the interval between consecutive images.

A bilinear model used in our experiments is as follows:

$$\begin{bmatrix} f_x^t \\ f_y^t \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 & a_7 \end{bmatrix} \begin{bmatrix} f_x^{t-1} & f_y^{t-1} & 1 & f_x^{t-1} f_y^{t-1} \end{bmatrix}^T. \quad (1)$$

The model parameters for ego-motion compensation are estimated by minimizing cost. Given a transformation model, the cost function for least square optimization is defined as

$$J = \frac{1}{2} \sum_{i=1}^N (f_i^t - T_{t-1}^t(f_i^{t-1}))^2. \quad (2)$$

where N is the number of features. When transformation is computed, some of the features cause inaccurate transformation because they are associated with moving objects. These features should be eliminated from the feature set if they satisfy the following conditions:

$$|f_i^t - T_{0_{t-1}}^t(f_i^{t-1})| \geq \epsilon. \quad (3)$$

where ϵ is the predefined threshold and T_0 is the initial estimate using the full feature set. Then, the final estimate T is computed.

For ego-motion compensation, image I^{t-1} is converted using the transformation model before being compared to image I^t . For each pixel (x,y) , the difference image between two consecutive images is computed using the compensated image as follows:

$$I_{di}(x,y) = \left| \left(I_{t-1} \left(T_{t-1}^t \right)^{-1}(x,y) \right) - I_t(x,y) \right|. \quad (4)$$

where T_{t-1}^t is the bilinear model and $I_{di}(x,y)$ is the difference image.

Figure 2 shows examples of difference image examples. Figure 2(a) and (b) provide the image at time $t-1$ and time t respectively. Figure 2(c) gives the difference image without ego-motion compensation, and Fig. 2(d) presents the difference image with ego-motion compensation.

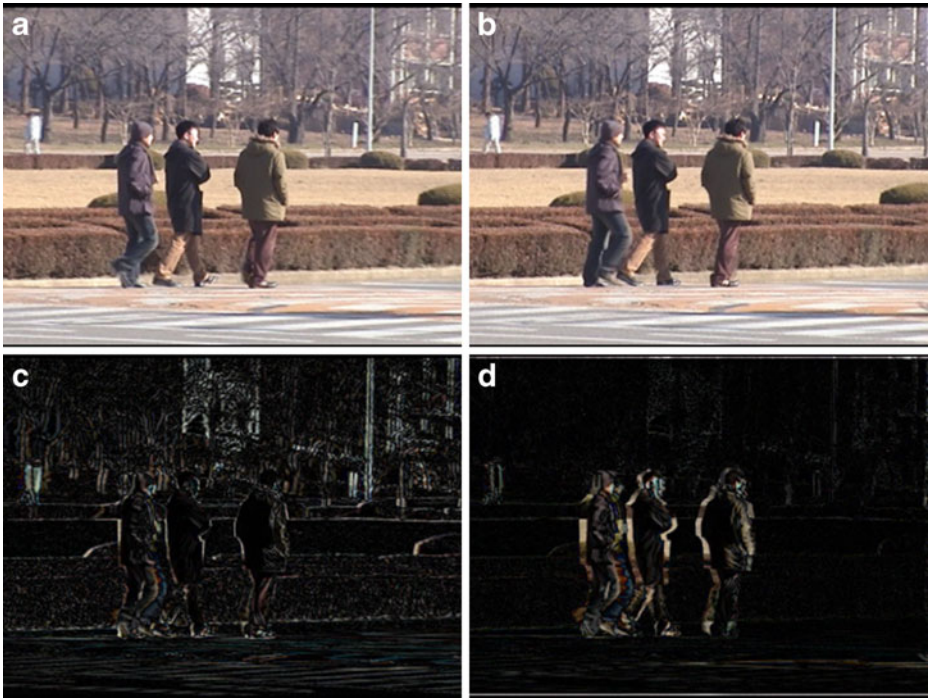


Fig. 2 Examples of difference image results

4.1.2 Preprocessing procedure

The difference image generated in the previous step is transformed into a binary image to find the region of interest efficiently. The binary image is greatly affected by the threshold. If a threshold is very small, then unnecessary noise sources are formed in the image space. These noise terms lead to serious trouble in the next step. On the other hand, if a threshold is very big, then nothing or only a small part of the objects remains in the image space. In this case, the region of interest becomes the entire image area. Thus, determining a suitable threshold is very important. We use a threshold (e.g. 40) obtained by many experiments.

The binary image has a lot of unnecessary noise that interferes with the detection of the region of interest. To remove noise terms, we use the area elimination method. This counts the number of pixels existing within the window area while moving the sliding window of $n \times n$ at first. Then, if the number is smaller than a threshold, all pixels in the window area are eliminated. In this paper, we use a sliding window of a fixed size (25×30), and a threshold of 72.

Figure 3 shows examples of preprocessing results. Figure 3(a) gives the binary image applied the threshold 40 to the difference image and Fig. 3(b) presents the image in which noise terms are eliminated by our approach.

4.1.3 Finding the region of interest

In general, the region of interest in which moving objects are likely to exist is searched by the projection approach using the image generated in the previous step. The projection

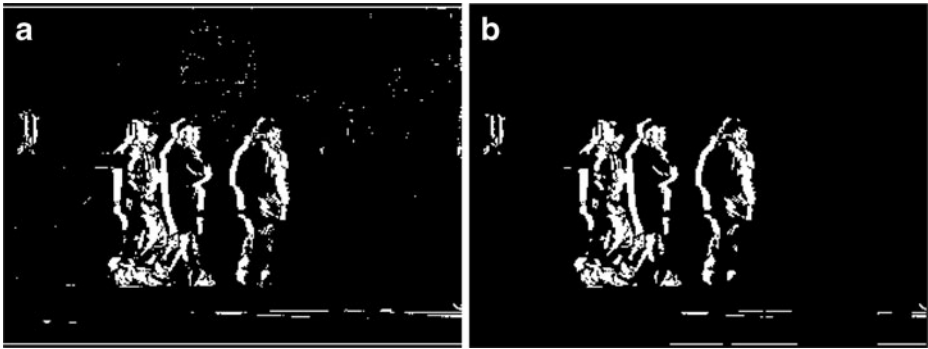


Fig. 3 Examples of preprocessing results

method generates the histogram by counting of the pixel values greater than 0 in the horizontal and vertical direction respectively. The formulation is as follows:

$$H_x = \sum_{y=0}^{n-1} I_{nd}(x, y). \quad (5)$$

$$V_y = \sum_{x=0}^{m-1} I_{nd}(x, y). \quad (6)$$

where H_x and V_y is the projection histogram for horizontal and vertical direction respectively. $I_{nd}(x, y)$ is the image in which noise terms are eliminated by our approach. In the above equation, x is from 0 to $m-1$, and y is from 0 to $n-1$: and m and n is the height and width of the image respectively.

This projection histogram is used to find the region of interest to detect pedestrians by the AdaBoost algorithm. We utilize the aspect ratio and shape information of a human body while finding the region of interest. The results are represented by a bounding box. The number of the bounding box varies depending on the type of moving objects. If pedestrians are adjacent to each other, the size of the bounding box is very large. Otherwise if they are far away from each other, the size is small, and the number is several.

Figure 4(a) shows the projection histogram created by our method, and Fig. 4(b) presents the region of interest searched by the projection approach.

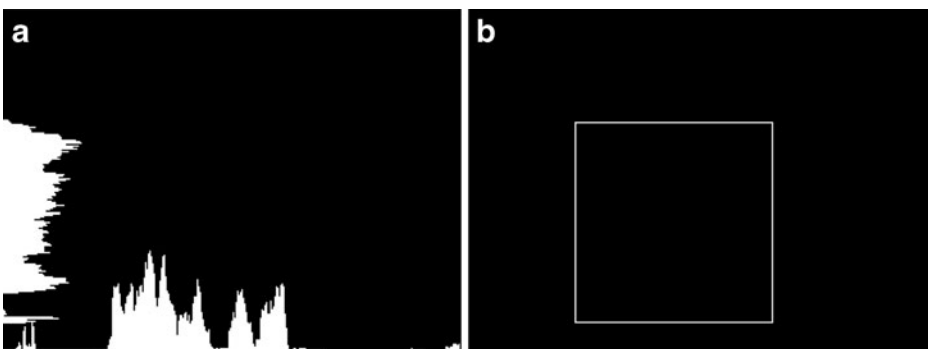


Fig. 4 The projection histogram and the region of interest

4.2 Pedestrian detection using the Adaboost algorithm

In the previous step, we found the region of interest. This is used to detect pedestrians using the AdaBoost algorithm. This algorithm generates a pedestrian detector that is learned from the Histogram of Oriented Gradient (HOG) features [5].

4.2.1 Histogram of oriented gradient features

Histogram of Oriented Gradient (HOG) descriptors provide an excellent performance for pedestrian detection relative to other existing feature sets. This is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. HOG feature extraction is based on magnitude and direction of gradient. Given the magnitude of a pixel $I(x,y)$, the magnitude $m(x,y)$ and the direction $\theta(x,y)$ of gradient are as follows:

$$m(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2}. \quad (7)$$

$$\theta(x,y) = \tan^{-1} \frac{G_y(x,y)}{G_x(x,y)}. \quad (8)$$

Where

$$G_x(x,y) = I(x+1,y) - I(x-1,y). \quad (9)$$

$$G_y(x,y) = I(x,y+1) - I(x,y-1). \quad (10)$$

Each pixel calculates a vote for an edge orientation histogram based on the orientation of the gradient element centered on it, and the votes are accumulated into orientation bins over 8×8 regions that it calls a cell. The orientation bins are spaced at intervals of 20° over 0° – 180° (“unsigned” gradient), and have 9 bins per cell.

Gradient intensities vary over a wide range owing to local variations in illumination and foreground-background contrast, so effective local contrast normalization is essential for good performance. The normalization scheme is based on grouping 3×3 cells into one block. The one block then has 81 bins. Typically the blocks are overlapped so that each scalar cell response contributes several components to the final descriptor vector, each normalized with respect to a different block. For a sample size of 64×128 , the overall number of possible blocks is 84 (6×14 for horizontal and vertical direction). The block normalization scheme uses $L2$ -norm. Let v be the unnormalized descriptor vector, $\|v\|_2$ be 2 -norm, and ε be a small constant ($\varepsilon=1$). The scheme is as follows:

$$v \rightarrow \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon^2}}. \quad (11)$$

We use 64×128 images to train a classifier, and then the number of HOG features is 6,804 (84 blocks \times 81 bins). These features are used to build a set of classifiers.

Table 1 The AdaBoost algorithm for classifier learning

1. Given example images $(x_1, y_1), \dots, (x_n, y_n)$ where $y_i=0,1$ for negative and positive examples respectively.
2. Initialize weights $w_{1,i} = \frac{1}{2^m}, \frac{1}{2^l}$ for $y_i=0,1$ respectively, where m and l are the number of negatives and positives respectively.
3. For $t=1, \dots, T$:
 - (a) Normalize the weights, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$ so that w_t is a probability distribution.
 - (b) For each feature, j , train a classifier h_j which is restricted to using a single feature.
The error is evaluated with respect to w_t , $\epsilon_j = \sum_i w_{t,i} |h_j(x_j) - y_i|$.
 - (c) Choose the classifier, h_t , with the lowest error ϵ_t .
 - (d) Update the weights: $w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$
where $e_i=0$ if example x_i is classified correctly, $e_i=1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
4. The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
 where $\alpha_t = \log \frac{1}{\beta_t}$

4.2.2 Classifier

To detect pedestrians, we use the AdaBoost algorithm proposed by Viola and Jones [27, 28]. By default, a strong classifier to build a reliable pedestrian detector can be formed by a linear combination of weak classifiers. Such classifiers are trained with a learning algorithm to give good classification results at a small computational cost. A weak learning algorithm is to select the single feature that best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples is misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j , a threshold θ_j , and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) > \theta_j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where x is a 64×128 learning image. Table 1 shows a summary of the boosting process.

Here we describe a method for constructing a cascade of classifiers that enhances the detection performance while drastically reducing computational time. Boosted classifiers can be constructed that reject many of the negative instances while detecting almost all

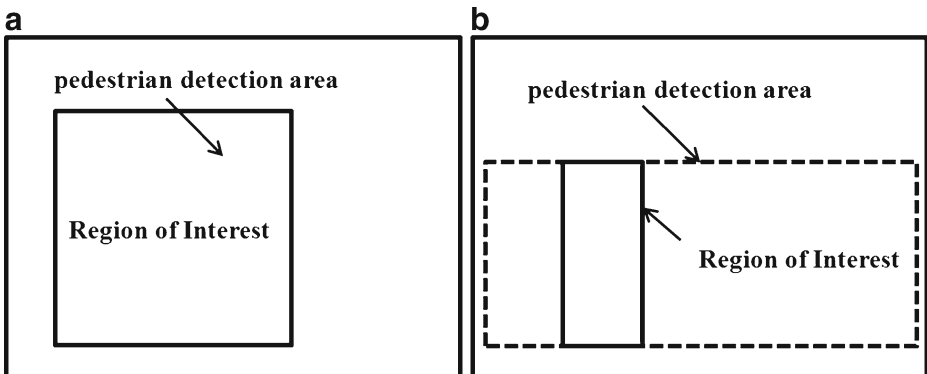


Fig. 5 Examples of the pedestrian detection area

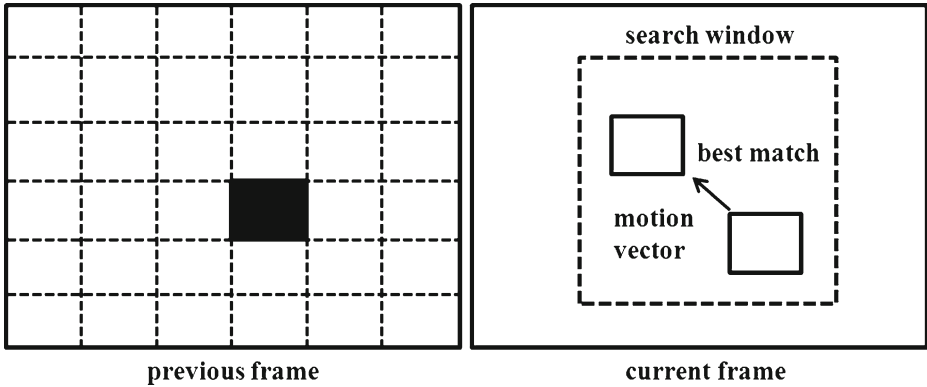


Fig. 6 Block matching motion estimation

positive ones. A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers a third classifier, and so on.

Stages in the cascade are constructed by training classifiers using AdaBoost and then adjusting the threshold to minimize false negatives. The default AdaBoost threshold is defined to yield a low error rate on the training data. In general a lower threshold yields higher detection rates and higher false positive rates.

4.2.3 Detection system

Detection proceeds in two stages: first, the region of interest is searched using motion information; second, pedestrians are detected by our classifier based on the sub window within the region of interest. Figure 5 shows examples of the pedestrian detection area. If the size of the region of interest is more than 2/3 of the input image, the detection conducts from the minimum window (24×48) to the maximum window (120×240) with a scale factor of 1.2 within the area. If the size of the region is less than 2/3 of the input image, the detection conducts in the horizontal direction that exists in the region of interest.

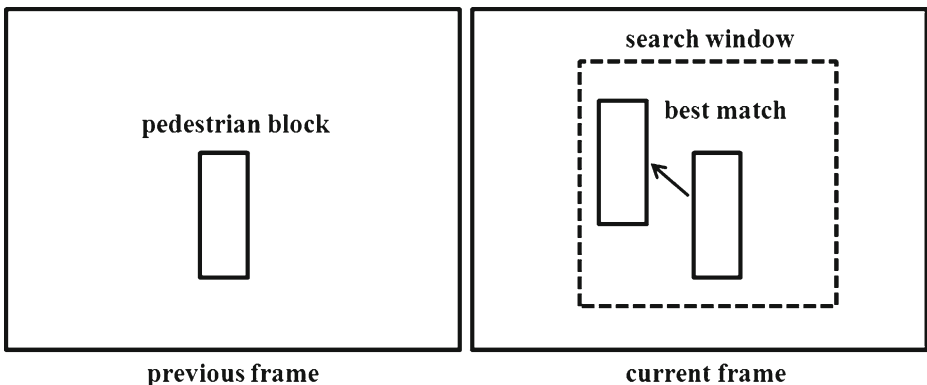


Fig. 7 The process of block matching



Fig. 8 Examples of positive and negative training samples

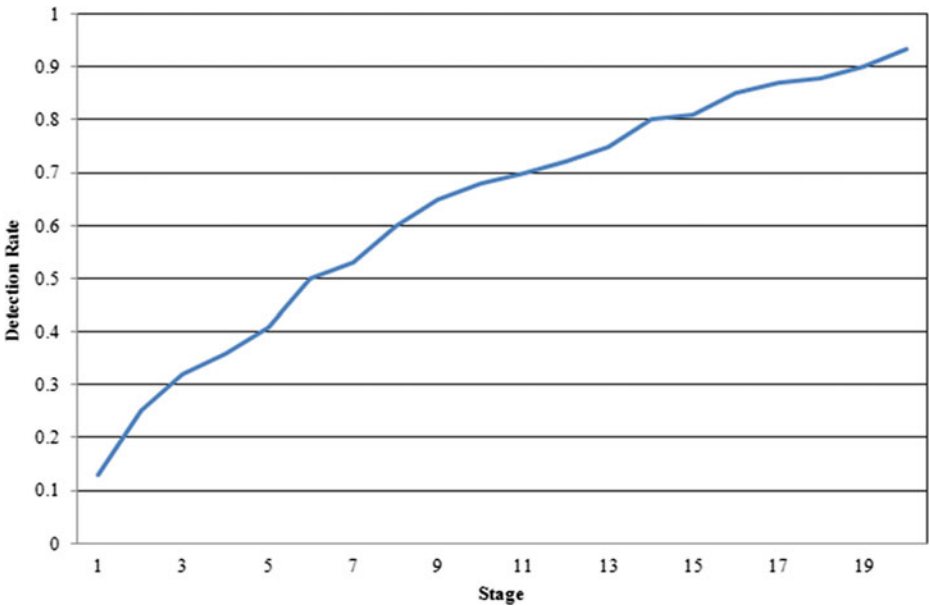


Fig. 9 Detection rate of each stage for our pedestrian detector on the test set



Fig. 10 Examples of detection results on images from our test set

Thus, our method has a relatively lower computational time than the previous method because the pedestrian detection area is the region of interest and smaller than the image. Meanwhile, the information of pedestrians detected is stored to use while tracking back after an occlusion has taken place.

5 Pedestrian tracking based on color information

The pedestrian tracking algorithm uses the block matching method based on color information. The process of the block matching algorithm is illustrated in Fig. 6. In a typical block matching algorithm, each frame is divided into blocks, each of which consists of luminance

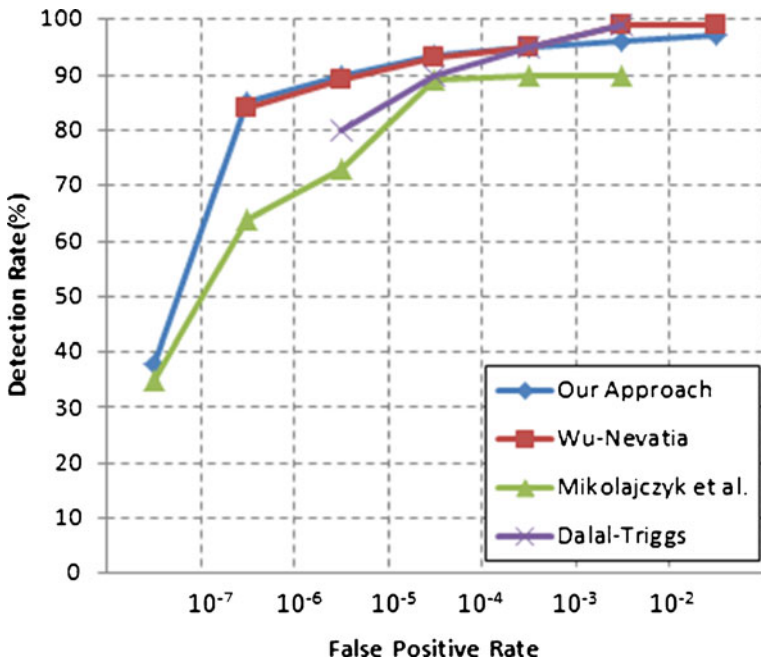


Fig. 11 ROC curves of evaluation for the detection rate and the false positive rate

and chrominance blocks. Generally, motion estimation is performed only on the luminance block. Each luminance block in the previous frame is matched against candidate blocks in a search area on the current frame. These candidate blocks are just displaced versions of the original block. The best candidate block is found and its displacement (motion vector) is recorded.

The process of our block matching algorithm using color information is illustrated in Fig. 7. In our block matching algorithm, the pedestrian block is the pedestrian area detected in the previous step. Like a typical block matching algorithm, the block in the previous frame is matched against candidate blocks in a search window on the current frame. The best candidate block is found and recorded.

In order to get the best match block in the current frame, it is necessary to compare the pedestrian block of the previous frame with all the candidate blocks of the current frame. The similarity of the block is estimated by the sum of absolute difference using RGB color. Assuming that the pedestrian block of $M \times N$ in the previous frame exists, the formula for the sum of absolute difference between that block and the block of $u \times v$ displacement in the current frame is as follows:

Table 2 Tracking performance comparison with the previous methods on the CAVIAR set

	Accur.	F. Neg.	F. Pos.	IDSw.
Zhao-Nevatia	64 %	14 %	22 %	20
Wu-Nevatia	74 %	2 %	24 %	19
Our approach	75 %	21.3 %	3.7 %	15

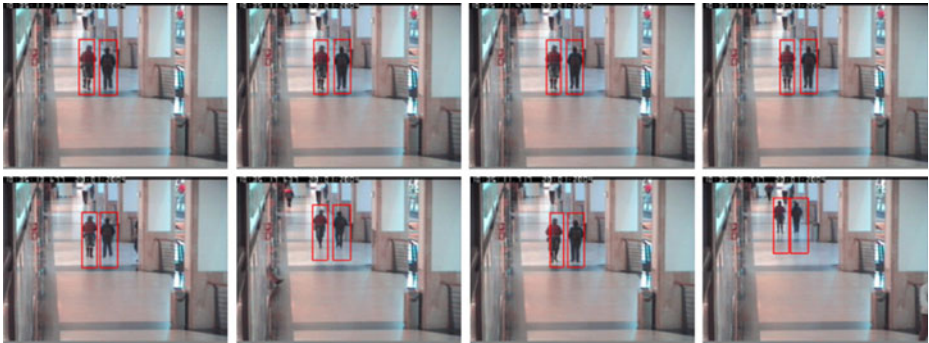


Fig. 12 Sample tracking results on the CAVIAR set

$$SAD_{(x,y)}(u, v) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{SAD_R + SAD_G + SAD_B}{3}. \tag{13}$$

$$SAD_R = |I_R^{t-1}(x + i, y + j) - I_R^t(x + u + i, y + v + j)|. \tag{14}$$

$$SAD_G = |I_G^{t-1}(x + i, y + j) - I_G^t(x + u + i, y + v + j)|. \tag{15}$$

$$SAD_B = |I_B^{t-1}(x + i, y + j) - I_B^t(x + u + i, y + v + j)|. \tag{16}$$

We have used the $u \times v$ displacement value to 16×16 in the experiment. The pedestrian tracking is to search the block that has the lowest SAD value.

Meanwhile, in order to check for occlusion, we investigate whether overlapping pedestrian blocks exist or not while tracking. If part of a pedestrian block overlaps another block even a little, it can be seen that an occlusion has occurred, or is likely to occur in the area. If more than 1/2 of a pedestrian is occluded by other pedestrians, it is difficult to track the pedestrian, and when tracking back after occlusion is ended, it is impossible to track the pedestrian due to loss of information. In such a case, our algorithm can keep track of the pedestrian using the information stored in advance.

6 Experimental results

The system described in the paper was tested by using image sequences with pedestrians of different size, pose, gait and clothing. We note that our focus is on the detection and tracking of pedestrians, where the camera may be stationary or moving. There are not many public data sets with moving camera characteristics. Thus, we captured our own data set from a

Table 3 Tracking performance on the passing pedestrians set

Accur.	F. Neg.	F. Pos.	IDS _w .
76 %	21 %	3.0 %	8

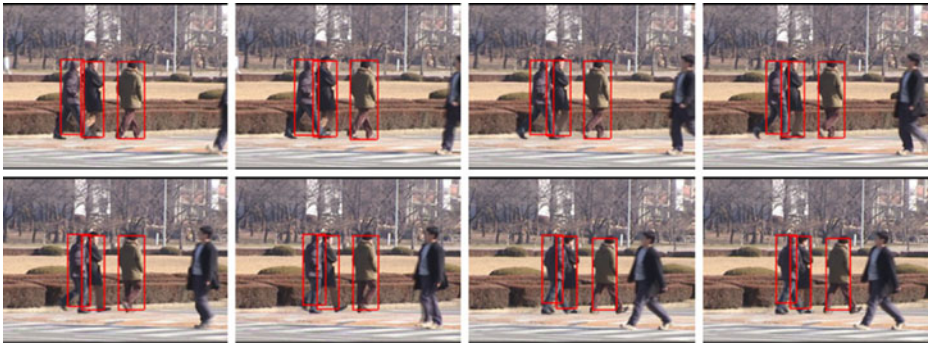


Fig. 13 Sample tracking results on the passing pedestrians set

camera held by a person walking on a road. We separate the performance evaluation of detection and tracking systems.

6.1 Detection evaluation

The experimental procedure consists of two stages: finding of region of interest and then applying of pedestrian detector. First of all, we train pedestrian detector using the AdaBoost algorithm. Then we find region of interest from two consecutive images and detect pedestrians by the pedestrian detector in the area.

To train and evaluate the pedestrian detector, we use a number of sample sets and test sets. Our training set consists of 3,547 pedestrian samples scaled and aligned to a base resolution of 64×128 pixels. The negative image set contains 7,100 negative images which were manually inspected and found to not contain any pedestrians. Figure 8 shows some examples from our training set. In the paper, some parameters to train define as follows: the correct detection rate and the false positive rate per each stage sets to 0.9 and 0.05 respectively. If the correct detection rate is less than 0.9 or the false positive rate is greater than 0.05, while training from each stage, it is transferred to the next stage. A 20 stage cascaded classifier was trained to detect pedestrians using the AdaBoost algorithm. Figure 9 shows the detection rate of each stage for our pedestrian detector on the test set. The detection rate in the first stage of the detector is 0.13 and the remaining stages have an increasingly larger detection rate.

To test the detection and false positive rate of our approach, we use a test set with 1,521 positive samples and 700 negative images. We performed our experiments on a 2.3GHZ Pentium IV CPU. Figure 10 shows some examples of successful detection results on images from our own test set and CAVIAR set.¹ The sizes of the pedestrians considered vary from 24×48 to 120×240 . The size of the images in the first four rows of Fig. 10 is 320×240 pixels on our own test set, and in the last two rows is 384×288 pixels on the CAVIAR set. It can be seen that false alarms occur where localized parts of the images look like pedestrians. On our test set, our detector has a detection rate of 93.5 % with a false positive rate of 10^{-4} .

We also compare our method with previous ones for the detection rate, on which many previous papers report quantitative results. A direct comparison of our method with previous ones is difficult due to variability in data set and dissimilarity in detection result. We show a comparison with other methods that report results on the test set. Figure 11 shows the ROC

¹ <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>

Table 4 Tracking performance on the oncoming pedestrians set

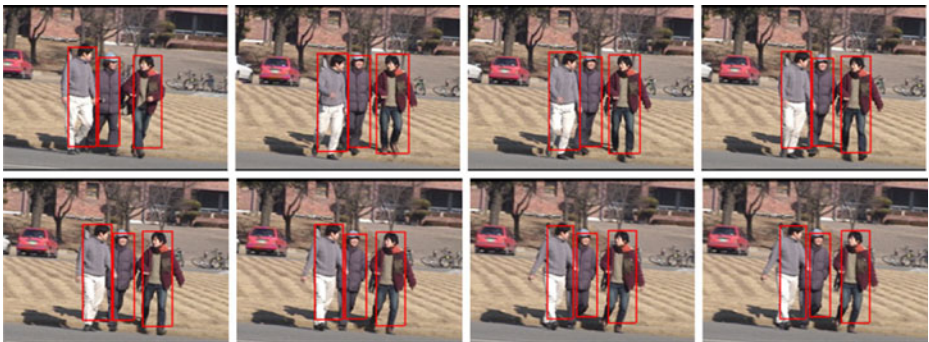
Accur.	F. Neg.	F. Pos.	IDS _w .
75.2 %	22.5 %	2.3 %	3

curves of evaluation for the detection rate and false positive rate. It can be seen that our approach is superior to previous ones for the detection rate.

6.2 Tracking evaluation

We also evaluated our tracking algorithm on three video sets. The first set and second set, called the “passing pedestrians set” and the “oncoming pedestrians set” respectively, are captured from a camera held by a person walking on a road. The frame size of these two sets is 320×240 and the sampling rate is 30 FPS. The third set is a selection from the CAVIAR video corpus [11], which is captured with a stationary camera, mounted a few meters above the ground and looking down towards a corridor. The frame size is 384×288 and the sampling rate is 25 FPS. We compare our results on the CAVIAR set with the previous methods. We are unable to compare with others as we are unaware of published, quantitative results for tracking on this set by a number of other researchers.

For comparison with the previous methods, our test set consists of the sequences for the “shopping center corridor view” of the CAVIAR set. The scene is relatively simple and the occlusion between objects tends to occur frequently. This makes it very difficult to track pedestrians. Our pedestrian detector requires the width of pedestrians to be larger than 24 pixels. In the CAVIAR set there are a number of pedestrians, which are smaller than 24 pixels most of the time, and many pedestrians, which are occluded by pillar or wall, the other objects. These pedestrians may not be detected by the pedestrian detector. Our method is possible to track only pedestrians detected by the one. Table 2 shows the tracking performance comparison with the previous methods on the CAVIAR set. It returns an accuracy score (composed of false negative rate, false positive rate, and number of ID switches). We cannot ascertain the exact value for the accuracy, false negative and false positive rate from the results of the previous methods. Thus we calculate as follows: an accuracy is derived from $MT(\text{number of mostly tracked trajectories})$ divided by $GT(\text{ground truth})$; a false negative rate is derived from $FAT(\text{number of false trajectories})$ divided by GT ; a false positive rate is derived from $100 - \text{Accuracy} - \text{False Negative rate}$.

**Fig. 14** Sample tracking results on the oncoming pedestrians set

As shown in Table 2, it can be seen that our method outperforms the previous methods. In the methods of the latter, the false negative is lower than the false positive. However, in the method of the former, the false negative is higher than the false positive. Thus it can be seen that our method is mostly possible to track the detected pedestrians. Figure 12 shows some sample tracking results on the CAVIAR set. As shown in Fig. 12, small pedestrians are not tracked because it has not been detected by the pedestrian detector.

Secondly, we evaluate results on passing pedestrians set. The main difficulties of this set are the emergence of a new pedestrian and frequency of occlusions. Table 3 gives the tracking performance of the system. It can be seen that our method works considerably well on this set. Figure 13 shows some sample tracking results on the passing pedestrians set.

Finally, we evaluate results on oncoming pedestrians set. The main difficulty of this set is due to some occlusions between objects. Table 4 gives the tracking performance of the system. It can be seen that our method works well on this set. Figure 14 shows some sample tracking results on the oncoming pedestrians set.

For tracking on three test sets, we have obtained average accuracy of 75.4 %. Even though the false negative is high, the false positive is low. Thus it can be seen that our tracking method is mostly possible to track pedestrians detected by the pedestrian detector.

7 Conclusion

In this paper we have presented an approach to detect and track multiple pedestrians using motion, color information and the AdaBoost algorithm. The key point of our detection approach is to find the region of interest to obtain a relatively low computational time. To do this, we detect moving objects using a frame difference image. In the case of a mobile system, the ego-motion of the camera is estimated using corresponding feature sets. However in the case of a stationary system, it is not necessary. To detect pedestrians within or surroundings the region of interest, we have used the AdaBoost classifier learned by the Histogram of Oriented Gradient (HOG) features. Pedestrians have been tracked by the block matching method using color information. Our tracking system can track pedestrians with possibly partially occlusions and without misses using information stored in advance, even after occlusion is ended.

This paper presented a set of various experiments with a complicated background, and showed excellent experiment results. Our pedestrian detector has obtained a detection rate of 93.5 %, and is compared to other systems to evaluate performance. We have not considered the interaction between detection and tracking. The proposed system works in a sequential way: the input of tracking takes the results of detection. Thus it cannot track pedestrians who have recently entered into image frame. We plan to study such problems in future work.

Acknowledgement This research was supported by the Yeungnam University research grants in 2010.

References

1. Arulampalam MS, Maskell S, Gordon N, Clapp T (2002) A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans Signal Process* 50(2):174–188
2. Broggi A, Fascioli A, Fedriga I, Tibaldi A, Rose MD (2003) Stereo-based preprocessing for human shape localization in unstructured environments, in: *Proc. of the IEEE Intell Vehicle Symp*, pp. 410–415

3. Censi A, Fusiello A, Roberto V (1999) Image stabilization by features tracking, in: Proc. of the Int Conf Image Anal Process, pp. 665–667
4. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods, Cambridge University Press
5. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection, in: Proc. of IEEE Conf Comput Vis Pattern Recogn, pp. 886–893
6. Gavrilu D (2000) Pedestrian detection from a moving vehicle, in: Proc. of the 6th Eur Conf Comput Vis, pp. 37–49
7. Gutman P, Velger M (1990) Tracking targets using adaptive Kalman filtering. EEE Trans Aero Electron Syst 26(5):691–699
8. Harris C, Stephens MJ (1998) A combined corner and edge detector, in: Proc. of the 4th Alvey Vis Conf, pp. 147–152
9. Irani M, Rousso B, Peleg S (1994) Recovery of ego-motion using image stabilization, in: Proc. of the IEEE Comput Vis Pattern Recogn, pp. 454–460
10. Isard M, MacCormick J (2001) BraMBLe: a Bayesian multiple-blob tracker, in: Int Conf Comput Vis, pp. 34–41
11. Jung JJ (2010) Integrating social networks for context fusion in mobile service platform. J Univers Comput Sci 16(15):2099–2110
12. Jung JJ (2012) Evolutionary approach for semantic-based query sampling in large-scale information sources. Inf Sci 182(1):30–39
13. Jung JJ (2012) Attribute selection-based recommendation framework for short-head user group: an empirical study by MovieLens and IMDB. Expert Syst Appl 39(4):4049–4054
14. Jung B, Sukhatme GS (2004) Detecting moving objects using a single camera on a mobile robot in an outdoor environment, in: Int Conf on Intell Autonom Syst, pp. 980–987
15. Lee M, Nevatia R (2006) Human pose tracking using multi-level structured models, in: Proc. of the ECCV, pp. 368–381
16. Leibe B, Seemann E, Schiele B (2005) Pedestrian detection in crowded scenes, in: Proc. of the IEEE Conf Comput Vis Pattern Recogn, pp. 878–885
17. Mikolajczyk C, Schmid C, Zisserman A (2004) Human detection based on a probabilistic assembly of robust part detectors, in: Proc. of the ECCV, pp. 69–82
18. Mohan A, Papageorgiou C, Poggio T (2001) Example-based object detection in images by components. IEEE Trans PAMI 23(4):156–177
19. Papageorgiou C, Poggio T (2000) A trainable system for object detection. Int J Comput Vis 38(1):15–33
20. Peter JR, Tu H, Krahnstoever N (2005) Simultaneous estimation of segmentation and shape, in: IEEE Conf on Comput Vis Pattern Recogn, pp. 271–278
21. Ramanan D, Forsyth DA, Zisserman A (2005) Strike a pose: tracking people by finding stylized poses, in: Proc. of IEEE Conf Comput Vis Pattern Recogn, pp. 271–278
22. Shashua A, Gdalyahu Y, Hayun G (2004) Pedestrian detection for driving assistance systems: single-frame classification and system level performance, in: Proc. of IEEE Intell Vehicle Symp, pp. 1–6
23. Shi J, Tomasi C (1994) Good features to track, in: Proc. of IEEE Conf Comput Vis Pattern Recogn, pp. 593–600
24. Sigal L, Bhatia S, Roth S, Black MJ, Isard M (2004) Tracking loose-limbed people, in: Proc. of IEEE Conf Comput Vis Pattern Recogn, pp. 421–428
25. Smith K, G.-Perez D, J.-Marc O (2005) Using particles to track varying numbers of interacting people, in: Proc of IEEE Conf Comput Vis Pattern Recogn, pp. 962–969
26. Srinivasan S, Chellappa R (1997) Image stabilization and mosaicking using the overlapped basis optical flow field, in: Proc. of IEEE Int Conf Image Process, pp. 420–425
27. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features, CVPR, pp. 511–518
28. Viola P, Jones M, Snow D (2003) Detecting pedestrians using patterns of motion and appearance, in: Proc. of IEEE Int Conf Comput Vis, pp. 734–741
29. Wu B, Nevatia R (2006) Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors. Int J Comput Vis 75(2):247–266
30. Zhao L, Nevatia R (2004) Tracking multiple humans in crowded environment. IEEE Trans PAMI 26 (9):1208–1221
31. Zhao T, Nevatia R (2004) Tracking multiple humans in crowded environment, in: Proc. of IEEE Conf Comput Vis Pattern Recogn, pp. 406–413
32. Zhao L, Thorpe CE (2000) Stereo- and neural network-based pedestrian detection. IEEE Trans Intell Trans Syst 1(3):148–154



JongSeok Lim received the B.S. degree in physics from Kyemyung University and the M.S. degree in Computer Science from Daegu Catholic University and the Ph.D. degree in Computer Engineering from Yeungnam University in 2004. He is currently a visiting professor of Department of Computer Engineering at Yeungnam University. His research interests include computer vision, image processing, pattern recognition, etc.



WookHyun Kim received his B.S. and M.S. degrees in electronic engineering from Kyungbuk National University in 1981, and 1983, respectively. He received his Ph.D. degree from University of Tsukuba in 1993. He is currently a professor of Department of Computer Engineering at Yeungnam University. His research interests include computer vision, image processing, pattern recognition, etc.