International Journal of Automotive Technology, Vol. 13, No. 4, pp. 653−669 (2012)
DOI 10.1007/s12239−012−0064−x

pISSN 1229−9138/ eISSN 1976-3832

VISION-BASED FUSION OF ROBUST LANE TRACKING AND FORWARD VEHICLE DETECTION IN A REAL DRIVING ENVIRONMENT

H.-C. CHOI[1)], J.-M. PARK[2)], W.-S. CHOI[3)] and S.-Y. OH[4)*]

[1)]Multimedia Research Team, AT R&D Center, Daum Space. 1, Jeju 690-140, Korea
[2)]Advanced Electronics 2, R&D Division, Mando Corporation, 188 Seongnam-si, Gyeonggi 463-870, Korea
[3,4)]Department of Electrical Engineering, Pohang University of Science and Technology, Gyeongbuk 790-784, Korea

(Received 30 March 2010; Revised 11 June 2011; Accepted 9 January 2012)

ABSTRACT−With the goal of developing an accurate and fast lane tracking system for the purpose of driver assistance, this paper proposes a vision-based fusion technique for lane tracking and forward vehicle detection to handle challenging conditions, i.e., lane occlusion by a forward vehicle, lane change, varying illumination, road traffic signs, and pitch motion, all of which often occur in real driving environments. First, our algorithm uses random sample consensus (RANSAC) and Kalman filtering to calculate the lane equation from the lane candidates found by template matching. Simple template matching and a combination of RANSAC and Kalman filtering makes calculating the lane equation as a hyperbola pair very quick and robust against varying illumination and discontinuities in the lane. Second, our algorithm uses a state transfer technique to maintain lane tracking continuously in spite of the lane changing situation. This reduces the computational time when dealing with the lane change because lane detection, which takes much more time than lane tracking, is not necessary with this algorithm. Third, false lane candidates from occlusions by frontal vehicles are eliminated using accurate regions of the forward vehicles from our improved forward vehicle detector. Fourth, our proposed method achieved robustness against road traffic signs and pitch motion using the adaptive region of interest and a constraint on the position of the vanishing point. Our algorithm was tested with image sequences from a real driving situation and demonstrated its robustness.

KEY WORDS : Lane detection, Lane tracking, Kalman filtering, Vehicle detection, Lane change, Occlusion handling

1. INTRODUCTION

One of the essential elements of a driver assistant system (DAS) is the detection of forward lanes and vehicles. As image processing techniques have been improved, many vision-based detection techniques for forward lanes and vehicles have been developed to substitute hardware range sensor-based method because hardware range sensor like laser sensor is expensive and inconvenient to be equipped. Previously, avery fast lane detection method on mobile system by using simple edge detection and Hough transform (Hsueh *et al*., 2009) and an hyperbola lane pair detection method in PC-based system (Chen and Wang, 2006; Borkar *et al.*, 2009) ere proposed. However, their method cannot deal with the discontinuity in the lane and pitch motion, which often occur in real driving situations; however, their method is fast because of its simplicity.

Other PC-based methods that can deal with noisy situations (Tsai *et al*., 2008; Wu *et al*., 2009) used a complicated random walk or a Kalman filter to find lane equation as a B-spline curve and achieved robustness against lane discontinuity and road traffic signs. A fusion technique of lane candidate detection by steerable filter and forward depth information from laser sensor (Jung *et al*., 2009) minimized the area of searching region for lanes and also dealt with occlusion by forward obstacles. For dealing with pitch motion of ego-vehicle, motion vector estimation techniques by using optical flow (Chang *et al*., 2002), motion inpainting (Matsushita *et al*., 2006), particle filtering (Yang *et al*., 2009), and adaptive RANSAC (Choi *et al*., 2009) have been proposed. However, the high processing time of those methods, from ten to several hundreds of millisecond on high performance PC system, and expensive range sensor are not suitable for DSP based stand-alone system which is the expected commercial form of DAS.

The most time-consuming part of a lane tracking system is the initial lane detection in which candidate searching over a large area of an image is necessary. Therefore, it takes a lot of time to execute a lane detection step every time the tracked lane is lost. To reduce the time consumed performing this repetitive detection, a novel method for maintaining tracking in a missing lane situation, such as a lane discontinuity, an abrupt illumination change, a lane change, or an occlusion by a forward vehicle, is necessary.

We propose a fusion system of fast lane tracking and vehicle detection in a single camera system that is robust against lane changes, abrupt illumination changes, road

**Corresponding author.* e-mail: syoh@postech.ac.kr

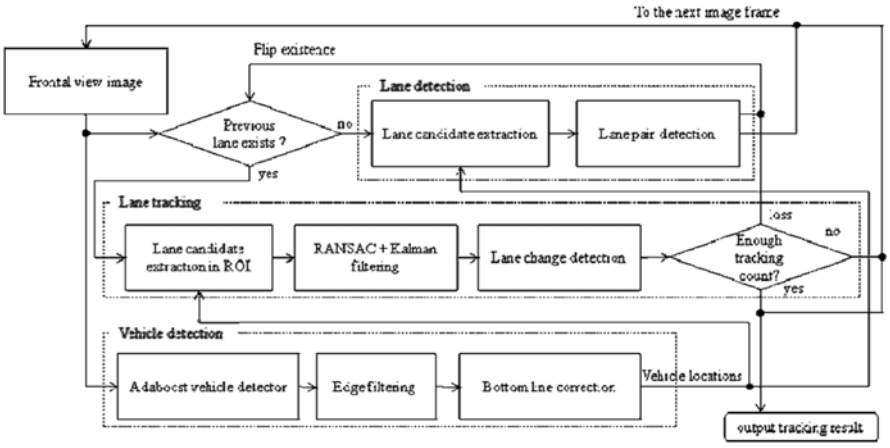


Figure 1. Proposed lane tracking system.

traffic signs, car occlusions, and pitch motion. Our system consists of simple lane detection using template matching, lane tracking using random sample consensus (RANSAC) (Fishchler and Bolles, 1981) and a Kalman filter (Welch and Bishop, 2006) and using Adaboost-based (Viola and Jones, 2001) vehicle detection. Figure 1 shows the whole flow chart of our proposed system. Once a lane pair is detected using simple template matching, the detected lane pair is tracked through the subsequent images using RANSAC and Kalman filtering, which determine the robustness against abrupt illumination changes and lane discontinuities. Robustness against road traffic signs and pitch motion is achieved with a small adaptive region of interest (ROI) for the lane equation and a vanishing point constraint. Additionally, the state transferring technique is used to maintain tracking of the lane when the ego-vehicle is changing lanes. The vehicle detection portion detects both the rough regions of forward vehicles using the Adaboost detector as well as the correct rectangular region using our own post-processing techniques, which consist of edge filtering and bottom line correction. This vehicle region information is used to deal with occlusions in the lane detection and tracking.

The contributions from this paper are the following.

- Very low computational complexity due to a very simple template matching using integral images and a rule-based lane pair detection, and the overall computational complexity is reduced to several milliseconds when using a PC.

- Robustness against illumination change, lane discontinuity, pitch motion, and road traffic signs. Local template matching for lane candidate extraction results in a number of lane candidates under various lighting conditions. Noisy candidates and temporal missing candidates from lane discontinuities are dealt with using RANSAC and Kalman filtering. Additionally, an adaptive ROI technique and a vanishing point constraint prevent failures in tracking as a result of pitch motion and road traffic signs.

- Maintaining tracking during a lane changing situation. Our proposed state transferring technique makes continuous lane tracking during lane changing situations possible without re-initialization using relatively time-consuming lane detection.

- Accurate region detection for forward vehicles. Edge filtering and bottom line correction followed by Adaboost vehicle detection accurately determine the region of forward vehicles.

- Dealing with occlusion by forward vehicles. Lane candidates in the forward vehicle regions are eliminated in the lane detection and tracking. Therefore, the effect of noisy candidates from occlusion is reduced. The remaining portions of this paper consist of the following. Section 2 describes the details about our lane detection and tracking algorithm. Section 3 describes the vehicle detection and its application to lane tracking for dealing with occlusions from forward vehicles. The performance tests of our algorithm are presented in section 4. The conclusions of this work and future work for further improvement are described in section 5.

2. FAST LANE DETECTION AND TRACKING

2.1. Extraction of Lane Candidates

Because the difference between the lanes and the road intensities is obvious in the green channel with various illumination conditions in a real driving situation, the green channel input image is used for the lane candidate extraction. Candidate extraction is performed by convolution of the input intensity and the templates, as depicted on the right side of Figure 1. The blue dotted lines on the left side of the image in Figure 2 represent the ROI for searching for lane candidates. The ROI is set as the forward area from 3 m to 30 m in which the lane is not occluded by an ego-vehicle and not too narrow to be detected.

Because this convolution needs N additive steps when

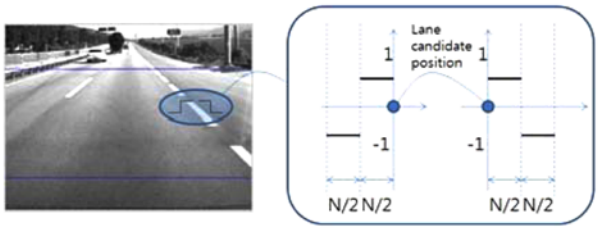


Figure 2. Input image and templates for the lane candidate extraction.

the length of the template is N, the computational cost can be reduced by several additions using an integral image (Viola and Jones, 2004). If the coordinate of the candidate position is (u, v) and the integral image is I, then the left and right convolution, CV_L and CV_R, can be calculated from (1) and (2).

$$CV_L = \left(I(u,v) + 2\times I\left(u-\frac{N}{2}, v-1\right) + I(u-N,v) - I(u,v-1) - 2\times I\left(u-\frac{N}{2}, v\right) - I(u-N,v-1)\right) \Big/ \left(\frac{N}{2}\right), \quad (1)$$

$$CV_R = \left(I(u+N,v-1) + 2\times I\left(u+\frac{N}{2}, v\right) + I(u,v-1) - I(u+N,v) - 2\times I\left(u+\frac{N}{2}, v-1\right) - I(u,v)\right) \Big/ \left(\frac{N}{2}\right), \quad (2)$$

These two convolutions calculate the average intensity difference between the road and the lane. Therefore, many lane candidates can be obtained along a roughly horizontal scan line by selecting the positions that have convolution values greater than a threshold. Among these roughly selected lane candidates, a representative candidate that has the maximum convolution values is selected. Figure 3(a) shows the result of candidate extraction. The extracted candidates are presented as green dots.

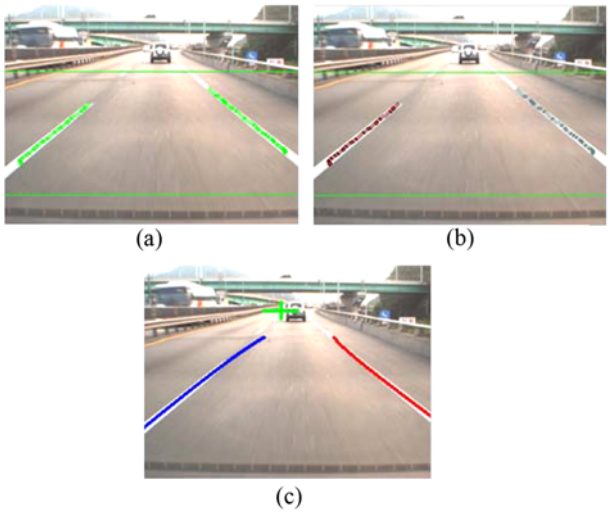


Figure 3. Results of lane candidate extraction and clustering.

2.2. Initial Lane Pair Detection

In the next step, the extracted representative candidates are clustered based on the distance in the image coordinates. If the distance of two candidates is less than 6 pixels, then they are selected to be in the same cluster. Here, the clusters that only have a small number of lane candidates (less than 5) are assumed to be a noise cluster and are excluded. Figure 3(b) shows the result of clustering. There are two clusters in Figure 3(b): the dark red cluster for the left lane and the light blue for the right lane.

After clustering the lane candidates, the lane equations for each cluster are calculated in the world coordinate system. The equations are all linear, as shown in Figure 4a. To calculate the linear equations for each cluster, first, the candidate points on the image coordinate (u, v) are transformed into the corresponding world coordinates (X, Z) using an inverse perspective transform matrix (IPT) (Haralick, 1989) T^{-1} from (3)~(5).

$$(X,Z) = \left(\frac{X'}{s}, \frac{Z'}{s}\right), \quad (3)$$

$$T^{-1}\begin{bmatrix} u & v & 1 \end{bmatrix}^T = \begin{bmatrix} X' & Z' & s \end{bmatrix}^T, \quad (4)$$

$$T^{-1} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & 1 \end{bmatrix}. \quad (5)$$

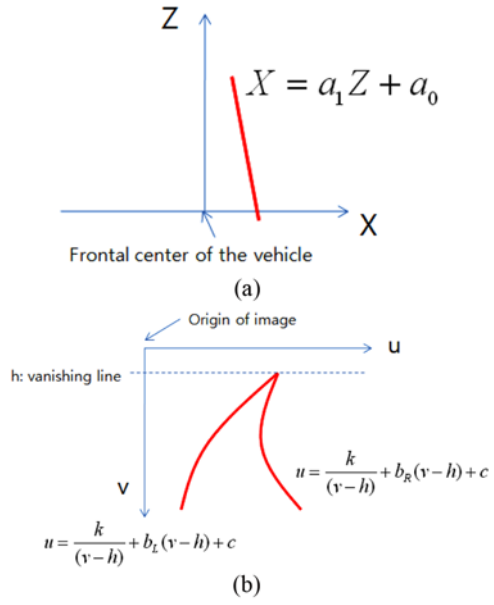


Figure 4. Lane equations in the world coordinates (a) and the image coordinates (b).

The set of candidates in the world coordinate (X_i, Z_i) is used to calculate the coefficients (a_1, a_0) of a linear equation using the least squares error method, as observed (6).

$$\begin{bmatrix} Z_1 & 1 \\ Z_2 & 1 \\ \vdots & \vdots \\ Z_n & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$
$$\rightarrow \boldsymbol{Z} = \begin{bmatrix} Z_1 Z_2 \cdots Z_n \\ 1 \quad 1 \quad \cdots \quad 1 \end{bmatrix}, \tag{6}$$
$$\begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = (\boldsymbol{Z}^T \boldsymbol{Z})^{-1} \boldsymbol{Z} \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix}$$

Among the linear equations in world coordinates, one pair of lanes that satisfies the following conditions is selected as the final lane pair.

Condition #1) Each lane of the lane pair has more than 10 candidate points in the forward range of [3m, 10m]

Condition #2) The distance between two lanes is greater than 2.5 m and less than 3.75 m at the forward position of 5 m, where the range of the lane width is based on the Korean standard lane width of 2.75 m (common road) ~ 3.6 m (expressway).

Condition #3) The angle between the two lanes is less than 2 degrees

If there are several lane pairs that satisfy these three conditions, then the lane pair that has the minimum distance between the two lanes is selected as the lane pair. From the two selected lanes, the lane that has a smaller X coordinate corresponding to 5 m in front of the vehicle is the left lane and the other lane is the right lane.

The last step in the initial lane detection is transforming the lane equations from world coordinates to image coordinates. This step is necessary because the lane tracking will be performed by the lane equations in the image coordinates. The equations themselves cannot be transformed because transferring between the world and the image coordinates is not linear. Therefore, the points on the lane equations are sampled and projected from the world coordinates to the image coordinates. Then, the hyperbola lane equation pair in image coordinates is calculated using the projected points. The hyperbola lane equation pair in the image coordinates is shown in Figure 4b, where (u, v) is the coordinate of the image point, k is the curvature, b_L and b_R are the left and the right tangent of the asymptotes, h is the height of the vanishing line, and c is the horizontal center of the input image. First, the height of the vanishing point h is calculated as the v coordinate corresponding to an infinite Z from (7) using the IPT described in (5).

$$h = \lim_{Z \to \infty} \frac{t_{21}X + t_{22}Z + t_{23}}{t_{31}X + t_{32}Z + 1} = \frac{t_{22}}{t_{32}}. \tag{7}$$

Next, the other parameters from the hyperbola equation are calculated using the left and right lane sample points, (u_{Li}, v_{Li}) and (u_{Ri}, v_{Ri}), in image coordinates using the least squares method given in (8). Figure 3c shows the calculated lane equation in image coordinates. The blue curve and the red curve represent the left and the right lane for each. The green cross represents the vanishing point (c, h).

$$\mathbf{A}\begin{bmatrix} k & b_L & b_R & c \end{bmatrix} = \mathbf{B}$$
$$\rightarrow \begin{bmatrix} k & b_L & b_R & c \end{bmatrix} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{B}, \tag{8}$$

$$\mathbf{A} = \begin{bmatrix} 1/(v_{L1}-h) & v_{L1}-h & 0 & 1 \\ 1/(v_{L2}-h) & v_{L2}-h & 0 & 1 \\ & \vdots & & \\ 1/(v_{Ln}-h) & v_{Ln}-h & 0 & 1 \\ 1/(v_{R1}-h) & 0 & v_{R1}-h & 1 \\ 1/(v_{R2}-h) & 0 & v_{R2}-h & 1 \\ & \vdots & & \\ 1/(v_{Rm}-h) & 0 & v_{Rm}-h & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} u_{L1} \\ u_{L2} \\ \vdots \\ u_{Ln} \\ u_{R1} \\ u_{R2} \\ \vdots \\ u_{Rm} \end{bmatrix}. \tag{9}$$

2.3. Adaptive ROI for Lane Candidate Extraction during the Tracking Phase

Once a lane pair is detected, a small region of interest (ROI) around the previously detected lane pair is used for fast and robust lane tracking against road traffic signs. The ROI in image coordinates for (u = 0, v) is calculated in (12) using equations (3), (4), (10) and (11). Here, the region from -0.6 m to +0.6 m centered on the previous lane equation is used as the default ROI. For the coordinate transformation from world coordinates to image coordinates, the perspective transform (PT) matrix T is used.

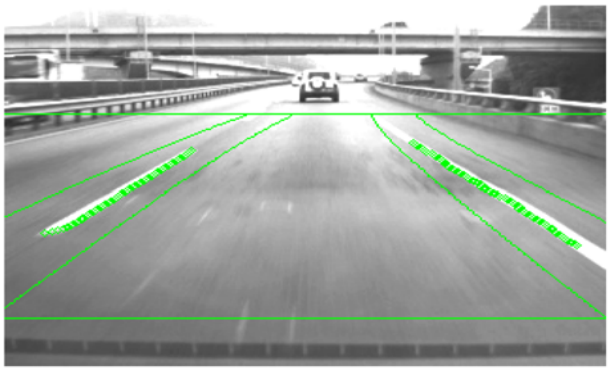
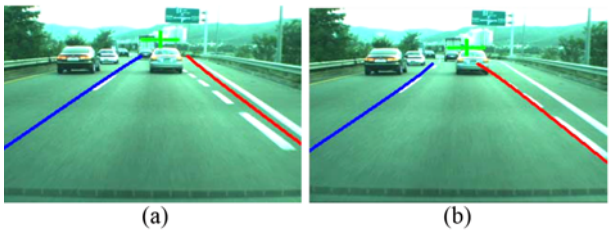


Figure 5. Example of the lane candidate extraction in the ROI.



(a) (b)

Figure 6. Effect of the representative lane candidate selection. In the case of double right lanes, there can be two representative lane candidates, and the incorrect right lane equation is calculated (a). However, by selecting the nearest candidate, the right lane equation is calculated correctly (b).

$$T\begin{bmatrix} X-0.6 & X+0.6 \\ Z & Z \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} u1' & u2' \\ v1' & v2' \\ s1 & s2 \end{bmatrix}, \tag{10}$$

$$\begin{bmatrix} u1 & u2 \\ v1 & v2 \end{bmatrix} = \begin{bmatrix} \frac{u1'}{s1} & \frac{u2'}{s2} \\ \frac{v1'}{s1} & \frac{v2'}{s2} \end{bmatrix}, \tag{11}$$

$$ROI(v) = [u1, u2]. \tag{12}$$

We used an adaptive ROI width to consider the uncertainty of the previous lane equation. If the uncertainty of the lane equation is large, then the lane candidates should be searched for in a large ROI because the large uncertainty of the lane equation means that the lane equation does not represent the actual lane equation well. Conversely, if the uncertainty of the lane equation is small, then the lane candidates are assumed to exist near the previous lane equation. The uncertainty of the lane equation is proportional to the variances of b_L and b_R. Using this uncertainty, ROI_L and ROI_R, the widths of the left and the right ROI, can be adaptively calculated, as given in (13) and (14). ROI_o is the default width of the ROI corresponding to the variance 0.05. p_{22} and p_{33} are the elements of the state covariance that correspond to b_L and b_R, respectively.

$$ROI_L = ROI_o \times \frac{\sqrt{p_{22}}}{0.05}, \tag{13}$$

$$ROI_R = ROI_o \times \frac{\sqrt{p_{33}}}{0.05}. \tag{14}$$

Figure 5 shows the result of the candidate extraction from the ROI. The green lines represent the ROI regions for the left and the right lanes, and the green dots represent the detected lane candidates. The lane candidate was extracted using the integral input image and the lane candidate templates, as explained in section 2.1.

In a different process from the initial lane detection process, the representative lane candidate in the tracking phase is selected as the closest to the ego-vehicle in the lane tracking process. In the case of Figure 6(a), if one lane is separated in two different lanes, the results are an incorrect lane equation. However, with the selection of the left-most or the right-most candidate, the correct lane tracking, as shown in Figure 6(b), is possible in the case of double lanes.

2.4. Tracking Lane Equation with Noisy Lane Candidates Using RANSAC and a Kalman Filter

Lane candidates often exist outside the ROI in situations where pitch motions are caused by an unstable road or speed bumps, obscure lane painting, or an illumination change. The RANSAC algorithm (Fishchler and Bolles, 1981) is used to deal with noisy lane candidates because of its good and fast performance when selecting inliers. Additionally, an extended Kalman filter (EKF) (Welch and Bishop, 2006) is used to track the non-linear lane equation when there are noisy candidates and unstable shaking in the real driving environment. Because the velocity of the vehicle and other controls from outside the vehicle are not used here, no control parameters or dynamic models are needed for the EKF. Therefore, the equations of Kalman prediction and correction are simplified as (15)~(18).

$$\hat{x}_k^- = \hat{x}_{k-1}, \tag{15}$$

$$P_x^- = P_{k-1} + Q_{k-1}, \tag{16}$$

$$\hat{x}_k = \hat{x}_k^- + K(z_k - h(\hat{x}_k^-)), \tag{17}$$

$$P_k = (I - KH)P_k^-, \tag{18}$$

where k is the time step, $\hat{x}_k$ is the Kalman state, $\hat{x}_k^-$ is the predicted state, P_k^- is the predicted covariance of the state, P_k is the covariance of the state, Q_k is the covariance of the state noise, z_k is the measurement, K is the Kalman gain, $h(x)$ is the function that maps the state to the measurement space, and H is the measurement matrix. For lane equation tracking using a Kalman filter, the initial parameters, such as the state $\hat{x}_0$, the covariance of state P_0, and the covariance of state noise Q_0, should be defined first. The initial state $\hat{x}_0$ can be defined as a vector that consists of the initially detected parameters of the hyperbola lane equation pair, (k, b_L, b_R, c, h) as calculated in (14).

$$\hat{x}_0 = [k\ b_L\ b_R\ c\ h]^T. \tag{19}$$

The covariance of the state and the covariance of the state noise have non-zero values only for the diagonal elements because each element of the state is independent except for b_L and b_R. b_L and b_R, which are not independent

and have some covariance value because the tangents of the two lanes in the hyperbola pair vary simultaneously. Therefore, the initial state covariance and the state noise covariance are defined, as shown (20). The values are experimentally determined. Once the initial parameters of the Kalman filter are defined, the Kalman prediction steps, as defined in (15) and (16), are executed.

$$P = Q = \begin{bmatrix} 10^2 & 0 & 0 & 0 & 0 \\ 0 & 0.05^2 & 0.9\times0.05^2 & 0 & 0 \\ 0 & 0.9\times0.05^2 & 0.05^2 & 0 & 0 \\ 0 & 0 & 0 & 1.0^2 & 0 \\ 0 & 0 & 0 & 0 & 0.1^2 \end{bmatrix}. \tag{20}$$

After extracting the lane candidates from the adaptive ROI, the outliers are removed using the RANSAC algorithm. Here, the threshold for the inliers is set to 0.2 m. The following descriptions outline the RANSAC process for the selection of inliers.

Step1) select 3 different points from the left lane candidates.

Step2) calculate the left lane equation (a, b_L, c) with the points selected using the least squares error method.

Step3) select, from the lane candidate, the inliers in which the distance from the lane equation is less than 0.2 m.

Step4) store the index of the inliers and repeat steps 1-3 until a maximum repeat number is reached or the number of inliers is equal to the number of left lane candidates.

Step5) repeat steps 1-4 for the right lane

Step6) if the number of inliers for the left or the right lane is less than 10, the inliers are abandoned.

The Kalman update, which is the last step of lane tracking, is executed with the selected inliers. If the inliers do not exist, the predicted state is used as the new lane equation without the Kalman update. If the inliers are valid, then their u coordinates are used as the measurement z_k for the Kalman filter, as defined in (21). Here, the distance l between the two lanes in world coordinates is 2.5 m, which is the usual lane width for a Korean highway.

$$z_k = \begin{bmatrix} u_{L1} \cdots u_{Ln} & u_{R1} \cdots u_{Rm} & l \end{bmatrix}^T. \tag{21}$$

The predicted measurement $h(\hat{x}_k^-)$ is calculated as a vector of u values from the predicted lane equation $(\hat{x}_k^-)$ corresponding to v of each lane candidate, as defined in (22)~(24).

$$h(\hat{x}_k^-) = \begin{bmatrix} u_L(v_{L1}|\hat{x}_k^-) & \cdots & u_L(v_{Ln}|\hat{x}_k^-) \\ u_R(v_{R1}|\hat{x}_k^-) & \cdots & u_R(v_{Rm}|\hat{x}_k^-) & b_R - b_L \end{bmatrix}^T. \tag{22}$$

$$u_L(v|x) = \frac{k}{v-h} + b_L(v-h) + c, \tag{23}$$

$$u_R(v|x) = \frac{k}{v-h} + b_R(v-h) + c. \tag{24}$$

$$H = \begin{bmatrix} \frac{\partial u_L(v_{L1})}{\partial x} \\ \vdots \\ \frac{\partial u_L(v_{Ln})}{\partial x} \\ \frac{\partial u_R(v_{R1})}{\partial x} \\ \vdots \\ \frac{\partial u_R(v_{Rm})}{\partial x} \\ \frac{\partial l}{\partial x} \end{bmatrix} = \begin{bmatrix} 1/(v_{L1}-h) & v_{L1}-h & 0 & 1 & k/(v_{L1}-h)^2 - b_L \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1/(v_{Ln}-h) & v_{Ln}-h & 0 & 1 & k/(v_{Ln}-h)^2 - b_L \\ 1/(v_{R1}-h) & 0 & v_{R1}-h & 1 & k/(v_{R1}-h)^2 - b_R \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1/(v_{Rm}-h) & 0 & v_{Rm}-h & 1 & k/(v_{Rm}-h)^2 - b_R \\ 0 & -1 & 1 & 0 & 0 \end{bmatrix}. \tag{25}$$

The measurement matrix H is a first order derivative of the predicted measurement with respect to the state, as defined in (25).

The measurement noise matrix R is the diagonal matrix in which the diagonal elements consists of the variances of the elements in the measurement vector. The noise variance is set as 5 pixels for each measurement element, for example, 0.1 m for the width of lane l. Therefore, the measurement noise matrix is shown in (26).

$$R_{(n+m+1)\times(n+m+1)} = \begin{bmatrix} 5^2 & & & \\ & 5^2 \ddots & 0 & \\ & 0 & 5^2 & \\ & & & 0.1^2 \end{bmatrix}. \tag{26}$$

Calculating the Kalman gain K requires the operation of matrix inversion with a $(n + m + 1)$ by $(n + m + 1)$ matrix, as shown in (27)

$$K = PH^T(HPH^T + R)^{-1}. \tag{27}$$

As the number of lane candidates increases, the amount of time consumed performing matrix inversion increases enormously. To prevent this, (27) is modified into (29) using the matrix inversion lemma (Haykin, 2001) (28) to reduce the computational complexity. In equation (29), the inversion of the measurement noise matrix R can be calculated simply by calculating the inverses of its diagonal elements in advance and the inversions of state covariance matrix P, and the matrix inside the parenthesis is also simple because its size is 6 by 6.

matrix inversion lemma:

$$A(I+BA)^{-1} = (I+AB)^{-1}A. \tag{28}$$

$$K = (H^T R^{-1} H + P^{-1})^{-1} H^T R^{-1} . \quad (29)$$

With these Kalman gains, measurements, and the predicted measurements, a corrected state of the lane equation and a state covariance matrix are calculated using (17) and (18).

For the stability against pitch motion, instead of complex image stabilization techniques (Chang *et al*., 2002; Matsushita *et al*., 2006; Yang *et al*., 2009; Choi *et al*., 2009), the height of the vanishing line of the hyperbola lane equation, h, is restricted to be in the range from 30 pixel index to 80 pixel index in image coordinates, as shown in (30).

$$h = \max(30, \min(80, h)) . \quad (30)$$

For the effective output of the lane equation, we assume that the tracked lane equation is valid if lane tracking is successful for 10 continuous frames. Successful tracking means that the number of lane candidates after RANSAC is greater than 10 for either the left or the right lane. Once the output is valid, the output remains valid until tracking fails for 10 continuous frames. If the output becomes invalid, then the initial detection described in sections 2.1 and 2.2 is executed again to find the new lane equation.

2.5. State Transferring for dealing with Lane Change

When the vehicle changes lanes to the left, the equation for the left lane becomes that of the right lane, and a new equation for the left lane should be recalculated. Instead of repeating the detection process as explained in sections 2.1 and 2.2, the state of the Kalman filter is transferred from the left to the right and a new estimate for the left state is made with the lane width. This scheme can reduce the number of detections during the lane change situation and can reduce the processing time. When the vehicle is approaching the left lane, the tangent of the left lane b_L changes from a negative value to zero. After the vehicle crosses the left lane, b_L becomes positive. Therefore, we assume that the lane change to the left occurs when b_L is greater than a positive margin of 0.2 to detect the moment of the lane change. State transferring is simply copying b_L to b_R when the lane change is to the left. Because the other parameters of the lane equation, (k, h, c), are the same for both the left and right lanes, and because p_{22}, the variance of b_L, is also copied to p_{33}, the variance of b_R. p_{23} and p_{32}, and the covariance between b_L and b_R is not changed because the relationship between them does not change much during the lane change situation. b_L of the new left lane equation can be estimated as b_L–l. Additionally, p_{22} is not changed because we assume that the uncertainty of the new left lane equation is the same as that of the right lane equation. In the case of a lane change to the right, the same process in the opposite direction is applied. Equations (31) and (32) describe the state transfer process for the lane change to the right and to the left, respectively.

$$\begin{aligned} &\textit{if } b_R < -0.2 \textit{ then} \\ &\quad b_L = b_R, p_{22} = p_{33}, \textit{and } b_R = b_R + l, \end{aligned} \quad (31)$$

$$\begin{aligned} &\textit{if } b_L > 0.2 \textit{ then} \\ &\quad b_R = b_R, p_{33} = p_{22}, \textit{and } b_L = b_L - l. \end{aligned} \quad (32)$$

Table 1. Algorithm for computing the range of the sub-window size.

- **for** (u, v)

Compute the real world coordinate (x_L, y) of (u, v)

$$\begin{bmatrix} z \cdot x_L \\ z \cdot y \\ z \end{bmatrix} = T^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

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:

$$\begin{bmatrix} w \cdot u_{\min} \\ w \cdot v_{\min} \\ w \end{bmatrix} = T \begin{bmatrix} x_{R,\min} \\ y \\ 1 \end{bmatrix}, \begin{bmatrix} w' \cdot u_{\max} \\ w' \cdot v_{\max} \\ w' \end{bmatrix} = T \begin{bmatrix} x_{R,\max} \\ y \\ 1 \end{bmatrix}$$

Compute the minimum and maximum window width: $W_{\min}(u, v) = u_{\min} - u$, $W_{\max}(u, v) = u_{\max} - u$.

3. DEALING WITH OCCLUSION USING FORWARD VEHICLE DETECTION

3.1. Fast Forward Vehicle Candidate Extraction using Adaboost and the ROI

The AdaBoost vehicle detector and 15 Haar-like features, which include the original features (Viola and Jones, 2004) and the extended features (Lienhart and Maydt, 2002) as shown in Figure 7, are used for the vehicle candidate extraction.

The ROI of the forward vehicle extraction is set to 6 m to 50 m because vehicles beyond 50 m are too small to be

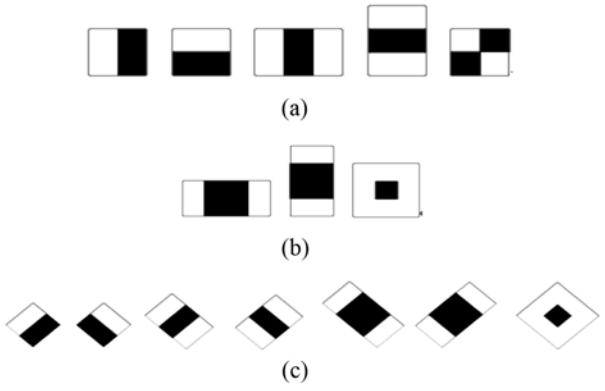


Figure 7. Haar-like features used to train the AdaBoost: (a) original Haar-like features; (b) upright extended features, and (c) 45° rotated features.

Table 2. Modified scanning scheme.

for ($scale = S_{\min};\ scale \le S_{\max};\ scale\times = S_{step}$)

$v_{\text{step}} \leftarrow S_{step}\times v_{step}$

$u_{\text{step}} \leftarrow S_{step}\times u_{step}$

$W \leftarrow S_{step}\times W$

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

detected, and vehicles closer than 6 m are both seriously affected by the camera lens distortion and are not completely shown in the input images. The image coordinates v_{6m} and v_{50m} corresponding to 6 m and 50 m are calculated using perspective transform (PT) matrix T, as defined in (33) and (34).

$$[v_{6m}, v_{50m}] = \left[\frac{v'_{6m}}{s_{6m}}, \frac{v'_{50m}}{s_{50m}}\right], \tag{33}$$

$$\begin{bmatrix} u'_{6m} & u'_{50m} \\ v'_{6m} & v'_{50m} \\ s_{6m} & s_{50m} \end{bmatrix} = T \begin{bmatrix} 0 & 0 \\ 6 & 50 \\ 1 & 1 \end{bmatrix}. \tag{34}$$

Then, the input image is scanned using a sub-window whose bottom position is between v_{6m} and v_{50m} to locate the vehicle candidates using AdaBoost. To speed up this process, we exploit the following facts: the width of vehicles of our interest range from 1.5 m to 2.7 m; vehicles at further distances are smaller and located at higher positions in the images compared with the vehicles that are closer to the vehicle. The range of the expected vehicle width [$W_{\min}$, $W_{\max}$] at each image coordinate (u, v) can be calculated using the IPT matrix as described in table 1.

Once the range of the vehicle width is computed, the width does not need to be computed again unless the camera position changes because the widths only depend on the IPT matrix. Then, our modified scanning scheme only classifies the sub-windows whose left-bottom position is (u, v) and whose width W is between $W_{\min}$ (u, v) and $W_{\max}$ (u, v) (Table 2). The sizes of the vehicles vary widely in the images depending on the distance from the scanning vehicle. Therefore, we set the parameters for the 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 is 24 by 24. This process both filters the extracted vehicle candidates based on their width and reduces the scanning time.

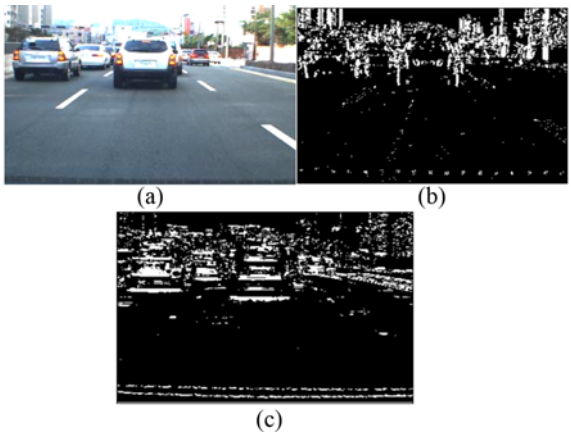


Figure 8. Detected vertical and horizontal edges; (a) original image; (b) vertical edges and (c) horizontal edges.

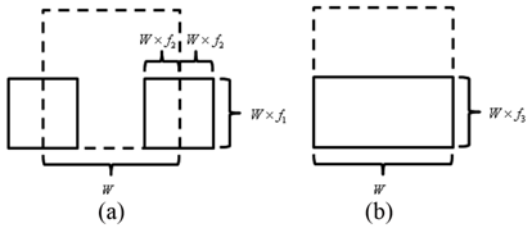


Figure 9. Region in which edges were searched for to verify the vehicle candidates dotted box: a detected region; solid box; regions in which edges were searched.

3.2. Vehicle Candidate Verification Using Edge Filtering

Although the extracted vehicle candidates are filtered based on their width, it is still probable that non-vehicles will be detected as vehicles. To reduce the number of detected false positives, the extracted vehicle candidates are verified based on the facts that vehicles have strong vertical edges on their left and right sides, and vehicles have strong horizontal edges at the bottom. The edges are detected using a 3 by 3 Sobel mask as:

$$G_x(u,\ v) = I(u,\ v)*\ S_x, \tag{35}$$

$$G_y(u,\ v) = I(u,\ v)*\ S_y, \tag{36}$$

Where S_x and S_y define the Sobel mask, I represents the grayscale input image and * represents the convolution operation. The condition for edges to be vertical is (37), and the condition for edges to be horizontal is (38).

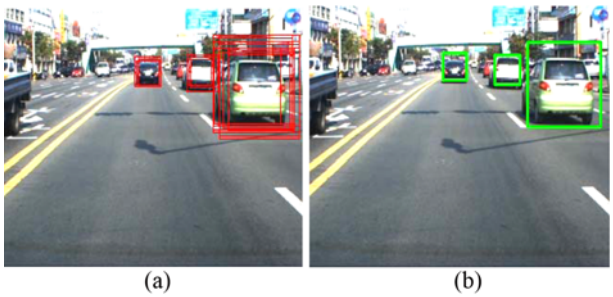


Figure 10. Detected vehicles: (a) detected region and (b) average of the detected regions.

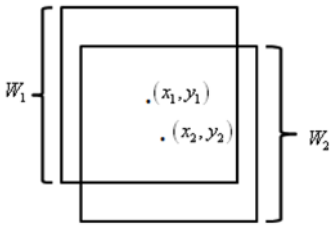


Figure 11. Parameters that are used in the clustering.

$$\left|\frac{G_y}{G_x}\right| < \frac{1}{3}, \tag{37}$$

$$\left|\frac{G_y}{G_x}\right| > 3. \tag{38}$$

The orientations of the edge pixels are calculated as the absolute value of the slope because this method is much faster than the arc tangent function. The edge images of Figure 8 show that vehicles have strong vertical edges on both sides (Figure 8(b)) and have strong horizontal edges on many parts of the vehicle: the bumper, the rear windshield, and the shadow (Figure 8(c)). Therefore, the existence of such edges is an important feature of the vehicle appearance.

In urban areas, however, many buildings exist that have strong vertical and horizontal edges as well. Those edges

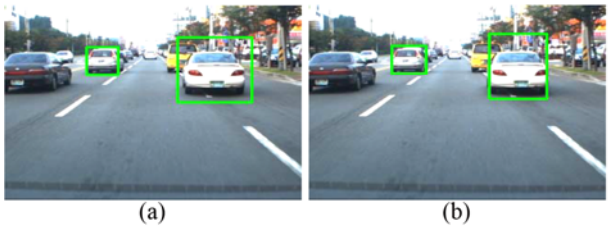


Figure 12. Position correction: (a) incorrect vehicle region and (b) corrected vehicle region.

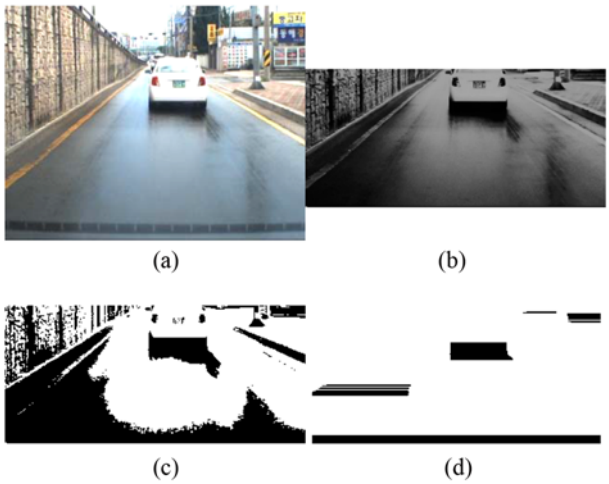


Figure 13. Shadow estimation: (a) input image; (b) histogram equalization; (c) binary image and (d) estimated shadow.

can cause false detections. To avoid this, the longest vertical edges are searched for in the lower regions of the detected regions (Figure 9(a)). The regions are centered at the left and right side of the detected regions, and the width and heights are proportional to the width of the detected window size. The longest horizontal edges are also searched for in the lower region of each vehicle candidate (Figure 9(b)). The size of the region is also proportional to the size of the vehicle candidates. $f_1 = 0.5$, $f_2 = 0.25$, and $f_3 = 0.5$ are used in this work.

Vehicles at a close distance may have discontinuous edge pixels because of their detailed texture and distortion caused by the camera. Therefore, we allow the longest edges to have a small number of discontinuous edge pixels. In our algorithm, the number of maximum discontinuous pixels is 5. The candidates that have edges longer than thresholds are selected as vehicles. The threshold for the vertical edges on the left and the right sides is determined to be $W \times T_{vertical}$. Likewise, the threshold length of horizontal edges is determined to be $W \times T_{horizontal}$. $T_{vertical} = 0.25$ and $T_{horizontal} = 0.5$ are used in this work.

3.3. Correct Vehicle Region Calculation by Edge and Shadow Filtering

Many regions around vehicles (Figure 10(a)) obtained as the result of AdaBoost detector with edge filtering should be clustered to represent one region for each vehicle (Figure 10(b)) when dealing with occlusion in the lane tracking process.

Let two detected regions have widths W_1 and W_2 and be centered at (w_1, y_1) and (x_2, y_2), respectively, as shown in Figure 11. Then, the criteria for the two detected regions to belong to the same vehicle are the distance, given in (39), and the size, given in (40).

Table 3. Shadow filtering algorithm.

- Histogram equalization
- Compute binary image B
- Estimate shadow

```
for v
  for u
    if B(u, v) = 0 and darkpixel = false
      start = u
      darkpixel = true
    else if B(u, v) ≠ 0 and darkpixel = true
      end = u-1
      width = end - start
      darkpixel = false
      if width >=W_min
        mark pixels between (start, v) and (end, v)
        as shadow
```

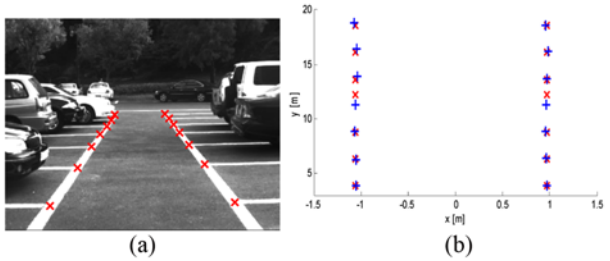
$$|x_1 - x_2| \le (W_1 + W_2) \times f_{overlap}, \quad (39)$$

$$|y_1 - y_2| \le (W_1 + W_2) \times f_{overlap}$$
$$W_{large} \times f_{size} \le W_{small} \le W_{large} \times (2 - f_{size}), \quad (40)$$

where W_{large} is the side length of the larger region, and W_{small} is the side length of the smaller region between W_1 and W_2. Here, the ratios of the overlap length $f_{overlap} = 0.5$ and the area $f_{size} = 0.5$.

Because the detected vehicles can have incorrect vehicle regions, as shown in Figure 12(a), the detected regions are corrected (Figure 12(b)) using the edges and the shadows of the detected vehicles. The left and the right positions of the detected vehicles are corrected to be the position at which the longest vertical edges are in the verification step. The bottom position of the detected vehicles can be corrected using the shadows below the vehicles. Although the shape and the intensity of shadows can vary depending on both the lighting and the surface conditions, the bottom position can be estimated from the shadow.

First, histogram equalization (Figure 13(b)) is performed to reduce the effects of the lighting condition. Next, binarization is executed to obtain the dark regions from the images, which include the vehicle's shadows (Figure 13(c)). Finally, the dark pixels are filtered out based on the expected width of vehicle in that position (Figure 13(d)).



(a) (b)

Figure 14. Camera Installation and calibration.

Table 4. Details of database for testing the lane detection and tracking.

DB	Time	Road condition	Lighting condition	#of images
#1	6:00	Highway including unnels curved lanes zebra crossings lane discontinuities lane changes	Twilight	19453
#2	9:00		Daylight	15050
#3	12:00		Daylight with specular light	9483
#4	16:00		Daylight	8500
#5	18:30		Twilight	5191
#6	21:30		Street lamp	12202

* Camera Setting: 320 × 240, 24-bit RGB color, 20 fps and FOV of 45 degrees
* All DB were obtained in a vehicle speed range of ~30 km/h - 60 km/h

The filtering process scans the binary image in the horizontal direction searching for the start position and the end position of the dark pixels. If the width is not shorter than the minimum width of vehicles at that position, the dark pixels are marked as a shadow. Then, the bottom position of a detected vehicle is estimated to be the bottom position of the filtered shadow. There is no information that indicates if the detected region is either below or above the actual vehicle region. Therefore, we scan the bottom of the filtered shadow around the bottom position of the detected vehicle region. The shadow searching range 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 3 presents the whole process of this shape filtering algorithm.

3.4. Dealing with Occlusion by Forward Vehicles

In the lane detection and tracking process, the falsely detected lane candidates on the rectangular regions of forward vehicles are eliminated using the result of the forward vehicle detection. Every detected lane candidate defined in sections 2.1 and 2.3 is tested if it is in these regions. If a candidate is in one of the regions, then we conclude that the candidate is not valid and it is not used as a measurement for the Kalman correction process.

4. EXPERIMENTAL RESULTS

4.1. Camera Installation and Calibration

For the experiment, a CCD camera (Point Grey FL2-03S2C) was installed on the front windshield of our experimental vehicle to be directed forward and slightly downward to obtain the road image sequence. Then, an IPT matrix, which maps the image coordinates to the world coordinates is calculated under the assumption that the vehicles are on a flat road. First, we specified markers and measured the world coordinates (X, Z) of the markers in

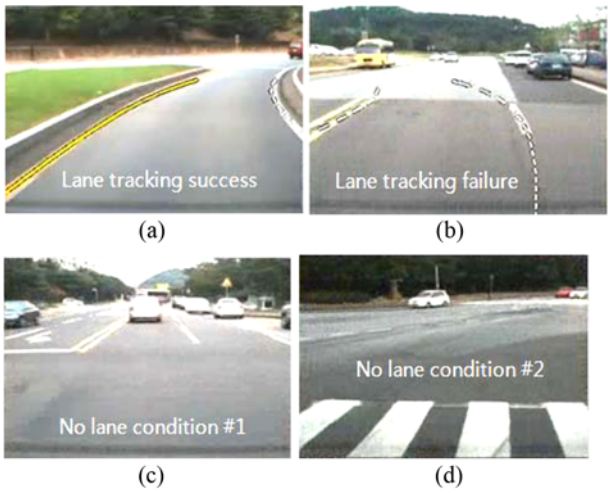


Figure 15. Definition of a lane tracking success (a), lane tracking failure (b), and no lane conditions (c, d).

Table 5. Overall test results of the lane tracking.

DB	True positive rate (true positive/positive)	False positive rate (false positive/negative)
#1	96.91 % (18562/19153)	1.7 % (5/300)
#2	96.82 % (14340/14800)	2 % (5/250)
#3	88.15 % (8271/9383) [A]	5 % (5/100)
#4	96.36 % (8094/8400)	2 % (2/100)
#5	98.37 % (5008/5091)	0 % (0/100)
#6	89.27 % (1239/11546) [B]	6.7 % (44/656)

meters. During this step, we set the origin to be the front center of the experiment vehicle. Second, we manually obtained the image coordinates (u, v) of the markers. Figure 14a shows the hand-labeled markers in an image sample used to calculate the IPT. Then, we computed the elements of the 3 by 3 IPT matrix T^{-1} using the least squares method given in equations (3)~(5). Mathematically, the computation requires only four matches of image and world coordinates. However, for accuracy, we used 14 matching points that are spread over the entire 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 describe the relation for the entire area of interest correctly. The calculated IPT matrix has a mean square error of 0.068 m for the 14 matching points (Figure 14(b)).

4.2. Test of Lane Tracking with an Illumination Change

To test the lane detection, we constructed a database that included 6 image sequences from different times (5 from the daytime and 1 from the night time) in the real highway driving environment. Varying illumination conditions for

Figure 16. Examples of lane tracking results during the daytime.

different times of the day and night, tunnels, curved lanes, lane discontinuity, road markers, and lane changing situations are included in this database. Table 4 shows the details of the database.

We used 10 cm as the lane marking width and 20 as the threshold for template matching during the lane candidate extraction. These values are determined experimentally to achieve the best lane tracking performance. To calculate the lane detection rate and the false detection rate, we defined the success and the failure in lane tracking as the existence or the absence of lane. We defined a success in lane tracking when the tracked hyperbola lane equation did not deviate by more than one lane marking width (20 cm) from the lane pair on the image (ground truth) in the frontal range from 3 m to 30 m, as shown in Figure 15(a).

If the tracked lane equation deviates by more than 20 cm from the real lane pair, as shown in Figure 15(b), then the

result is a failure. For the lane existence, we assumed that no lane exists in the image if there is no lane in the frontal range from 3 m to 15 m (Figure 15(c)) or if some part of a zebra crossing is on the bottom of the image (Figure 15(d)). We also defined the true positive rate and the false positive rate as an indicator of the lane tracking performance, as given in (41) and (42).

$$\text{true positive rate} = \frac{\text{\# of success lane tracking}}{\text{\# of images on which lane exists}}, \quad (41)$$

$$\text{false postitve rate} = \frac{\text{\# of positive lane tracking}}{\text{\# of images on which lane does not exist}}. \quad (42)$$

The overall test results of the lane tracking using the database shown in table IV are presented in table 5. We find that the overall test performance of our algorithm is greater than 96 % for the true positive rate and less than 2 % for the false positive rate except in the case of DB #3 and DB #6. The reason for the low true positive rate in DB #3 [A] is that the road image was affected by specular light due to the very strong sun light at noon. If the specular light affects a small region of the image, then our algorithm does not fail to track the lanes because the Kalman filtering can cope with a small loss of lane candidates or noisy candidates. However, in the case of DB #3, the interference by the specular light was continuous and on most of the image region. Therefore, our algorithm failed to track the lane equation occasionally. The low true positive rate in DB #6 [B] is because of the low illumination condition at night and the specular light projected on the frontal window. During night time illumination conditions, the performance of the lane tracking depends on how adjustable the camera is with respect to the lighting condition and how intense the specular light is from the street lamps or the tail lights of vehicles in front of the test vehicle.

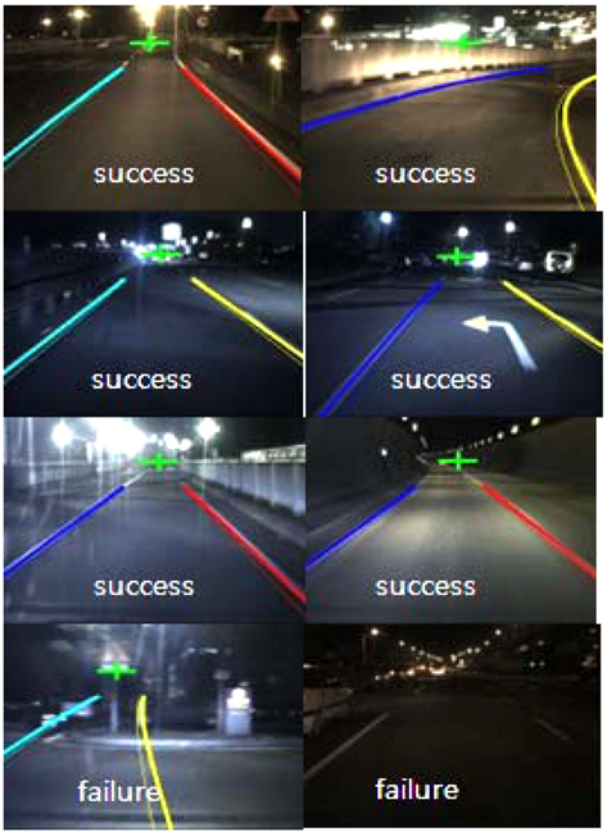


Figure 17. Examples of lane tracking results during night time.

Figure 16 shows some examples of lane tracking results using our algorithm. The blue and sky blue lines represent the tracked left lane. The red and yellow lines represent the tracked right lane. Our algorithm could track straight and curved lanes in several illumination conditions of specular

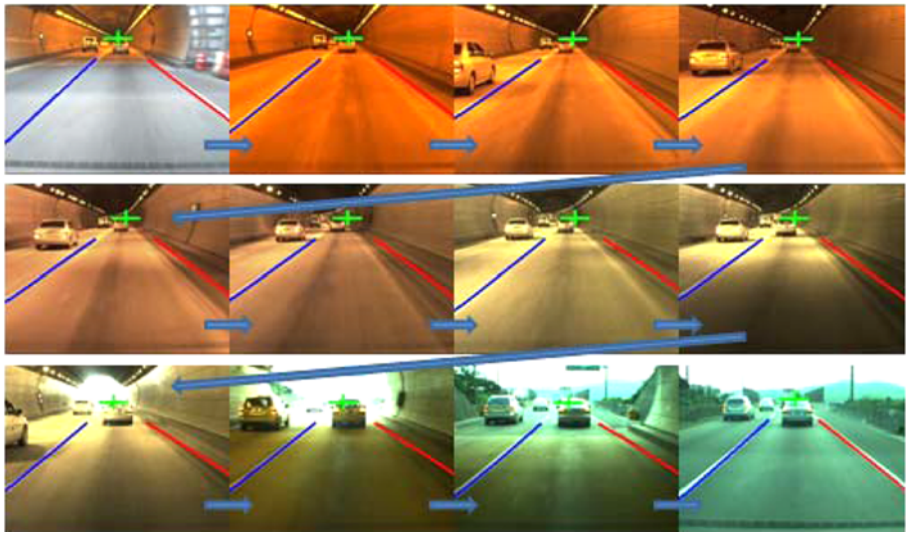


Figure 18. Result of lane tracking going into and out of a tunnel.

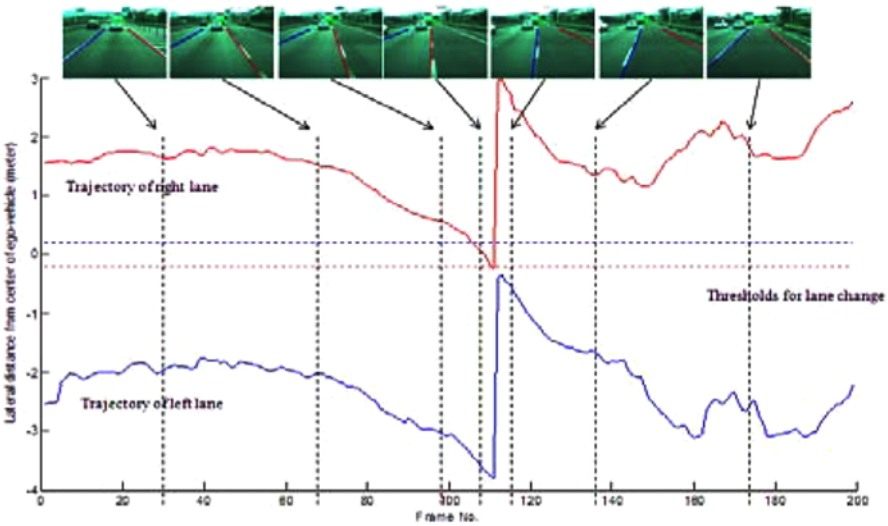


Figure 19. Lane tracking during a lane changing situation.

light and in tunnels, with speed bumps, and road traffic signs (six left images marked as successes in Figure 16) although there were some failures with a small mismatch between the tracked lane and real lane just after a speed bump (top right image marked as a failure in Figure 16) or in the beginning of strong curved lanes that had a curvature radius of less than 35 m (bottom right image marked as a failure in Figure 16).

Figure 17 shows some examples of lane tracking at night. Although the brightness of the image was low and some specular light from the street lamps and vehicles in front of the test vehicle existed, the lane tracking results were correct, as shown in the six left images of Figure 17. When the brightness of the image was very low (bottom right of Figure 17) or the specular light was strong enough to affect the entire image (top right of Figure 17), our algorithm occasionally failed to track the lane.

Figure 18 shows the lane tracking result when the vehicle is going in and out of a tunnel. When the vehicle is going into the tunnel (the images on the first row of Figure 18), there is an abrupt change in the white balance and a reduction in the brightness. After allowing some time for the camera to adjust to the illumination, the brightness increases to a normal condition and the white balance becomes stable (the images on the second row of Figure 18). When the vehicle is exiting the tunnel, the brightness of the image abruptly increases and the white balance changes again because of the light coming from outside the tunnel (the bottom row of Figure 18).

Because of these abrupt changes in the illumination, the number of lane candidates became too small to track the lane with the detection technique alone. However, our algorithm using RANSAC plus Kalman filtering can track the lane successfully in this situation because it can predict the lane equation without the lane candidates for a short duration of time, as shown in Figure 18.

4.3. Test of Lane Tracking during a Lane Changing Situation

We tested our lane tracking algorithm using an image sequence that records a lane changing situation. Figure 19 shows how our state transferring method can deal with a lane changing situation. Here, the solid red line on the graph represents the trajectory of the lateral position of the right lane, the solid blue line the trajectory of the left lane, the dashed red line the threshold of the rightward lane change, and the dashed blue line the threshold of the leftward lane change. We set the threshold to 0.2 m for the leftward lane change and -0.2 m for the rightward lane change. From frame no. 50 to 110, the ego-vehicle was approaching the right lane, which is the red line on the left most image of Figure 19. When the lateral position met the threshold for the rightward lane change, around frame no. 111, the right lane equation was transferred to the left lane

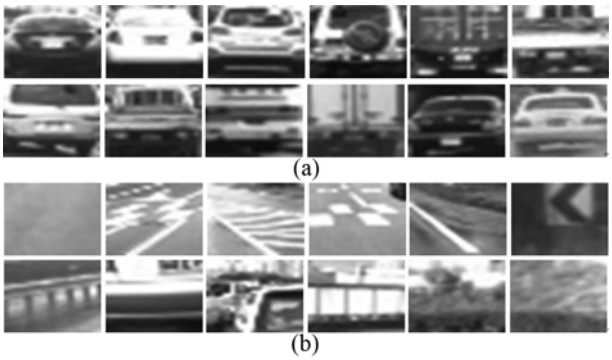


Figure 20. Examples of training images: (a) vehicle images and (b) non-vehicle images.

equation and a new right lane was estimated, as defined in (31). As the ego-vehicle was approaching the center of the new lane pair, our algorithm kept tracking the new lanes without an additional detection step.

4.4. Test of the Vehicle Detection and dealing with Occlusions

The AdaBoost vehicle detector was trained with a number of 24 by 24 sized training images in which 671 positive (vehicle) samples and 24,593 negative (non-vehicle) samples were included. The positive sample set includes various types of vehicles, i.e., sedans, buses, trucks, and SUVs (Figure 20(a)). The bottom of the positive training image is aligned at the position where the road and the rear tires make contact. Additionally, the left and right sides of the positive training image are aligned at the left and right sides of the vehicles, respectively. Negative image samples include the surface of the road, the traffic signs on the road, parts of vehicles, and many other objects that can be seen on the road (Figure 20(b)).

During urban driving, drivers should pay attention to the three closest vehicles in the left, the right and the ego lanes because it is possible that those vehicles can collide with the ego-vehicle if the drivers are careless. Therefore, we counted the detections of those vehicles and the false detections in the three lanes under various lighting conditions in urban traffic. The vehicle detection rate was defined as the ratio of the number of positive detections to the number of vehicles, as defined in (43).

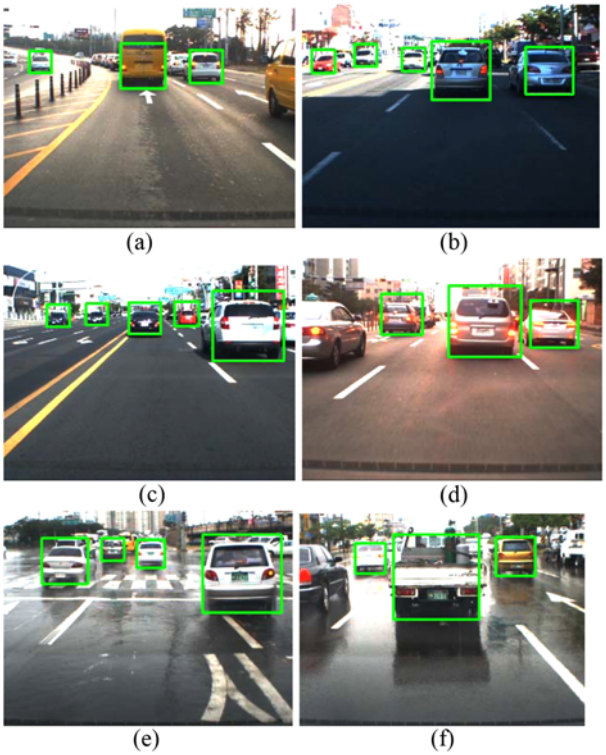


Figure 21. Illumination conditions of the test images obtained at dawn (a), in the morning (b), at noon (c), in the afternoon (d), and on a rainy day (e, f).

Table 6. Vehicle detection rates and false detection rates under various lighting conditions.

Time	Detection rate	False detection rate	Range of forward vehicle position
Dawn	79.37 %	7.33 %	5 m (near) ~ 50 m (far)
Morning	70.83 %	16.78 %	
Noon	90.77 %	7.81 %	
Afternoon	85.57 %	8.42 %	

* Average bottom line error is only the amount of several pixels

$$\text{Detection rate} = \frac{\#\text{ of positive detections}}{\#\text{ of vehicles}}. \quad (43)$$

The false detection rate was defined as the ratio of the number of false detections to the number of detections, as defined in (44). Detections of non-vehicle regions and parts of vehicles were considered false detections.

$$\text{False detection rate} = \frac{\#\text{ of false detections}}{\#\text{ of detections}}. \quad (44)$$

The detection rate and the false detection rate are presented in table 6.

The lighting conditions at noon (Figure 21(c)) were the best for our vehicle detection algorithm. In the morning, shadows of buildings and trees are on the road and on vehicles (Figure 21(b)), which caused lighting conditions to frequently change. Therefore, the vehicle detection rate and false detection rate were the worst in the morning. Under the lighting conditions at dawn (Figure 21(a)) and in the afternoon (Figure 21(d)), vehicles are shown less clearly in the images because the input images are darker than they are in the daytime. Back lighting sometimes makes the situation worse. Therefore, the detection rates at dawn were worse than the detection rate at noon. On rainy days, the shape and the intensity of the shadow appear differently than the way they appear on sunny days, and the appearances of the vehicles are reflected on the wet road surface (Figure 21(e) and (f)). The proposed algorithm can detect vehicles on a rainy day because our algorithm does not utilize the shadow below the vehicles in the detection. For the detection range in the forward vehicle position, vehicles in the range of 5 m to 50 m were considered. Although vehicles up to 100 m that corresponded to the minimum size of the scanning window (24 × 24 pixels) can be detected with low precision, we used a limited range of up to 50 m for more accurate results.

However, there are two major factors that negatively affect our algorithm's detection rate. One factor is the raindrops on the front windshield, which cause the input

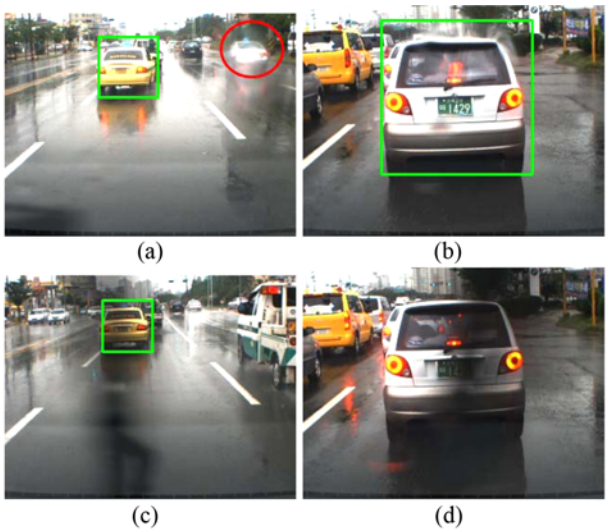


Figure 22. Major factors that degrade the vehicle detection rate on a rainy day: (a, b) distortion caused by raindrops on the front windshield and (c, d) occlusion by the wipers.

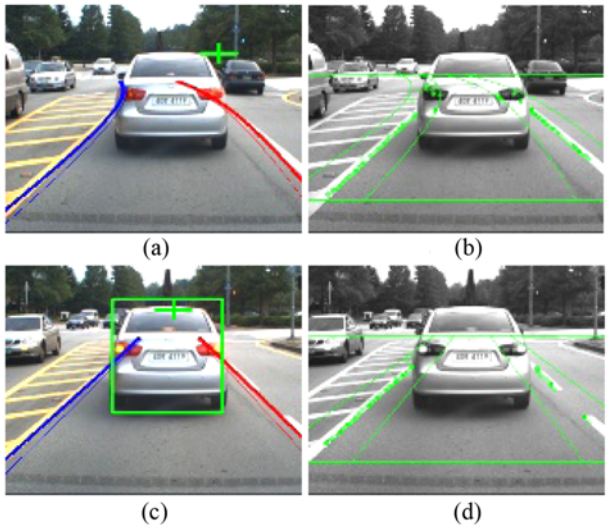


Figure 23. Result of occlusion handling.

images to be locally distorted. The cause of this problem is that raindrops on the front windshield are not immediately wiped off as they fell on the front windshield. If the vehicle is completely distorted (Figure 22(a)), our algorithm fails to detect the vehicle. Conversely, our algorithm can partially detect distorted vehicles (Figure 22(b)). Another factor is occlusion by the window wipers. In the case where the wipers on the window make a vehicle appear darker, our algorithm could detect the vehicle (Figure 22(c)). However, wipers can interfere with the verification of a vehicle using a horizontal edge, which causes our algorithm to fail to detect the vehicle (Figure 22(d)). These problems are inevitable on a rainy day. Therefore, the vehicle detection rate of our algorithm is degraded on a rainy day.

Figure 23 shows an example of dealing with occlusions caused by forward vehicles. When a forward vehicle was close to the ego-vehicle, several false lane candidates were detected in the vehicle region (Figure 23(b)) because the pattern of tail lights of the forward vehicle is similar to the lane pattern. Therefore, the tracked lane equation was affected by the false candidates and resulted in the wrong equation (Figure 23(a)). Using the result of the vehicle detection, the false lane candidates were omitted (Figure 23(d)), and the calculated lane equation without the false lane candidates was corrected (Figure 23(c)).

4.5. Processing Time

Our algorithm was tested on a laptop computer (Intel ® Core™2 T7200, 2.00 GHz, 1.00 GB RAM) with a 320 by 240 input image size sequences. Table 7 shows the time consumption for each part of our algorithm. In the detection mode of our algorithm (initial lane detection in sections 2.1 and 2.2), the processing time without vehicle detection is 36 ms per frame, which is almost nine times greater than the tracking mode, which was 4 ms. Because of lane candidate extraction on a small ROI, no clustering, and very simple RANSAC and Kalman filtering for the lane equation in the tracking mode, the time consumption for each portion of the tracking mode is much less than that of the detection mode. Even with the most time-consuming part, which is the forward vehicle detection, time consumption in the detection mode is 73 ms, which is greater than the 41 ms observed in the tracking mode. Therefore, continuous tracking in abnormal cases, especially during a lane changing situation, is much more efficient than repeating the detection step. Because our algorithm can continue tracking in this situation with the state transferring technique, this system can save much of the computational time that is expected when repeating the

Table 7. Time consumption for each part of the proposed algorithm.

Modes\Operations	Lane candidate extraction	Lane candidate clustering	Lane equation calculation	Forward vehicle detection	Lane change and occlusion handling	Total time
Detection	10 ms	20 ms	5 ms	37 ms	1 ms	73 ms
Tracking	1 ms	.	2 ms	37 ms	1 ms	41 ms

detection.

5. CONCLUSION

This paper proposed a vision-based fusion technique for lane tracking and forward vehicle detection to handle the lane occlusions, lane changing situations, abrupt illumination changes, and pitch motions. First, the initial lane equation was calculated using the template matching method for lane candidates and several constraints for the lane pair. Then, RANSAC and a Kalman filter were sequentially used to calculate the new lane equation from the lane candidates found in the adaptive ROI from the next image frame. In this process, we used a state transferring technique to deal with the lane changing situation without stopping the tracking. At the same time, we detected all of the rectangular regions of forward vehicles using the AdaBoost object detector followed by edge and shadow filtering. The detected vehicle regions were used to prevent falsely detected lane candidates from being used to track the lane equation. Finally, to prevent the algorithm from diverging due to the car shaking, a constraint for the vanishing point of the hyperbola lane equation pair is used.

Our algorithm could find the lane equation very quickly and robustly against illumination changes and discontinuities in the lane, which usually occur when the ego-vehicle is going into or out of a tunnel. Due to the combination of the vehicle detection results, RANSAC, and the Kalman filtering, falsely detected lane candidates could be omitted and more accurate and stable lane tracking was possible. Our algorithm could obtain the correct lane tracking results in the situation of pitch motion on speed bumps with reasonable constraints for the vanishing point of the lane equation. The overall lane tracking performance of our algorithm was over 96 % for our own road image database with varying illumination conditions during different times of the day except when the specular light was too strong or lighting conditions were too dark to differentiate the lane from the road. Additionally, the lane transferring technique of our algorithm made it possible to maintain lane tracking continuously in spite of the lane changing situation with a very low computational cost: 4 ms on a standard laptop computing system.

However, the vehicle detection, the most time-consuming portion of our algorithm, still needs to be improved for faster processing with an object tracking technique, i.e., robust second order minimization (Benhimane and Malis, 2004). Additionally, when very strong specular light exists on the road or road structures, our algorithm did not work well. Dealing with specular light is one of the hardest problems in the field of computer vision research because image restoration from damaged images by specular light needs a complicated computation and is not appropriate for real-time applications. Some hardware-based techniques, such as using optical filters to prevent intensity saturation in the image, can be a rough solution for this problem. These two types of problems, vehicle tracking and dealing with specular light, will remain the focus of future work.

ACKNOWLEDGEMENT–This research was supported by IMAGENEXT, Korea, under the development of intelligent multifunctional multi-camera system platform for high-safety car in the urban environment.

REFERENCES

Benhimane, S. and Malis, E. (2004). Real-time image-based tracking of planes using efficient second-order minimization. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 943–948.

Borkar, A., Hayes, M., Smith, M. T. and Pankanti, S. (2009). A layered approach to robust lane detection at night. *IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, 51–57.

Chang, J. Y., Hu, W. F., Cheng, M. H. and Chang, B. S. (2002). Digital image translational and rotational motion stabilization using optical flow technique. *IEEE Trans. Consumer Electronics* **48**, **1**, 108–115.

Chen, Q. and Wang, H. (2006). A real-time lane detection algorithm based on a hyperbola-pair model. *IEEE Intelligent Vehicles Symp.*, 510–515.

Choi, S., Kim, T. and Yu, W. (2009). Robust video stabilization to outlier motion using adaptive RANSAC. *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 1897–1902.

Fishchler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24**, 381–395.

Haralick, R. M. (1989). Monocular vision using inverse perspective projection geometry: Analytic relations. *IEEE Conf. Computer Vision and Pattern Recognition*, 370–378.

Haykin, S. (2001). *Kalman Filtering and Neural Networks*. John Wiley & Sons. New York.

Hsueh, P. C., Yeh, C. W., Cheng, C. H., Hsiao, P. Y., Jeng, M. J. and Chang, L. B. (2009). Realize a mobile lane detection system based on pocket PC portable devices. *Int. Conf. Signal Processing, Computational Geometry & Artificial Vision*, 135–140.

Jung, H. G., Lee, Y. H., Kang, H. J. and Kim, J. (2009). Sensor fusion-based lane detection for LKS+ACC System. *Int. J. Automotive Technology* **10**, **2**, 219–228.

Lienhart, R. and Maydt, J. (2002). An extended set of haar-like features for rapid object detection. *Int. Conf. Image Processing*, 900–903.

Matsushita, Y., Ofek, E., Ge, W., Tang, X. and Shum, H. Y. (2006). Full-frame video stabilization with motion inpainting. *IEEE Trans. Pattern Analysis and Machine Intelligence* **28**, **7**, 1150–1163.

Tsai, L. W., Hsieh, J. W., Chuang, C. H. and Fan. K. C. (2008). *Lane Detection using Directional Random Walks. IEEE Intelligent Vehicles Symp.*, 303–306.

Viola, P. and Jones, M. (2004). Robust real-time face detection. *Int. J. Computer Vision* **57**, **2**, 137-154.

Welch, G. and Bishop, G. (2006). *An Introduction to the Kalman Filter*. TR95-041. University North Carolina at Chapel Hill.

Wu, B. F., Lin, C. T. and Chen, Y. L. (2009). Dynamic calibration and occlusion handling algorithm for lane tracking. *IEEE Trans. Industrial Electronics* **56**, **5**, 1757–1773.

Yang, J., Schonfeld, D. and Mohamed, M. (2009). Robust video stabilization based on particle filter tracking of projected camera motion. *IEEE Trans. Circuits and Systems for Video Technology* **19**, **7**, 945–954.