Real-Time Illegal Parking Detection in Outdoor Environments Using 1-D Transformation

Jong T. Lee, *Student Member, IEEE*, M. S. Ryoo, *Member, IEEE*, Matthew Riley, and J. K. Aggarwal, *Life Fellow, IEEE*

***Abstract*— With decreasing costs of high-quality surveillance systems, human activity detection and tracking has become increasingly practical. Accordingly, automated systems have been designed for numerous detection tasks, but the task of detecting illegally parked vehicles has been left largely to the human operators of surveillance systems. We propose a methodology for detecting this event in real time by applying a novel image projection that reduces the dimensionality of the data and, thus, reduces the computational complexity of the segmentation and tracking processes. After event detection, we invert the transformation to recover the original appearance of the vehicle and to allow for further processing that may require 2-D data. We evaluate the performance of our algorithm using the i-LIDS vehicle detection challenge datasets as well as videos we have taken ourselves. These videos test the algorithm in a variety of outdoor conditions, including nighttime video and instances of sudden changes in weather.**

***Index Terms*— Machine vision, surveillance, tracking, video signal processing.**

I. Introduction

THE PROBLEM of tracking vehicles in different scenarios has been studied widely due to its applicability to numerous practical situations. One significant application is the problem of detecting vehicles that have been parked illegally in various locations. A system capable of accurate real-time detection of this nature would serve to automate and greatly improve surveillance systems and would be of great value to those monitoring both public and private locales.

In this paper, we present an algorithm for the automated detection of illegally parked vehicles in real time. The contribution of this paper is the development of an image transformation that allows us to perform event detection computations quickly without compromising accuracy. The simplicity and efficiency of our algorithm make our system potentially applicable in a real-time low-power embedded system. Additionally, we construct the entire real-time system in transformed domain and evaluate its performance with other comparable systems.

The proposed algorithm consists of four general stages: projection, segmentation, tracking, and reconstruction. These stages are processed step by step. In the *projection* stage we apply a 1-D transformation to the source video data, thus preparing the frames for further processing at a reduced computational complexity. Next, foreground blobs that represent vehicles are *segmented* in each 1-D projected image. The activities of the segmented blobs are then *tracked*, frame by frame, as the video progresses. Finally, *reconstruction* is performed for those blobs that have been flagged as being illegally parked vehicles. The final result after reconstruction is a 2-D image of the illegally parked vehicle that can be used for documentation or postprocessing.

In the projection stage, we first estimate the background using an adaptive Gaussian mixture model to detect foreground pixels in No-Parking (NP) zones. Here, NP zones are any locations along the road in which vehicles are not permitted to be parked freely as shown in Fig. 1. Next, we perform an image projection that transforms the 2-D data of each NP zone into representative 1-D data. This transformation accomplishes two tasks. The first is that the transformation reduces the time complexity of the segmentation and tracking tasks. The computing time for segmenting foreground objects in the NP zone is reduced from quadratic to linear time with respect to the length of the images, and tracking is simplified significantly by restricting vehicle motion to a single dimension. The second accomplishment is that the transformation accounts for the variation in size of the vehicle as it moves toward or away from the camera. Utilizing the knowledge of the camera's orientation with respect to the NP zones, the transformation produces a representation in which the size of a moving vehicle will be consistent across each frame of the video.

After transforming an image region to one dimension, the segmentation process relies on simple morphological operations. Tracking is performed by matching the segmented blobs with a cost function. When vehicles are tracked, merging and splitting operations are used to account for vehicle occlusion. Together, segmentation and tracking allow for the automatic detection of vehicles that are parked illegally.

In the final step of the algorithm, we invert the transformation to restore the original appearance of the vehicles for which a special event (e.g., illegal parking) has been detected in one dimension. This process is essential for documentation or further processing, such as vehicle model classification. Thus,

Manuscript received March 3, 2008; revised September 10, 2008. First version published April 7, 2009; current version published July 22, 2009. This paper was recommended by Associate Editor D. Xu.

J. T. Lee and J. K. Aggarwal are with the Department of Electrical and Computer Engineering, University of Texas (UT), Austin, TX 78712 USA (e-mail: jongtaeklee@mail.utexas.edu; aggarwaljk@mail.utexas.edu).

M. S. Ryoo was with the University of Texas (UT), Austin. He is currently with the Electronics and Telecommunications Research Institute, Daejeon, 305-700 Korea (e-mail: mryoo@ece.utexas.edu).

M. Riley was with the University of Texas (UT), Austin. He is now working in Scribd, Inc., San Francisco, CA 94108 USA (e-mail: mriley@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2009.2020249

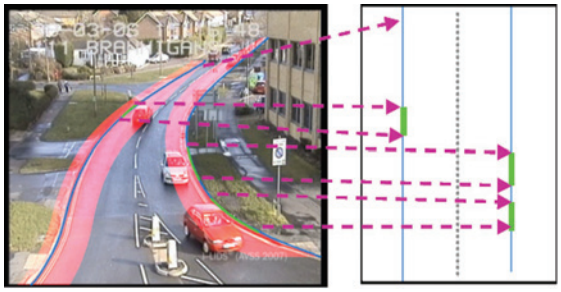


Fig. 1. Transformation of curved representative lines of No Parking zones (on the red regions) to adjusted straight lines.

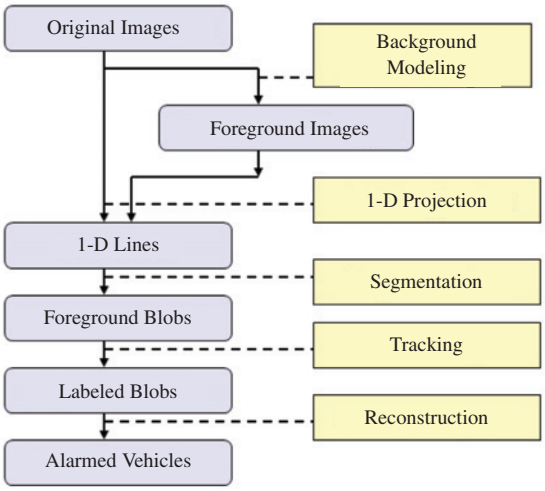


Fig. 2. Framework of the proposed system.

the transformation allows us to simplify the computationally expensive components of the vision system and make decisions in real time, all the while preserving the original 2-D information. This dramatically increases the practical applicability of our system. The frame work is shown in Fig. 2.

Relevant previous work is given in Section II. In Section III, the proposed method is described. Details of the 1-D projection (Section III-A), segmentation (Section III-B), tracking (Section III-C), and reconstruction (Section III-D) processes are also provided. We present tracking results with comments in Section IV. The paper concludes with Section V.

II. Previous Work

There has been considerable amount of research related to special tasks of vehicle detection. Vehicle detection has been attempted using a stationary single camera [1]–[4], stationary multiple cameras [5], a moving single camera [6]–[8], and moving multiple cameras [9]. The systems for vehicle detection using stationary cameras [1]–[5] are the most typical and our system falls into this category. The systems for vehicle detection with moving cameras [6], [7], [9] use various methods such as template matching, temporal differencing, and specific pattern searching instead of using a background model. Rajagopalan and Chellappa [8] applied an image stabilization algorithm to detect motion regions. After stabilizing the images, they processed thresholding and morphological operations to detect motion regions. Additional related papers have been reviewed thoroughly in [10].

Background subtraction has most widely been used for foreground detection with one or multiple stationary cameras. In background subtraction, it is desirable to obtain good background models. Many papers address this problem using an adaptive background mixture model [11], which works well in outdoor environments. Zivkovic [12], [13] proposed an improved adaptive background mixture model for background subtraction. The algorithm provides decent performance in various environments, and works especially well for shadow removal. Morphological operations are performed on the foreground to reduce errors in foreground and background models after processing [14]. We use the methods from [12]–[14] for successful low-level processing.

Our system performs the 1-D transformation directly based on the original image since the roads in the given sequences are not straight lines. In the case where the roads are straight lines, image rectification process such as [15] and [16] could be useful to perform the 1-D transformation automatically.

For vehicle tracking, Gupte *et al.* [2] tracked blobs using an association graph to connect objects in continuous frames. They used Kalman filtering to predict the position of a vehicle in each consecutive frame.

Many papers have tried to handle occlusions among vehicles by segmenting occluded blobs. Song and Nevatia [3] were able to overcome difficulties arising from occlusions and track vehicles individually. In order to solve the problem, they used Bayesian formulation and Markov chain Monte Carlo (MCMC)-based posterior probability estimation. Jun *et al.* [4] proposed a method of segmentation for vehicles under severe occlusions. Their system first finds feature points of vehicles using scale-invariant feature transform (SIFT) and tracks those features to compute motion vectors. Oversegmented image fragments are then clustered based on motion vectors of the fragments, and occluded vehicles are separated finally. On the other hand, Haritaoglu *et al.* [17] tracked and defined the status of objects as splitting and merging to handle occlusions. They tracked objects under different conditions: one-to-one, one-to-many, many-to-one, one-to-zero, and zero-to-one matching instead of trying to segment occluded blobs.

There are several papers related to event detection using machine vision techniques for surveillance. Ivanov and Bobick [18] described a system which is able to recognize gestures and to interpret activities of multiple objects for visual surveillance using a probabilistic syntactic approach. Joo and Chellappa [19] recognized normal events and detected abnormal events using attribute grammars. They tested their approach in video from a business parking lot. Bhargava *et al.* [20] detected abandoned objects in crowded environments such as airports. Their system is able to detect abandoned baggage in the i-LIDS datasets by using a backtracking algorithm. More complicated events have been tried to be recognized using dynamic Baysian network based methods [21]–[23].

Specialized vehicle activities, such as detection of illegally parked vehicles that we are tackling in this paper, have not been studied in depth with one possible exception of [24],

which presents a system that detects and warns of an illegally parked vehicle. In 2007, however, the i-Lids vehicle detection challenge dataset was released and a large number of the papers provided distinctive solutions on this challenge: Bevilacqua and Vaccari [25], Boragno *et al*. [26], Guler *et al*. [27], Lee *et al*. [28], Porikli [29], and Venetianer *et al*. [30].

In the previous conference version of our paper [28], the system was able to detect illegal parking in given three sequences using a very simple and efficient method. In this paper, we extend the projection process and tracking algorithm to improve the system. We show that the current system is more robust with experiments of several longer sequences on the system. The comparison among results of the current system, those of the previous version of our system, and those of other related works will be presented in the experimental section.

III. Proposed Method

Our proposed method consists of five consecutive processes: background modeling, 1-D projection, segmentation, tracking, and reconstruction. After one process is completed, the next step is executed.

A. Projection

In this section, we present an algorithm that converts regions in NP zones in the original image to 1-D vectors. Before applying the 1-D projection, we perform background modeling in order to estimate foreground pixels in the NP zones.

The previous version of our system [28] performed 1-D projection first, followed by background modeling using color information on the 1-D line. Here, the color information at each pixel in the 1-D line consists of the mean and variance of the values of a set of pixels which have been mapped to that pixel by the transformation. The system required very small amounts of computation, but the system suffered if the weather condition changed or occlusions occurred. In order to make the system more robust, our system presented in this paper performs background modeling before projecting the image into the 1-D lines.

The background modeling step is the most computationally expensive phase of the algorithm. To help the system process faster, we perform the background modeling only on the NP zones as shown in Fig. 3.

Here, the NP zones are pre-described for each given video sequence. We use corresponding masks to extract these zones in each image. Usually, NP zones lie along the sides of roads, so we focus on a typical condition where the road is curved, and therefore the zones of interest are also curved.

The 1-D lines can represent these NP zones successfully under this condition. In order to detect and track vehicles in the NP zones more efficiently and reliably, our system transforms these zones into our 1-D representation. Using our proposed transformation, vehicles moving at a constant speed along the direction of the road are transformed into blobs moving along a straight line with a constant velocity. Each vehicle is normalized in order to have a constant size across every frame.

To this end, we first define a representative line for each NP zone, which is composed of a set of pixels. This set of pixels

(a)

(b)

Fig. 3. (a) Sample original image and (b) result of background modeling. Black pixels are background pixels, red pixels are foreground pixels in NP zones, and gray pixels are foreground pixels out of NP zones. (This figure is best viewed in color.)

on the representative line is mapped onto one straight line, according to each representative pixel's location, as shown in Fig. 1. In order to map other pixels in the region of the NP zone onto the straight line, we first locate the closest pixel on the representative line for each pixel in the NP zone. Those pixels are mapped to the same point on the straight line as their closest pixel on the representative line. We then calculate RGB color histogram of each representative pixel. This process is explained in more depth below.

Let $u = \{u_i | 1 \leq i \leq n\}$ be the ordered set of pixels representing a line of an NP zone in the 2-D image as shown in Fig. 4. Now, we first define how the transformed coordinate of each pixel u_i on the representative (curved) line u is calculated. u is transformed into a new straight line in the form of a 1-D vector which only has vertical location information as

$$Coordinate(u_1) = 0 \tag{1}$$

$$Coordinate(u_{i+1}) = Coordinate(u_i) + Distance(u_{i+1}, u_i). \tag{2}$$

$Coordinate(u_i)$ is the new coordinate of u_i on the straight line, and $Distance(u_{i+1}, u_i)$ represents the normalized

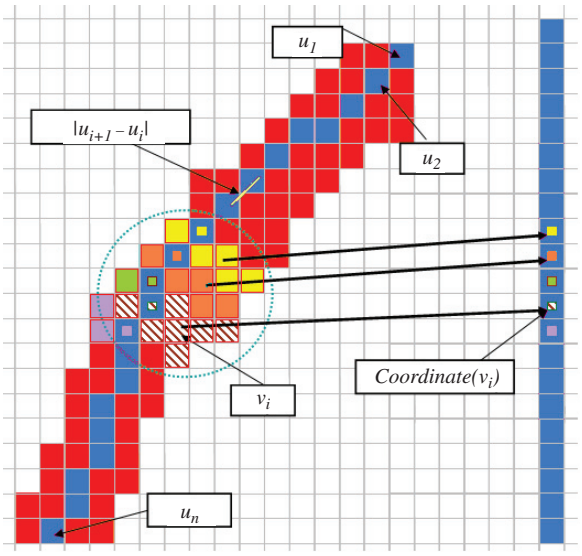


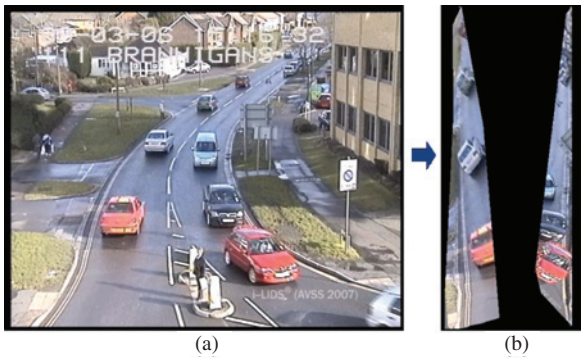
Fig. 4. Image with grid to show a process to define the ordered set of pixels u and to find coordinates of pixels on an NP zone. A sequence of blue pixels corresponds to the representative line, and a sequence of red pixels corresponds to the NP zone. (This figure is best viewed in color.)

Fig. 5. Same vehicles represented in the different places. White vehicle on the left image looks larger than the vehicle on the right image because of perspective projection of the camera.

distance between the two adjacent pixels u_{i+1} and u_i in the original image, such that

$$Distance(u_{i+1}, u_i) = |u_{i+1} - u_i| \times \frac{mean_{c,1\le i\le n}(Length_c(u_i))}{mean_c(Length_c(u_i))}. \tag{3}$$

This *Distance* function is designed such that the lengths of transformed vehicles are similar. On the right-hand side of (3), the numerator $|u_{i+1} - u_i|$ is the distance between pixel u_{i+1} and u_i in the original image, which is distorted by a camera as shown in Fig. 5. To compensate for camera distortion errors, the denominator should be inversely proportional to the distance of the point u_i from the camera. This is because objects close to the camera are bigger in the image than those of the same size but further away from the camera. To this end, we track several sample cars(car_c) through a sequence and measure the length of the cars $Length_c(u_i)$ for those cars located on a pixel of the representative line u_i. $Length_c(u_i)$ can be estimated either manually or automatically using the



(a) (b)

Fig. 6. (a) Original image and (b) intermediate straightened translations of two No Parking zones.

information of the width and height of car blobs. With this length information, we are able to estimate the normalized distance as shown in (3). The size of the objects represented in the transformed 1-D vector can be normalized. As shown in Fig. 6(a) and (b), the size of all similarly sized vehicles at different locations in the original image becomes similar on the projected image.

After obtaining the new coordinates on the 1-D vector line, we transform the whole image region of a NP zone into the same vector representation as shown in Fig. 4. For each pixel v in the image region of an NP zone, we find the corresponding coordinate on the representative line from u_i, which is the closest pixel to v among all pixels on the representative line as shown in (4)

$$Coordinate\,(v) = Coordinate\left(\arg\min_{u_i \in u}(Distance(v, u_i))\right). \tag{4}$$

Next, the histogram of RGB color is calculated along a set of foreground pixels on an NP zone in two dimensions $\{v\}$ such that $Coordinate(\{v\})$ equals $Coordinate(u_s)$, where us is the closest pixel to the set of pixels $\{v\}$. The RGB color histogram will be used for tracking in daytime sequences after the segmentation part is completed.

With a similar method, we classify each pixel in u, which is set of pixels in 1-D, as a foreground pixel or a background pixel as follows:

$$\begin{aligned} &F(u_i) \\ &= 0 \text{ if} \frac{\#\text{ of foreground pixels in}\{v|Coordinate(v) = Coordinate(u_i)\}}{\text{of all pixels in } \{v|Coordinate(v) = Coordinate(u_i)\}} \\ &\quad < Threshold \\ &= 1 \text{ (foreground), Otherwise.} \end{aligned}$$

As a result, the image region in a 2-D space associated with the NP zone is converted into a 1-D vector of color histogram.

B. Segmentation

After our system performs the 1-D projection, we are able to obtain two types of information for each pixel in the 1-D vector: the foreground/background classification, and RGB color information. We segment foreground blobs from

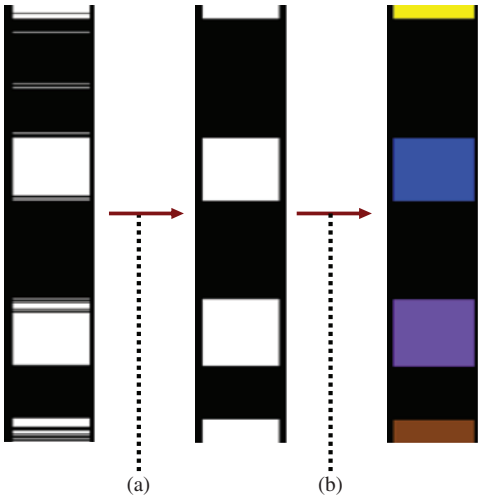


Fig. 7. Segmentation process through (a) morphological operation and (b) connected component analysis.

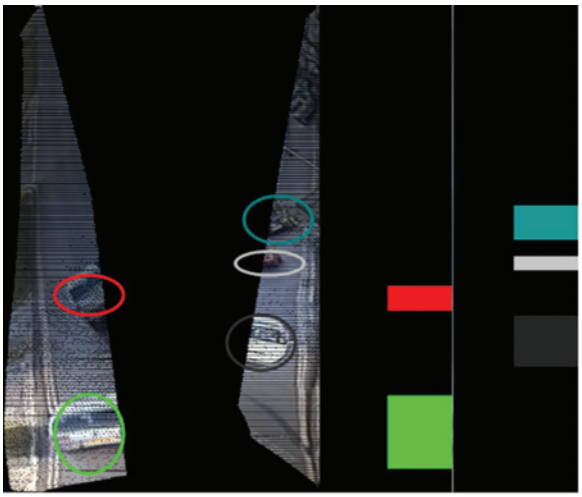


Fig. 8. Result of segmentation. Extracted blobs on the right hand represent individual objects. (This figure is best viewed in color.)

foreground pixels on the 1-D vector. To do this, we first perform morphological operations such as dilating, erosion, and filling holes [14] on foreground pixels to remove the noise produced by low-level processing. Then, we use connected component analysis to label the 1-D foreground. Fig. 7 shows the process of segmentation explicitly. Since we are segmenting blobs in the 1-D vector line and not in a 2-D image, the time required to label the blobs decreases. As a result of segmentation, we obtain a set of regions in the foreground that are likely to represent the true foreground objects as shown in Fig. 8. Thus, the time complexity for segmentation is reduced to O(n) in one dimension, whereas the time complexity for segmentation in two dimensions is O(n^2), where n is the length for transformed 1-D line.

Our current segmentation process has two advantages in comparison with the previous version of our system. One is that the background is updated automatically. This means that we do not have to set the background for each scene, which was required for our previous system. As our system updates the background automatically, the system can handle a long video sequence even in the case of severe weather changes.

The other advantage is that our system is more robust on finding exact foreground pixels than our previous system by sacrificing a slight amount of speed. As a result, the processing time on segmentation for each frame has been increased five times from the processing time of the previous system. Decisions are still made in real time while the system is slower.

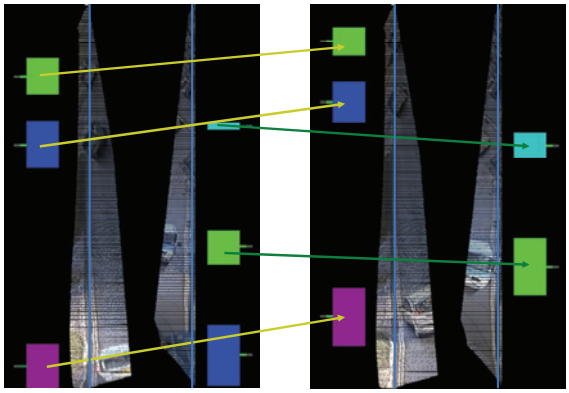


Fig. 9. Matching process in order to track vehicles.

C. Tracking

In this subsystem, tracking is based on matching projected blobs in frame t to projected blobs in frame $t+1$ as shown in Fig. 9. To match blobs, we measure feature distances between blobs on consecutive frames. The total cost function for matching is defined as a weighted sum of the feature distances. The features used are length, location, (R, G, B) color histogram, and velocity. In addition to these features, we also use the duration for which that blob is visible. It is reasonable to expect that a blob tracked longer on the previous frames has a greater possibility that the blob matches one of blobs on the current frame

$$\begin{aligned} &Cost(b_{k_1,t}, b_{k_2,t+1}) \\ &\quad = \sum_{f_j \in Features} Difference_{f_j}(b_{k_1,t}, b_{k_2,t+1}) \\ &\qquad \times w(f_j) - Duration(b_{k_1,t}) \times w(Duration). \end{aligned} \tag{5}$$

Here, $b_{k,t}$ is a kth blob on tth frame. The function $Difference_{fj}$ represents the 1-norm (L_1) distance of feature f_j between the two blobs, $b_{k1,t}$ and $b_{k2,t+1}$. The function *Duration* is dependent on the number of frames where a tracked object is shown. In our system, the *Duration* function is a logarithmic function of the number of frames. The cost function can have negative values. $w(f)$ is the weight of each feature, with location given by the greatest weight. Location is a very reliable feature even in the presence of severe noise. If two blobs are too far from each other, then we can assume that the blobs cannot be matched regardless of how well other features of the blobs match. Therefore, the *Difference* function for the location feature is presented differently compared to

the *Difference* functions for other features as shown in (6)

$$\begin{aligned} &Difference_{\text{location}}(b_{k_1,t}, b_{k_2,t+1}) \\ &\quad = Distance(b_{k_1,t}, b_{k_2,t+1}), \text{ if } Distance(b_{k_1,t}, b_{k_2,t+1}) \\ &\quad < \text{Threshold} \\ &\quad = \infty, \quad \text{otherwise.} \end{aligned} \tag{6}$$

A good match is quickly found for most common vehicle blobs using a noniterative greedy algorithm based on this cost function.

When a blob passes another blob, occlusion may occur. However, it is easier to handle occlusions that occur in a 1-D than in a 2-D image for several reasons. Since all blobs in the 1-D projection can only move in one direction, it is easy to estimate whether the blob is occluded or not. If we are able to recognize blobs merging into each other, we can then detect blob splitting as well.

Our system handles those occlusions differently, depending on the types of occlusions. We categorize two typical types of occlusions between two vehicles as follows: 1) *occlusion caused by a moving vehicle passing a stationary vehicle* and 2) *occlusion between two moving vehicles.*

The first type of occlusions should be handled with the highest priority among all those types of occlusions. The reason is as follows. We are detecting illegally parked vehicles, which must be stationary for a pre-described duration. Handling stationary vehicle-related occlusions will enable the system to track the stationary vehicles more accurately. The system can, therefore, detect illegally parked vehicles with less chance of error. The second reason is that the first type of occlusions is the most typical type and so it can occur more frequently. Most tracking errors occur from this type of occlusions. By handling this type of occlusions, the system can handle most occlusions.

Occlusions between two moving vehicles may occur as well, even though this type of occlusions occurs less frequently compared to the first type of occlusions. What we have noted is that the tracking failures of moving vehicles do not cause the system to fail in finding illegally parked vehicles in most cases. The purpose of our system is not on illegal activity recognition of moving vehicles such as red light violations. So, our system handles this type of occlusions with less priority than the first type of occlusions to concentrate on the improvement of the tracking performance of illegally parked vehicles in our system.

We have to accurately define two terms, *stationary* and *moving* to make the system robust. The terms have been defined based on a blob's location [28], [31]. In [28], a vehicle is counted as *stationary* if its location changes no more than the given threshold for certain frames. This definition is very intuitive and simple for implementation in the system but it may fail to classify the correct status of blobs when the locations of the blobs change a great deal due to a huge amount of noise or occlusion by a large vehicle. Another problem with this definition is that the previous system can classify a vehicle moving at a very slow speed as a stationary vehicle.

To fix this kind of error which may occur due to the previous definition, we distinguish between stationary vehicles and moving vehicles using a different methodology. The decision to distinguish the status is not based on the locations of the vehicle blobs but on all pixels of the vehicle blobs on a 1-D line. No matter how severely the occlusions happen, there should be enough number of pixels u, detected continuously for certain duration on the same stationary blob. Since an illegally parked vehicle is supposed to occupy a certain region, the region should not change at all as long as the vehicle is parked. As our system uses a 1-D line in this stage, the foreground blob for each vehicle will be represented by a set of foreground pixels in a 1-D line. If the actual length of a set of pixels on the foreground blob, where the pixels of the blob are detected for certain duration or more, is longer than the given threshold, then we define the blob as a stationary blob. Otherwise, we define the blob as a moving blob as shown in (7)

$$\begin{aligned} &Stationary(b) \\ &\quad = 1 \text{ (stationary), if } \sum_{u_i \in U_S} Distance(u_i) \\ &\quad > length\_threshold \\ &\quad = 0 \text{ (non-stationary), Otherwise} \end{aligned} \tag{7}$$

As defined in the Section III-A, $Distance(u_i)$ denotes $Distance(u_{i+1}, u_i)$. U_S is a set of pixels of blob b where the time of occupation of the same blob b is larger than *duration_threshold*. The value of *length_threshold* has been decided for the system to detect vehicles properly. In our system, we set *duration_threshold* to 3 s and *length_threshold* to two-thirds of the average length of vehicles on a 1-D line.

The idea of splitting and merging in [17], [32] motivated our system to handle occlusions. After detecting a stationary vehicle, we focus on the blob as follows.

1) Self-Split/Merge. As a result of noise, the probably parked blob may be split into smaller blobs. As long as these smaller blobs lie within the region of the probably parked car, we merge them into one.
2) Merge. When another blob approaches the probably-parked blob and subsequently merges with it, the current information of the probably parked blob is saved.
3) Split. When the two merged blobs split, we decide which among two blobs is most likely to be the previously probably parked blob by matching them with the blob information saved during the merging process, using the cost function defined in (5).

This algorithm is shown with a flowchart in Fig. 10. Figs. 11 and 12 show an example of Self-Split/Merge event and Merge event, respectively.

The system issues an alert if the probably parked blob remains "stationary" for longer than 60 s, labeling it as an illegally parked car. The time duration for an alert can change for various situations.

D. Reconstruction From 1-D Projection

After tracking objects, for additional processing on tracked objects; such as the classification of objects, it is meaningful to reconstruct the original object images from 1-D blob information. The reconstruction process is simple. Using the

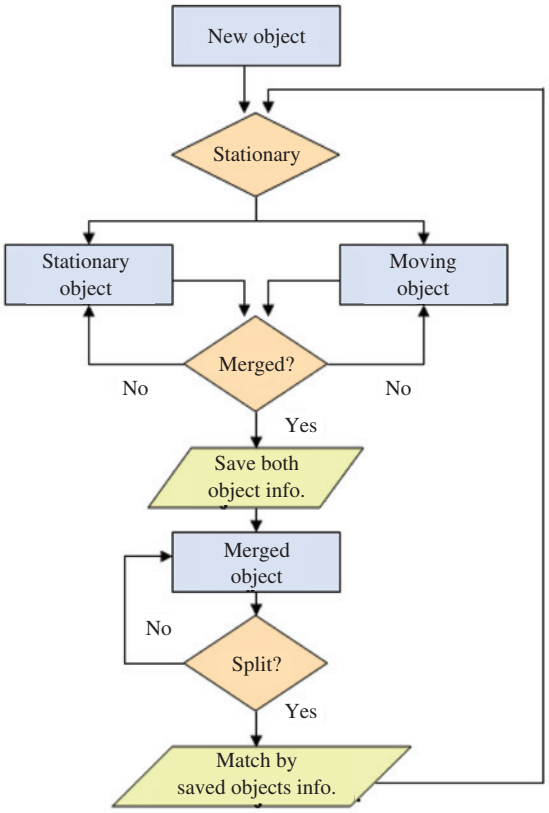


Fig. 10. Flowchart for tracked objects in order to handle occlusions on stationary objects.

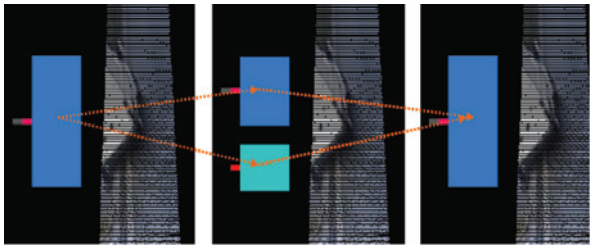


Fig. 11. Example of a self-split/merge event.

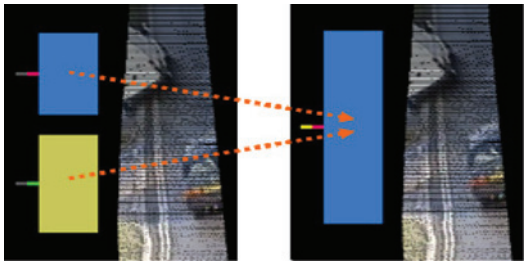


Fig. 12. Example of a merge event between different two vehicles.

length and location of each blob in its 1-D projection, we can find the corresponding part of the curved NP zone in the original image. By dilating the part of the NP zone, related regions can be obtained as shown in Fig. 13.

IV. Experimental Results

We use two datasets for the experiment: the i-LIDS vehicle detection dataset [33] which was produced in London, U.K.,

Fig. 13. Reconstructed image patches of tracked vehicles. The size of the patches can be customized to display as much of the vehicles as desired.

and a set of video sequences we have taken in Austin, TX. These datasets have been chosen for testing our system in various environments. We process six frames per second (frames/s). We have tested at larger frames/s as well, and found that there is no difference in their tracking performance. The frame rate is high enough for tracking illegally parked vehicles because those vehicles are decelerating to park and parked vehicles have zero speed.

We do not want to lose object information by resampling the image, especially in the case of objects whose size in an image is small. Therefore, each frame is processed in its original resolution. Real-time processing of a large image is possible for our system because we work on a 1-D projection line instead of 2-D images for segmentation and tracking. The resolution of the i-LIDS data set and TX dataset are 720×576 and 720×480, respectively. The processing time for each frame is less than 0.2 s in our C++ implementation. Because background modeling on 2-D images is the computationally expensive part of our system, the processing time for each frame has been reduced by focusing on the NP zones.

Our system performs in real time with the described frame rate and processing time. Furthermore, the system is able to perform in real time even with more complex and time-consuming jobs such as vehicle classification in this system.

Our system stores the information of vehicles from the given video sequences when the system tracks the vehicles. However, whenever the tracked vehicles disappear from the scene, the system removes the information stored about them, and does not retain it for the entire process. Our system is therefore adequate for real-time analysis on any long video streams which could be from closed-circuit televisions.

The i-LIDS dataset we used consists of four short video sequences and one long video sequence. The time duration for the short video sequences is about 3–4 min and each sequence has one illegal parking event. The play time of the long video sequence is about 18 min and the long video sequence consists of four illegal parking events.

Results for all four daytime sequences in i-LIDS are accurate. We not only detect the illegally parked vehicles correctly without any false positive detection but also measure the durations of the illegal parking events precisely.

The other dataset which we have taken in Austin, TX, consists of two video sequences. The time durations for two video sequences are 12 and 8 min, and each sequence consists

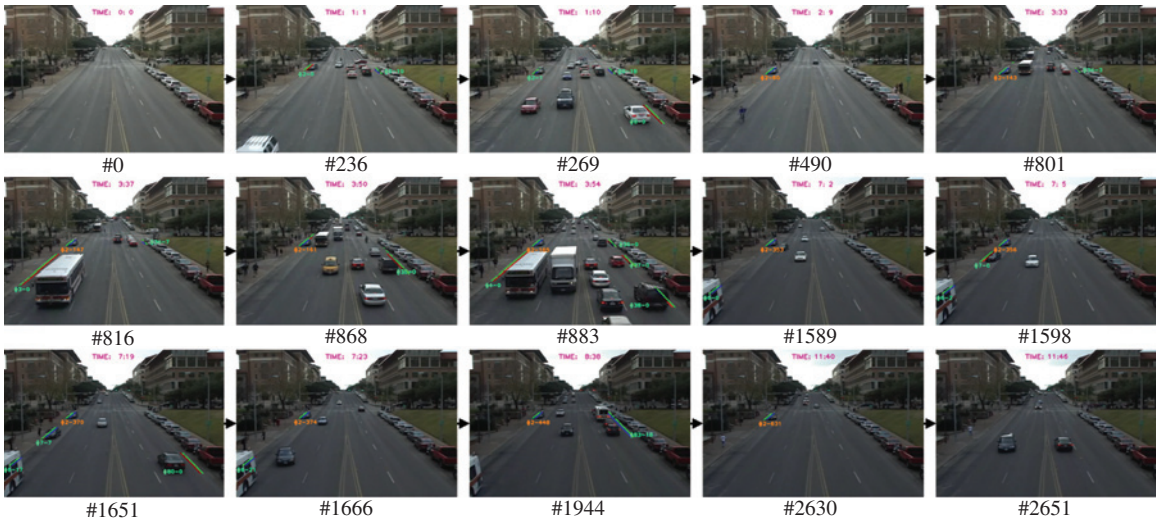


Fig. 14. Image sequence we have taken in Austin, TX, showing tracking results. A vehicle detected as illegally parked is labeled by orange color. Other vehicles in NP zones are labeled by cyan color. Between #236 and #269, one vehicle comes in an NP zone and stays at a fixed location. Between #236 and #490, the system counts frames for 1 min to raise alarm at the frame #490. Between #490 and #2630, the illegally parked vehicle is successfully detected and tracked. Merging and splitting events occur in the intervals of [#801, #816] and [#1589, #1598]. (This figure is best viewed in color.)

TABLE I
ALARM NOTIFICATION RESULTS ON THE I-LIDS DATASET AND OUR DATASET

Sequence (Event)	Length	Start Time (min, s)		Duration (min, s)		Avg Err. (s)
		Ground truth	Our result	Ground truth	Our result	
Easy	Short (i-LIDS)	02:48	02:51	00:27	00:27	6.25
Medium		01:28	01:33	00:19	00:19	
Hard		02:12	02:16	00:21	00:18	
Night		03:25	03:25	00:15	00:11	
1st event	Long (i-LIDS)	06:10	06:12	00:30	00:25	3.5
2nd event		08:12	08:14	00:13	00:12	
3rd event		12:47	12:47	00:21	00:23	
4th event		16:10	16:11	01:04	01:06	
1st event	Medium (1st, TX)	02:06	02:10	09:32	09:34	12
1st event	Medium (2nd, TX)	04:41	04:41	02:21	02:35	
2nd event		04:41	failed	01:24	failed	

Each short sequence has one illegal parking event. Long sequence has four illegal parking events. Two medium sequences from our detaset have one and two illegal parking events, respectively.

TABLE II
TRACKING ACCURACY RATE/TIME COST OF OUR SYSTEMS

System	Tracking accuracy (%)		Time (ms per frame)			Total frames/s
	Stopped vehicles	Moving vehicles	BG Modeling	Transformation	Tracking	
1-D (Previous)	99.8	95.2	1.1	70	1.3	~ 10
2-D (Meanshift)	99.0	98.8	110	21	$\sim$900	$\sim$0.8
CURRENT	100	96.2	23	70	31	~ 6

i-LIDS PV_MEDIUM data have been tested for the experiment in this table.

of one and two illegally parking events, respectively. The tracking results of the 12-min-long sequence are shown in Fig. 14. Our system is able to detect two illegally parked vehicles but failed to detect one illegally parked vehicle. The main reason is that two vehicles came to the NP zone together and they parked very close to each other in the NP zone. Because our system was not able to segment vehicles that arrive together, we could detect only one of the two illegally parked vehicles. No false positive was detected in these sequences. Table I shows our results. Average error is calculated by adding the difference times between the result and ground truth of start time and end time.

We have compared our system with our previous conference version system [28] as well as an existing 2-D system that uses traditional meanshift tracking [34]. Table II shows the comparison results of tracking accuracy rate and speed of the systems. Tracking accuracy is calculated only for detected vehicles frame by frame. Meanshift 2-D system is superior at the tracking accuracy on average but the system is not good for tracking stationary vehicles and is the worst in computing speed. The previous system [28] is the best in computing speed but the performance is not as good as the current system; this system will be appropriate for low-power embedded systems. On the other hand, the current system is the most robust among the three systems and works real time with 6 frames/s; this system will be appropriate for general usages.

Table III shows the comparison results of other existing state-of-the art systems. Average error has been measured by adding result-ground truth difference on start time and end time. Especially Boragno *et al.* [26] and Guler *et al.* [27] detect illegal parking in the all of given four sequences of i-Lids challenge data. Our current system performs comparably well with those systems using the simple but efficient method.

For more accurate tracking of vehicles in a nighttime video sequence, our system must be modified to accommodate the

TABLE III
RESULTS OF OTHER APPROACHES ON THE I-LIDS DATASET

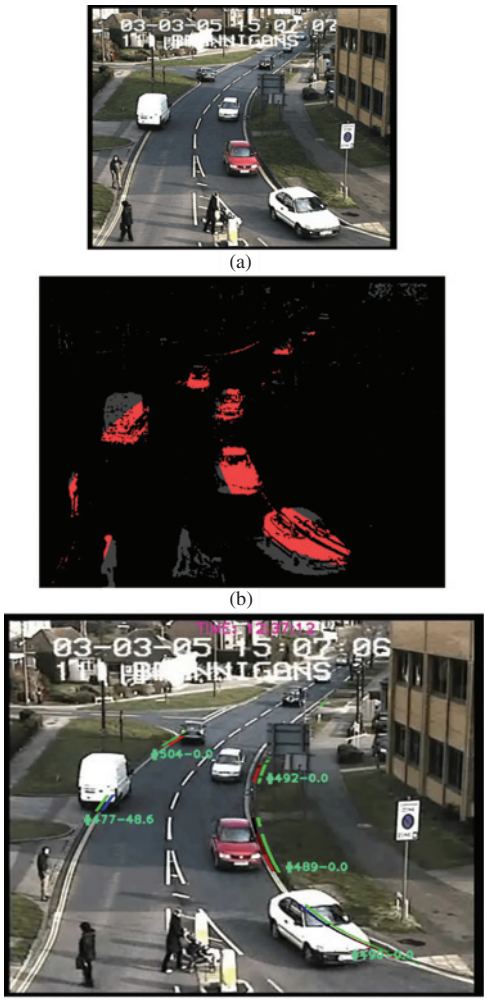
Author	Sequence	Start Time (min, s)		Duration (min, s)		Avg Error (s)
		Ground truth	result	Ground truth	result	
Bevilacqua and Vaccarri [23]	Easy	02:48	N/A	00:27	00:31	4.3* (3/4)
	Medium	01:28	N/A	00:19	00:24	
	Hard	02:12	N/A	00:21	00:25	
	Night	03:25	N/A	00:15	N/A	
Boragno *et al.* [24]	Easy	02:48	02:48	00:27	00:31	5.75 (4/4)
	Medium	01:28	01:28	00:19	00:27	
	Hard	02:12	02:12	00:21	00:24	
	Night	03:25	03:27	00:15	00:19	
Guler *et al.* [25]	Easy	02:48	02:46	00:27	00:32	6.25 (4/4)
	Medium	01:28	01:28	00:19	00:26	
	Hard	02:12	02:13	00:21	00:23	
	Night	03:25	03:28	00:15	00:20	
Lee *et al.* [26] (AVSS '07)	Easy	02:48	02:52	00:27	00:27	12.3 (3/4)
	Medium	01:28	01:41	00:19	00:14	
	Hard	02:12	02:08	00:21	00:29	
	Night	03:25	N/A	00:15	N/A	
F Porikli [27]	Easy	02:48	N/A	00:27	N/A	11.0 (1/1)
	Medium	01:28	01:39	00:19	00:08	
	Hard	02:12	N/A	00:21	N/A	
	Night	03:25	N/A	00:15	N/A	
Venetianer *et al.* [28]	Easy	02:48	02:52	00:27	00:24	10.0 (3/4)
	Medium	01:28	01:43	00:19	00:04	
	Hard	02:12	02:19	00:21	00:15	
	Night	03:25	03:34	00:15	N/A	

All test video sequences are available Online [35]. N/A: non applicable since the system either failed to detect or did not try to detect

Fig. 15. Image from PV_Night sequence. It shows the difficulties in handling night scenes due to headlights and color quality of the video.

effects of headlights. The current system was able to detect the illegally parked car correctly for a precise duration. However, one false positive was also detected at the beginning of the scene, due to the continuous glare of the headlights.

The difficulties of tracking vehicles in a nighttime video sequence depend not only on the headlights but also on the quality of the video scene as shown in Fig. 15. Compared with the daytime video sequence, the nighttime video sequence



(a) (b) (c)

Fig. 16. Three images showing process of our system. (a) A sample original image, (b) Result of background modeling. Background modeling is applied only on the red zones in the system, and (c) Corresponding tracking result after the background modeling, 1-D projection, segmentation, and matching process. All lines represent corresponding vehicles.

has significantly more noise and it is more challenging to distinguish the color of vehicles in the nighttime video sequence. In our system, intensity value histogram is used in the nighttime video sequences instead of using RGB color histogram because of color ambiguity. Therefore, a specialized video camera for activity analysis on night scenes will be desirable for future systems.

Our system is able to detect most of illegally parked vehicles in the given video sequences with a common stationary video camera. Start times and durations of the alarm notification on illegally parked vehicles are very accurate as shown in Table I. In addition, our system is successfully able to run in real time under various conditions in different locations. Fig. 16 shows sample processed images in our system, which consist of an input image, its foreground image, and its tracking result.

V. CONCLUSION

We are able to successfully detect illegally parked vehicles by accurately tracking all the vehicles in the scene. The

proposed algorithm, which is based on the 1-D projection, can be implemented in real time and is effective even in poor scene conditions. The algorithm benefits greatly from the decreased complexity, allowing us to use a more time-consuming segmentation and tracking procedure. Finally, the 1-D transformation can be reversed, allowing us to reconstruct the original appearance of the objects and thereby enabling future processing steps that require a 2-D representation.

REFERENCES

[1] S. Gupte, O. Masoud, and N. P. Papanikolopoulos, "Vision-based vehicle classification," in *Proc. IEEE Conf. Intell. Transp. Syst.*, Oct. 2000, pp. 46–51.

[2] S. Gupte, O. Masoud, R. Martin, and N. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Trans. Intell. Transportation Syst.*, vol. 3, no. 1, pp. 37–47, Mar. 2002.

[3] X. Song and R. Nevatia, "Detection and tracking of moving vehicles in crowded scenes," in *Proc. IEEE Workshop Motion Video Computing 2007*, Austin, TX, Feb. 2007, pp. 4–4.

[4] G. Jun, J. K. Aggarwal, and M. Gokmen, "Tracking and segmentation of highway vehicles in cluttered and crowded scenes," in *Proc. IEEE Workshop Applicat. Comput. Vision (WACV)*, Copper Mountain, CO, 2008, pp. 1–6.

[5] Z. Kim and J. Malik, "Fast vehicle detection with probabilistic feature grouping and its application to vehicle tracking," in *Proc. IEEE 9th Int. Conf. Comput. Vision*, Nice, France, 2003, pp. 524–531.

[6] M. Betke, E. Haritaoglu, and L. Davis, "Multiple vehicle detection and tracking in hard real-time," in *Proc. IEEE Intell. Vehicles Symp.*, 1996, pp. 351–356.

[7] M. Betke, E. Haritaglu, and L. Davis, "Real-time multiple vehicle detection and tracking from a moving vehicle," *Mach. Vision and Applicat.*, vol. 12, no. 2, pp. 69–83, Aug. 2000.

[8] A. Rajagopalan and R. Chellappa, "Vehicle detection and tracking in video," in *Proc. IEEE Int. Conf. Image Process.*, Vancouver, BC, 2000, pp. 351–354.

[9] D. M. Bertozzi, A. Broggi, A. Fascioli, and S. Nichele, "Stereo vision-based vehicle detection," in *Proc. IEEE Intell. Transportation Symp.*, 2000, pp. 39–44.

[10] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 28, no. 5, pp. 694–711, May 2006.

[11] C. Stauffer, W. E. L. Grimson, "Adaptive background mixture models for real-time tracking: Background modeling," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, vol. 2. Fort Collins, CO, 1999, pp. 246–252.

[12] Z. Zivkovic and F. van der Heijden, "Recursive unsupervised learning of finite mixture models," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 26, no. 5, pp. 651–656, May 2004.

[13] Z. Zivkovic, "Improved adaptive gausian mixture model for background subtraction," in *Proc. Int. Conf. Pattern Recognition*, vol. 2. 2004, pp. 28–31.

[14] M. L. Comer and E. J. Delp, "Morphological operations for color image processing," *J. Electron. Imaging*, vol. 8, no. 3, pp. 279–289, Jul. 1999.

[15] Z. Chen, C. Wu, and H. T. Tsui, "A new image rectification algorithm," *Pattern Recognition Lett.*, vol. 24, pp. 251–260, Jan. 2003.

[16] J. Zhou and B. Li, "Exploiting vertical lines in vision-based navigation for mobile robot platforms," in *Proc. IEEE Int. Conf. Acoustics, Speech Signal Process.*, vol. 1. 2007, pp. I-465–I-468.

[17] I. Haritaoglu, D. Harwood, and L. Davis, "W4S: A real time system for detecting and tracking people in 2 1/2-D," in *Proc. Eur. Conf. Comput. Vision*, vol. 1406. Freiburg, Germany, 1998, pp. 877–892.

[18] Y. A. Ivanov and A. F. Bobick, "Recognition of visual activities and interactions by stochastic parsing," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 22, no. 8, pp. 852–872, Aug. 2000.

[19] S. W. Joo and R. Chellappa, "Attribute grammar-based event recognition and anomaly detection," in *Proc. Conf. Comput. Vision Pattern Recognition Workshop*, 2006, pp. 107–107.

[20] M. Bhargava, C.-C. Chen, M. S. Ryoo, and J. K. Aggarwal, "Detection of abandoned objects in Crowded environments," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, London, U.K., 2007, pp. 271–276.

[21] S. Yifan, Y. Huang, D. Minnen, A. Bobick, and I. Essa, "Propagation networks for recognition of partially ordered sequential action," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, vol. 2. 2004, pp. II-862–II-869.

[22] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui, "Learning and detecting activities from movement trajectories using the hierarchical hidden Markov model," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognition*, vol. 2. 2005, pp. 955–960.

[23] T. Xiang and S. Gong, "Beyond tracking: Modelling activity and understanding behavior," *Int. J. Comput. Vision (IJCV)*, vol. 67, no. 1, pp. 21–51, Apr. 2006.

[24] K. Morimoto, "System for detecting and warning an illegally parked vehicle," U.S. Patent 5 343 237, Aug. 30, 1994.

[25] A. Bevilacqua and S. Vaccari, "Real time detection of stopped vehicles in traffic scenes," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, London, U.K., 2007, pp. 266–270.

[26] S. Boragno, B. Boghossian, J. Black, D. Makris, and S. Velastin, "A DSP-based system for the detection of vehicles parked in prohibited areas," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, 2007, pp. 260–265.

[27] S. Guler, J. A. Silverstein, and I. H. Pushee, "Stationary objects in multiple object tracking," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, London, U.K., 2007, pp. 248–253.

[28] J. T. Lee, M. S. Ryoo, M. Riley, and J. K. Aggarwal, "Real-time detection of illegally parked vehicles using 1-D transformation," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, 2007, pp. 254–259.

[29] F. Porikli, "Detection of temporarily static regions by processing video at different frame rates," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, London, U.K., 2007, pp. 236–241.

[30] P. L. Venetianer, Z. Zhang, W. Yin, A. J. Lipton, P. L. Venetianer, Z. Zhang, W. Yin, and A. J. Lipton, "Stationary target detection using the objectvideo surveillance system," in *Proc. IEEE Int. Conf. Advanced Video Signal Based Surveillance*, London, U.K., 2007, pp. 242–247.

[31] X. Liu and K. Fujimura, "Pedestrian detection using stereo night vision," *IEEE Trans. Vehicular Technol.*, vol. 53, no. 6, pp. 1657–1665, Nov. 2004.

[32] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. and Mach. Intell.*, vol. 22, no. 8, pp. 809–830, Aug. 2000.

[33] *i-Lids dataset for AVSS 2007*.

[34] G. Bradski, "The OpenCV Library," *Dr. Dobb's J.*, Computer Security, Nov. 2000.

[35] Available at ftp://motinas.elec.qmul.ac.uk/pub/iLids/

Jong Taek Lee (S'08) received the B.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2005, and the M.S. degree in Electrical and Computer Engineering, University of Texas (UT), Austin, TX, in 2007. He is currently working toward the Ph.D. degree in electrical and computer engineering at UT, where he is also a Research Assistant in the Computer and Vision Research Center. His research interests include recognition of vehicle activities, classification of vehicles, and detecting events between a vehicle and human.

Michael Sahngwon Ryoo (S'06–M'08) received the B.S. degree in computer science from Korea Advanced Institute of Science and Technology, Daejeon, South Korea, in 2004, and the M.S. and Ph.D. degrees in computer engineering from the University of Texas (UT), Austin, in 2006 and 2008, respectively.

From 2005 until August 2008, he was a Research Assistant in the Computer and Vision Research Center, UT. Currently, he is a Research Scientist at the Electronics and Telecommunications Research Institute, Daejeon, Korea. His research interests include representation and recognition of human activities, tracking of humans and objects, human–computer interfaces, and intelligent environments.

Dr. Ryoo is a member of the IEEE Computer Society.

Matthew Riley received the B.S. and M.S. degrees in electrical engineering from the University of Texas, Austin, in 2006 and 2008, respectively.

He is a Software Engineer working in Scribd, Inc., San Francisco, CA. His research interests include machine vision applications of content-based image analysis.

J. K. Aggarwal (LF'02) has been on the faculty with the Department of Electrical and Computer Engineering, University of Texas (UT), Austin, TX, since 1964, and is currently Cullen Professor of Electrical and Computer Engineering and Director of the Computer and Vision Research Center.

His research interests include computer vision, pattern recognition and image processing focusing on human motion. He received the Senior Research Award of the American Society of Engineering Education in 1992, the 1996 Technical Achievement Award of the IEEE Computer Society, and the Graduate Teaching award of the UT in 1992. More recently, he has been the recipient of the 2004 K. S. FU prize of the International Association for Pattern Recognition, the 2005 Kirchmayer Graduate Teaching Award of the IEEE, and the 2007 Okawa Prize of the Okawa Foundation of Japan. He has authored and edited a number of books, chapters, conference proceedings, and research papers.

Prof. Aggarwal is a Fellow of the International Association of Pattern Recognition and the American Association for the Advancement of Science and a Golden Core member of the IEEE Computer Society.