The 2010 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 18-22, 2010, Taipei, Taiwan

Real-Time Vehicle Detection in Urban Traffic Using AdaBoost

Jong-Min Park, Hyun-Chul Choi, and Se-Young Oh, *Member, IEEE*

Abstract—**This paper proposes a method for detecting vehicles in urban traffic. The proposed method extracts vehicle candidates using AdaBoost. The candidate extraction process was speeded up further, exploiting inverse perspective transform matrix. Then the vehicle candidates were verified by the existence of vertical and horizontal edges. The detected vehicle regions were corrected by the vertical edges and shadow. Our algorithm showed the detection rate of 90.77% in urban traffic under normal lighting condition. The proposed algorithm can also detect vehicles in heavy rain. Our algorithm takes 37.13ms on average to detect vehicles in 320 by 240 images on a laptop computer (Intel ® Core^{TM}2 T7200, 2.00GHz, 1.00GB RAM).**

I. Introduction

VEHICLE detection is a key technology for autonomous vehicle systems and driver assistance systems. In autonomous vehicles, vehicle detection systems can help the vehicles avoid collision with other vehicles. In driver assistance systems, vehicle detection systems can provide drivers with collision warning to help drivers drive safely.

Many algorithms for vehicle detection have been proposed so far and the algorithms exploited various features of vehicles. Vehicle detection systems have high computational requirements as they have to process the input images at real-time to take actions for avoiding collisions. The majority of methods have two basic steps instead of searching the whole images to locate potential vehicles. The first step is hypothesis generation (HG) in which the location of possible vehicles in an image are hypothesized. The second step is hypothesis verification (HV) in which tests are performed to verify the presence of vehicles in an image [1].

Various HG methods have been proposed. Shadows underneath vehicles are one of widely used hypothesis [2], [3]. Another hypothesis of presence of vehicle is vertical and horizontal edges. Vehicles have strong vertical edges on the left and the right sides, and strong horizontal edges at the bottom of vehicles [4], [5].

In HV stage, AdaBoost [6] is widely used as an appearance-based verification method recently. Ayoub *et al.* [7] extracted vehicle candidates based on the gradient of shadow and verified the candidates using AdaBoost. Song *et al.* [8] extracted vehicle candidates using edge-based method. Then they also verified the candidates by AdaBoost. This algorithm assumed that strong horizontal edges of shadows underneath vehicles are tire-road contacts. However the shape, and the intensity of shadow are highly sensitive to lighting and weather conditions, and those assumptions do not always hold.

To detect vehicles in various environments, candidate extraction process should be performed based on the appearance of vehicles. Sun *et al.* used Gabor filters to extract features of vehicles [9]. Han *et al.* extracted features of vehicles using histograms of oriented gradients (HoG) [10]. Then the features were classified by support vector machines (SVMs). However, those methods are not fast enough to run in real-time.

In this paper, we propose a new method for detecting front vehicles in urban traffic. In HG step, we extracted vehicle candidates using AdaBoost. The candidate extraction process was speeded up further by exploiting inverse perspective transform (IPT) matrix. In HV step, the candidates were verified by the existence of vertical and horizontal edges for more accurate detection results. The detected vehicle regions were corrected by shadow and edges. The proposed method extracted vehicle candidates based on the appearance of vehicles. Then the extracted candidates were verified by weak hypotheses, because the candidates extracted by our method were much more correct than other candidate extraction method. Our method can detect vehicles in various environments including rainy day.

Section II describes the proposed algorithm in detail. In section III, the results of the proposed algorithm, detection rate and false detection rates are presented. Analysis of vehicle detection in rainy day is also presented in section III. Section IV presents the conclusion.

II. Vehicle Detection

A. Camera Installation and Calibration

Our objective was to detect vehicles that are situated in front of the ego-vehicle. Therefore, we installed a camera on the front windshield of our experimental vehicle (Fig. 1).

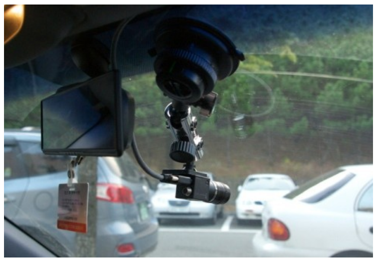
Fig. 1 Camera Installation

Manuscript received February 28, 2010. This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the Core Technology Development for Breakthrough of Robot Vision Research support program supervised by the NIPA(National IT Industry Promotion Agency)
Jong-Min Park, Hyun-Chul Choi and Se-Young Oh are with the Pohang University of Science and Technology (POSTECH), Pohang, Gyeongbuk, 790-784, South Korea (phone: 82-54-279-2880, 2214; fax: 82-54-279-5594; e-mail: {alchemist, dreaming, syoh}@postech.ac.kr).

Then we computed an IPT matrix which describes the relation between the image coordinate and the real world coordinate under the assumption that vehicles are on flat road. First, we specified markers to be used in the computation and measured the real-world coordinates (x, y) of the markers in meter. In this step, we set the origin to be the front-center of the experimental vehicle. Second, we specified the image coordinates (u, v) corresponding to the real-world coordinates of the markers (Fig. 2).

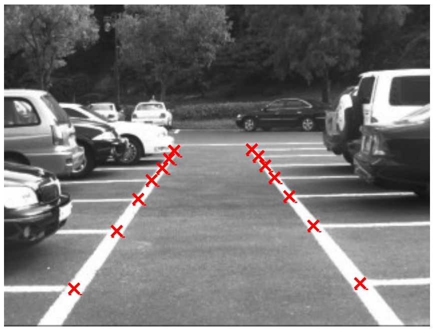
Fig. 2 Markers used in the camera calibration, X: markers used in the calibration.

Then the relation between the real-world coordinate (x, y) and the image coordinate (u, v) can be expressed as:

$$\begin{bmatrix} x \cdot z \\ y \cdot z \\ z \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (1)$$

We can compute the elements of the 3 by 3 IPT matrix, using least squares method. Mathematically, the computation requires only four matches of image and world coordinates. However, we need at least six matching points and the matching points should be widely spreaded over the whole area of interest. If the matching points are concentrated in a small area, the resulting IPT matrix would be over-fitted to the small area and would fail to correctly describe the relation for the entire area of interest. We computed the IPT matrix using 14 correspondences, and estimated the real-world coordinates from image coordinates using expression (1) with the mean square error of 0.068 m (Fig. 3).

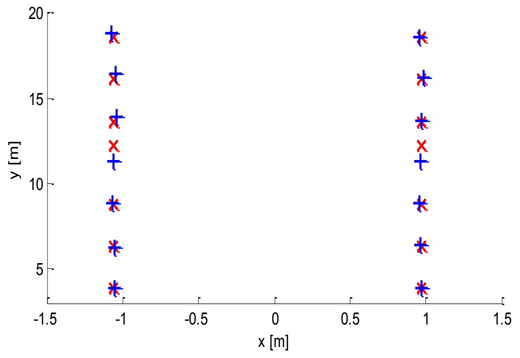

Fig. 3 The result of calibration. x: real world coordinates, +: estimated real world coordinates

B. *AdaBoost Training*

We trained AdaBoost [6] vehicle detector using 24 by 24 training images. The vehicle image set includes rear views of various kinds of vehicles: sedan, bus, truck, and SUV (Fig. 4a). However, we did not considered detection of special purpose vehicles in this research. When we were cropping vehicle images, the bottoms of the bounding boxes were aligned at the position where road and rear tires contact. The left and the right sides of the bounding boxes were aligned at the left and the right sides of vehicles, respectively.

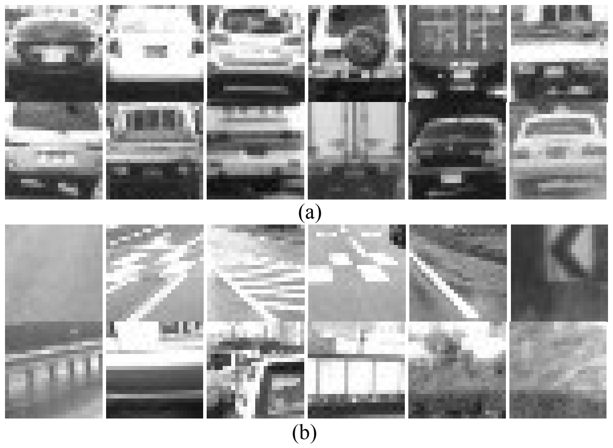

Fig. 4 Examples of training images (a) vehicle images, (b) non-vehicle images

Negative image samples include surface of road, traffic signs on road, parts of vehicles, and many other objects that can be seen on road (Fig. 4b). We cropped 671 vehicle image samples and 24,593 non-vehicle image samples. We trained AdaBoost vehicle detector using 15 haar-like features: original features [6] plus extended features [11] (Fig. 5) .

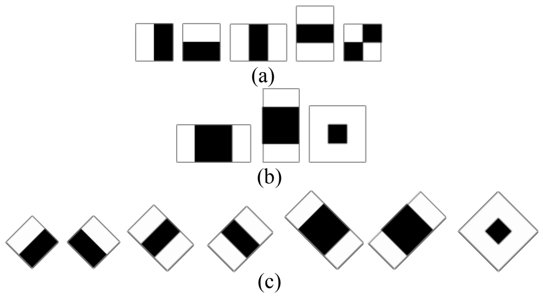

Fig. 5 Haar-like features used to train AdaBoost (a) Original haar-like features (b) upright extended features (c) 45° rotated features

The resulting cascaded AdaBoost classifier has 22 stages with 687 weak classifiers.

C. *Candidate Extraction*

We extracted vehicle candidates that are situated at the distance between 6m and 50m from the experimental vehicle, because of the limited sight of the camera. Vehicles situated beyond 50m are shown too small to be detected by AdaBoost in the input images. The vehicles situated at close distance are so large that they are not completely shown in the input image.

We computed the image coordinate v_{50m} corresponding to 50m in the real world coordinate as:

$$\begin{bmatrix} w\cdot u \\ w\cdot v_{50m} \\ w \end{bmatrix} = PT \begin{bmatrix} 0 \\ 50 \\ 1 \end{bmatrix} \tag{2}$$

where PT is perspective transform matrix and is inverse of IPT. Likewise, the image coordinate v_{6m} corresponding to 6m in the real world coordinate can be computed as:

$$\begin{bmatrix} w\cdot u \\ w\cdot v_{6m} \\ w \end{bmatrix} = PT \begin{bmatrix} 0 \\ 6 \\ 1 \end{bmatrix} \tag{3}$$

Then we scanned the input images with sub-window whose bottom position is between v_{6m} and v_{50m} to locate vehicle candidates using AdaBoost. This scanning process is time-consuming, because AdaBoost searches the entire area of interest with sub-window of various sizes for vehicles. We could speeded up this process, exploiting the facts: the width of vehicles of our interest is ranged from 1.5m to 2.7m; vehicles situated at further distance are located at higher position and their size is smaller than vehicles that are situated at closer distance in the images. We can compute the range of vehicle width $[W_{\min}, W_{\max}]$ at each image coordinate (u,v) using IPT matrix (Table I).

Table I.
Algorithm for computing the range of sub-window size

```
• for $(u,v)$
    Compute the real world coordinate $(x_L, y)$ of $(u,v)$
    $\begin{bmatrix} z\cdot x_L \\ z\cdot y \\ z \end{bmatrix} = IPT \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$
    Compute the minimum and maximum right position
      $x_{R,\min} = x_L + 1.5;\ x_{R,\max} = x_L + 2.7$
    Compute the image coordinate of the right positions
    $\begin{bmatrix} w\cdot u_{\min} \\ w\cdot v_{\min} \\ w \end{bmatrix} = PT \begin{bmatrix} x_{R,\min} \\ y \\ 1 \end{bmatrix};\ \begin{bmatrix} w'\cdot u_{\max} \\ w'\cdot v_{\max} \\ w' \end{bmatrix} = PT \begin{bmatrix} x_{R,\max} \\ y \\ 1 \end{bmatrix}$
    Compute the minimum and maximum window width
      $W_{\min}(u,v) = u_{\min} - u;\ W_{\max}(u,v) = u_{\max} - u$
```

Once the range of vehicle width is computed, we do not need to compute the range again, unless camera position is changed, because the range depends only on IPT matrix.

Then our modified scanning scheme only classifies the sub-windows whose left-bottom position is (u,v) and width W is between $W_{\min}(u,v)$ and $W_{\max}(u,v)$ (Table II).

The size of vehicles widely varies in the images depending on distance. Therefore we set the parameters for sub-window scale: $S_{\min} = 1$, $S_{\max} = 10$, $S_{step} = 1.2$. The vertical and horizontal scanning steps were tuned to be $u_{step} = 2$ and $v_{step} = 1$. The initial sub-window size was 24.

Table II.
Modified scanning scheme

```
for ( $scale = S_{\min}$ ; $scale \le S_{\max}$ ; $scale \times= S_{step}$ )
    $v_{step} \leftarrow S_{step} \times v_{step}$
    $u_{step} \leftarrow S_{step} \times u_{step}$
    $W \leftarrow S_{step} \times W$
    for ( $v = v_{50m}$ ; $v \le v_{6m}$ ; $v += v_{step}$ )
        for ( $u = 0$ ; $u \le W_{image}$ ; $u += u_{step}$ )
            if $W_{\min}(u,v) \le W \le W_{\max}(u,v)$
                classify the sub-window
            else
                skip
```

This modified scanning scheme reduced the entire processing time by factor of 6 in our experiments. This process also performs filtering the extracted vehicle candidates based on their width.

D. Candidate Verification

Even though AdaBoost detected vehicle candidates based on their appearances and the extracted candidates were filtered based on their width in the candidate extraction step, it is still probable that non-vehicle images were detected as vehicles. To reduce the false positive detections, we verified the extracted vehicle candidates based on the facts that vehicles have strong vertical edges on the left and right sides; vehicles have strong horizontal edges at the bottom.

We detected edges using the 3 by 3 Sobel mask as:

$$\begin{aligned} G_x(u,v) &= I(u,v) * S_x \\ G_y(u,v) &= I(u,v) * S_y. \end{aligned} \tag{4}$$

where S_x and S_y denote the Sobel mask, I represents the grayscale input image and $*$ denotes convolution. We detected only vertical and horizontal edges and rejected diagonal edges that vehicles do not have. The condition for edges to be vertical was:

$$\left|\frac{G_y}{G_x}\right| < \frac{1}{3} \tag{5}$$

The condition for edges to be horizontal was:

$$\left|\frac{G_y}{G_x}\right| > 3 \tag{6}$$

We computed the orientations of edge pixels as the absolute values of the slopes, because this method is much faster than arc tangent function.

The edges show that vehicles have strong vertical edges on both the left and the right sides (Fig. 6b) and have strong horizontal edges in many parts of vehicles (Fig. 6c): bumper, rear windshield, and shadow. Therefore the existence of such edges can be important feature of vehicles.

In urban area, however, many buildings exist and they have strong vertical and horizontal edges.

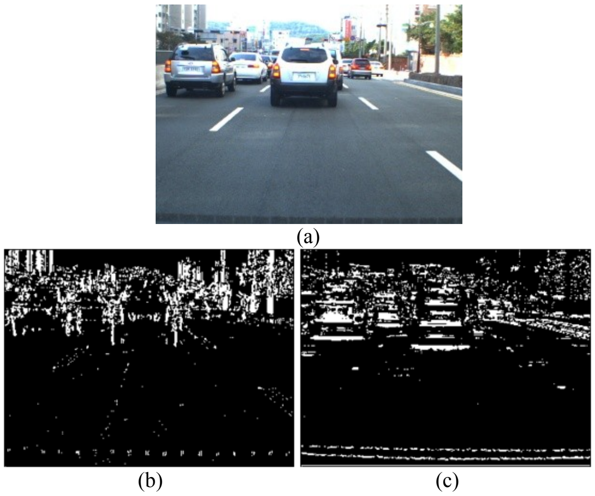

Fig. 6 Detected vertical and horizontal edges (a) original image (b) vertical edges (c) horizontal edges

Those edges can cause false detections to be verified as positive detections. To avoid this, we searched the longest vertical edges in lower regions of the vehicle candidates (Fig. 7a). The regions were centered at the left and the right side of the detected regions. The width and heights were proportional to the width of the detected window size. In our experiments, $f_1 = 0.5$ and $f_2 = 0.25$ were used.

We also searched the longest horizontal edges in the lower region of each vehicle candidate (Fig. 7b). The size of the region was also proportional to the size of the vehicle candidates. In our experiments $f_3 = 0.5$ was used.

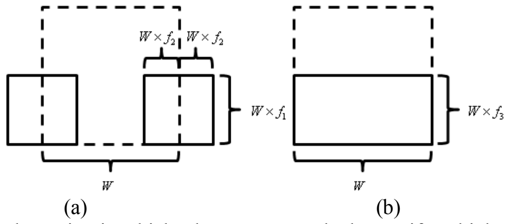

Fig. 7 The region in which edges were searched to verify vehicle candidates, dotted box : a detected region, solid box : regions in which edges were searched

Vehicles that are situated at close distance may have discontinuous edge pixels, because of their detailed texture and distortion caused by the camera. Therefore we allowed the longest edges to have small number of discontinuous edge pixels. In our experiments, the number of maximum discontinuous edge pixels was 5.

We verified candidates that have edges longer than thresholds as vehicles. The threshold for the vertical edges on the left and the right sides was determined to be $W \times T_{vertical}$. Likewise, the threshold length of horizontal edges was determined to be $W \times T_{horizontal}$. In our experiments we used the parameters: $T_{vertical} = 0.25$ and $T_{horizontal} = 0.5$.

E. Clustering Detected Regions

AdaBoost detects many regions around vehicles (Fig. 8a) and the regions should be clustered to represent one region for each vehicle (Fig. 8b).

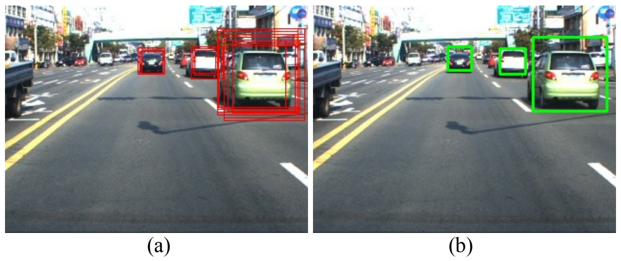

Fig. 8 Detected vehicles (a) detected region (b) average of detected region

Let one detected region have width W_1 and be centered at (x_1, y_1) ; another detected region have width W_2 and be centered at (x_2, y_2) (Fig. 9).

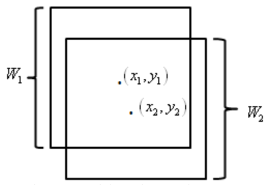

Fig. 9 The parameters that used in clustering

Then the criterions for the two detected regions to belong to the same vehicle are the distance (4) and the size (5).

$$\begin{aligned} |x_1 - x_2| &\le (W_1 + W_2) \times f_{overlap} \\ |y_1 - y_2| &\le (W_1 + W_2) \times f_{overlap} \end{aligned} \tag{4}$$

$$W_{large} \times f_{size} \le W_{small} \le W_{large} \times (2 - f_{size}) \tag{5}$$

where W_{large} is the width of the larger region; W_{small} is the width of the smaller region between W_1 and W_2 . In our experiment, $f_{overlap} = 0.5$ and $f_{size} = 0.5$ were used.

F. Position Correction

The detected vehicles can have incorrect vehicle regions (Fig. 10a). Driver assistance systems and autonomous vehicle systems require exact regions of vehicles to avoid collision. We can also estimate approximated distance from the ego-vehicle to the detected vehicles using IPT matrix if we know the exact bottom positions of detected vehicles. The detected region can be corrected using the edges and the shadows of the detected vehicles (Fig. 10b).

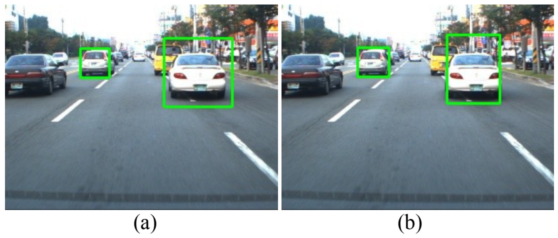

Fig. 10 Position correction (a) incorrect vehicle region (b) corrected vehicle region

The left and the right positions of the detected vehicles were corrected to be the position at which we found the longest vertical edges in the verification step. The bottom position of the detected vehicles can be corrected using shadows underneath the vehicles. Even though, the shape and the intensity of shadow vary depending on both lighting and surface conditions, we can estimate the bottom position form the shadow. First, we performed histogram equalization (Fig. 11b) to reduce the effects of lighting condition. Second, we performed binarization to obtain dark regions of images which include shadows of vehicles (Fig. 11c). Third, we performed filtering of shadow based on the width (Fig. 11d).

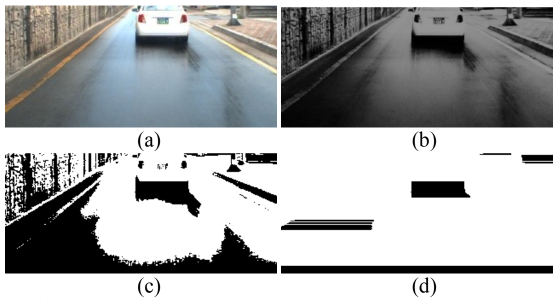

Fig. 11 Shadow estimation (a) input image (b) histogram equalization (c) binarized image (d) estimated shadow

The filtering process scans the binary image in horizontal direction, searching the start position and the end position of dark pixels. If the width is not shorter than the minimum width of vehicles at the position, the dark pixels are marked as shadow (Table III).

Then the bottom position of a detected vehicle is estimated to be the bottom position of the filtered shadow. We have no clue that indicates whether the detected region is either lower or higher than the actual vehicle region. Therefore we scanned the bottom of the filtered shadow around the bottom position of the detected vehicle region. The range in which shadow is searched is determined to be proportional to the height of the detected vehicle. The bottom position is corrected if the estimated bottom position exists in the search range.

Table III.
Shadow filtering algorithm

```
• Histogram Equalization
• Compute binary image $B$
• Estimate shadow
  for $v$
    for $u$
      if $B(u,v)=0$ and shadow = false
        $start = u$;  shadow = true
      else if $B(u,v)\neq 0$ and shadow = true
        $end = u-1$;  shadow = false
        $width = end - start$
        if $width \geq W_{\min}$
          mark pixels between $(start, v)$ and $(end, v)$ as shadow
```

III. Results

Our algorithm was tested on a laptop computer (Intel ® Core^{TM}2 T7200, 2.00GHz, 1.00GB RAM). We detected vehicles in 320 by 240 input images obtained from a CCD camera (Point Grey FL2-03S2C) mounted on the front windshield of the experimental vehicle. Under the test condition, our algorithm took 37.13ms on average to process one frame.

During urban driving, drivers should pay attention to the three closest vehicles in the left, the right and the ego lanes, because those vehicles are probable to collide with the ego-vehicle if drivers are careless. Therefore we counted the detections of those vehicles and all false detections in the three lanes under various lighting conditions (Table IV) in urban traffic (Fig. 12).

Vehicle detection rate was computed as the ratio of the number of positive detections to the number of vehicles (7).

$$\frac{\text{\# of positive detections}}{\text{\# of vehicles}} \tag{7}$$

False detection rate was computed as the ratio of the number of false detections to the number of detections (8). Detections of non-vehicle regions and parts of vehicles in whole region of interest were considered as false detections.

$$\frac{\text{\# of false detections}}{\text{\# of positive detections +\# of false detections}} \tag{8}$$

Table IV.
Vehicle detection rates and false detection rates under various lighting conditions

	Detection rate	False detection rate
Dawn	79.37	7.33
Morning	70.83	16.78
Noon	90.77	7.81
Afternoon	85.57	8.42

The lighting condition at noon (Fig. 12c) was the best for our algorithm. In morning, shadows of buildings and trees are lying on road and vehicles (Fig. 12b), causing lighting conditions to be frequently changed. Therefore the vehicle detection rate and false detection rate was the worst in morning. Under the lighting conditions in dawn (Fig. 12a) and afternoon (Fig.12d), vehicles are shown less clearly in the images, because the input images are shown darker than they are shown at noon. Sometimes, backlight makes the situation worse. Therefore the detection rates were worse than the detection rate at noon.

We can also see that some oncoming vehicles were detected even though front views of vehicles are not included in the training set because the front and rear views of some vehicles are very similar (Fig. 12abc).

The average speed of the experiment vehicle was about 60 km/h and frame rate was 20 frames/sec. During the experiments, no motion-blurred image was taken even though the speed of the experiment vehicle was higher than 100 km/h because of high shutter speed of the camera (0.02ms). Therefore, there was no degradation of detection rate caused by motion blur.

Though camera saturation occurred in the sky region (Fig. 12), the saturation did not affect the detection rate because the region of our interest was road region in images. Vibration of experiment vehicle on float roads also did not degrade the detection rate.

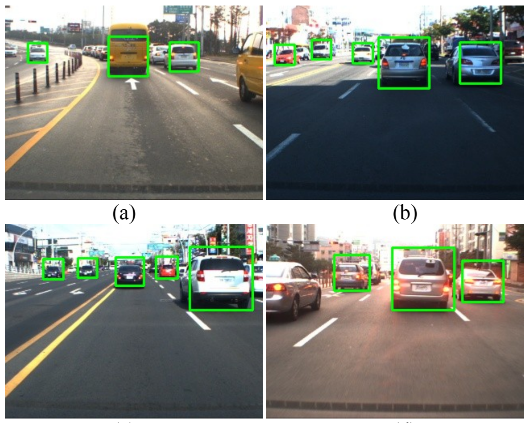
Fig. 12 Test conditions (a) dawn (b) morning (c) noon (d) afternoon

In rainy day, the shape and the intensity of shadow are shown differently compared to they are shown in sunny day because the appearances of vehicles are reflected on the wet road surface (Fig. 13). However, the proposed algorithm can detect vehicles in heavy rain, because our algorithm does not utilize shadow underneath vehicles in the detection. The proposed position correction method also works well on wet roads.

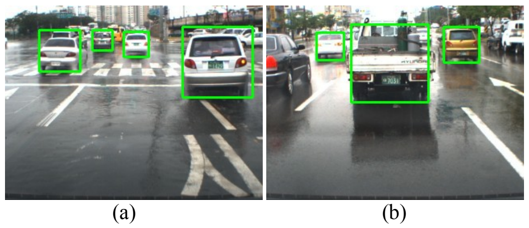
Fig. 13 Vehicle detection in rainy day

However, there are two major factors that degrade the detection rate of our algorithm. One factor is raindrops on the front windshield, causing input images to be locally distorted. The cause of this problem is that raindrops on the front windshield are not immediately wiped out as they fell on the front windshield. If a vehicle is completely distorted (Fig. 14a), our algorithm fails to detect the vehicle. In contrast, our algorithm can detect partially distorted vehicles (Fig. 14b).

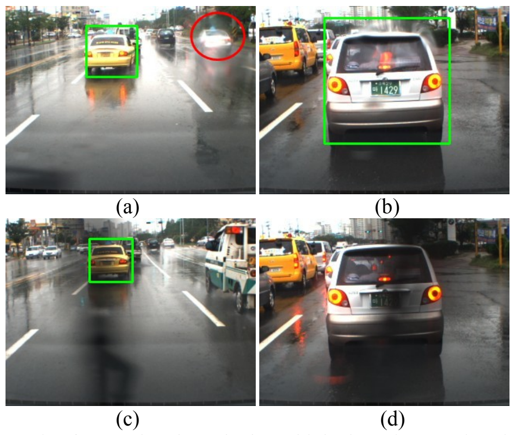
Fig. 14 Major factors that degrade the vehicle detection rate in rainy day (a, b) distortion caused by raindrops on the front windshield (c, d) occlusion by wipers

Another factor is occlusion by wipers. In case where after-image of wipers makes a vehicle shown darker, our algorithm could detect the vehicle (Fig. 14c). However wipers can interfere with the verification of a vehicle by horizontal edge, causing our algorithm to fail to detect the vehicle (Fig. 14d).

These problems are inevitable in rainy day. Therefore the vehicle detection rate of our algorithm is degraded in rainy day.

IV. Conclusion

We detected front vehicles in the images that were obtained from a camera mounted on the front windshield. First, we computed IPT matrix which describes the relation between image coordinate and the real-world coordinate. Second, we extracted vehicle candidates using AdaBoost. This process was speeded up using IPT matrix. The extracted candidates were verified by the existence of vertical and horizontal edges. Then the detected regions were clustered based on the distance and the size to represent one region per vehicle. The vehicle regions were then corrected based on the edge and shadow. The proposed algorithm took average time of 37.13ms per frame. Under normal lighting condition, the detection rate and false detection rate were 90.77% and 7.81%, respectively. This detection performance can be improved if more images are added to the training image set. The proposed algorithm can also detect vehicles in rainy day. However, wipers and raindrops on front windshield of the experimental vehicle sometimes occluded or distorted vehicle images, causing degradation of the vehicle detection rate.

References

[1] Z. Sun, G. Bebis, and R.Miller“On-Road Vehicle Detection: A Review,” IEEE Trans. PAMI, vol. 28, no. 5, pp. 694-711, 2006.

[2] H. Mori and N. Charai, “Shadow and Rhythm as Sign Patterns of Obstacle Detection,” *Proc. Int'l Symp., Industrial Electronics*, pp. 271-277, 1993.

[3] C. Tzomakas and W. Seelen, “Vehicle Detection in Traffic Scenes Using Shadows,” Technical Report 98-96, Institut fu¨ r Neuroinformatik, Ruht-Universitat, Bochum, Germany, 1998.

[4] N. Matthews, P. An, D. Charnley and C. Harris, “Vehicle Detection and Recognition in grayscale Imagery,” *Control Eng. Practice*, vol. 4, pp. 473- 479, 1996.

[5] N. Srinivasa, “A Vision-Based Vehicle Detection and Tracking Method for Forward Collision Warning,” *Proc. IEEE Intelligent Vehicle Symposium,* pp. 626-631, 2002.

[6] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *IJCV* 2004.

[7] A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau, "Vehicle detection combining gradient analysis and AdaBoost classification," in. Intelligent Transportation Systems, 2005.

[8] G. Y. Song, K. Y. Lee, and J. W. Lee, “Vehicle detection by edge-based candidate generation and appearance-based classification,” Intelligent Vehicles Symposium, 2008 IEEE, pages 428–433, June 2008.

[9] Z. Sun, G. Bebis, and R. Miller, “On-Road Vehicle Detection Using Gabor Filters and Support Vector Machines,” *Proc. IEEE Int'l Conf Digital Signal Processing*, July 2002.

[10] F. Han, Y. Shan, R. Cekander, H. S. Sawhney, and R. Kumar, “A Two-Stage Approach to People and Vehicle Detection with HOG-Based SVM,” *In: PerMIS proceeding*, pp. 133-140, 2006.

[11] R. Lienhart and J. Maydt, “An Extended Set of Haar-like Features for Rapid Object Detection.” *IEEE ICIP*, pp. 900-903, 2002.