# Lane Detection and Tracking Using a New Lane Model and a Distance Transform

Ruyi Jiang[1], Reinhard Klette[2], Shigang Wang[1], and Tobi Vaudrey[2]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] The University of Auckland, Auckland, New Zealand

**Abstract.** Lane detection is an important component of driver assistance systems (DAS), and highway-based lane departure solutions have been in the market since the mid 1990*s*. However, improving and generalizing vision-based lane detection solutions remains to be a challenging task. Particle filtering of boundary points is a robust way to estimate lanes. This paper introduces a new lane model in correspondence to this particle filter-based approach. Furthermore, a modified version of an Euclidean distance transform is applied on an edge map to provide information for boundary point detection. In comparison to the edge map, properties of the distance transform support improved lane detection including a novel initialization method. Two lane tracking methods are also discussed while focusing on efficiency and robustness, respectively. Finally, the paper reports about experiments on lane detection and tracking.

**Keywords.** driver assistance system, lane detection, lane tracking, particle filter, distance transform

## 1 Introduction

Lane detection plays a significant role in driver assistance systems (DAS), as it can help one to estimate the geometry of the road ahead, as well as the lateral position of the ego-vehicle on the road. Lane detection is used in intelligent cruise control systems, for lane departure warning, road modeling, and so on. Typically, lane detection and tracking are used for localizing lane boundaries in given road images.

Lane detection and tracking have been widely studied for driving on a freeway [6, 16] or an urban road [22], for single [6, 23] or multiple [1, 17] lanes, with [3] or without [24] lane marks, based on region (texture [29] or color [8]) or edge [19] features. Various models have been applied to describe the borders of a lane, such as *piecewise linear segments* [19], *clothoids* [6, 16], *parabola* [11], *hyperbola* [25, 15], *splines* [23, 24], or *snakes* [24, 28]. Several lane detectors have been implemented and named in the literature, such as GOLD [3], SCARF [8],

RALPH [20], MANIAC [10], or LANA [14, 15]. Though some of them work well in a particular environments, generally, it is a challenging task to robustly detect lanes in various situations. Reasons for this difficulty are, for example, as follows:

- difficulties caused by various conditions of illumination and poor quality of lane markings,
- violation of some commonly used assumptions, such as constant road width, parallel left and right lane boundaries, or the use of other simplified geometric road models (e.g., parabolic boundaries),
- difficulties caused by surrounding objects, such as trees on the roadside or above the road, occlusions caused by pedestrians or other vehicles on the road, and
- missing information in the real world about the actual lanes or roads (such as no lane markers, or unpaved roads).

For the reasons stated above, it is proposed in [22] that *weak models* (i.e., with no assumption about the global shape of a lane) are preferable than *strong models*, which use several parameters to model the global geometry of a lane. This paper introduces a new weak road model for lane detection and tracking. Instead of modeling global road geometry, this new model only constrains relations between points on the left and right lane boundaries. Tracking based on these points in the bird's-eye image (using a particle filter) provides lane detection results. Furthermore, a modified version of a standard Euclidean Distance Transform (EDT) is applied on the edge map of the bird's-eye image. Utilizing some beneficial properties of this distance transform for lane detection, this paper specifies an innovative initialization method which also uses a particle filter. Furthermore, the distance transform also provides more information (such as the centerline of a lane) compared to general edge-based lane detection.

This paper is organized as follows: Section 2 describes a new lane model. Lane detection using this new lane model, as well as particle filter is introduced in Section 3. Low-level image processing, especially the distance transform, is presented in detail in the lane detection section. Two lane tracking methods focusing on efficiency or robustness, respectively, are discussed in Section 4. Experimental results are given in Section 5. Finally, conclusions are provided in Section 6.

## 2 A New Lane Model

The lane model as used in this paper is illustrated in Figure 1; the 3D view also contains the $xy$-coordinate system on the *ground manifold*. In this paper we assume a *ground plane*. ([26] proposes cubic B-splines for modeling the ground manifold, which is more correct.)

Five parameters $x_c$, $y_c$, $\alpha$, $\beta_1$, and $\beta_2$ are used to model opposite points $P_L$ and $P_R$, located on the left or right lane boundary, respectively. $P_C = (x_c, y_c)$ is the *centerline point* of a lane in the ground plane. $\alpha$ is the *zenith angle* above $P_C$, defined by an upward straight line segment between $P_C$ and the *zenith* $P_Z$

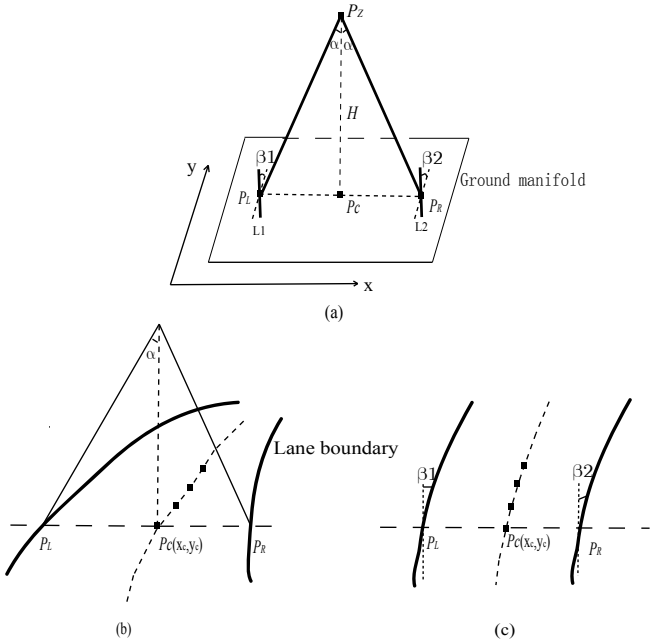


**Fig. 1.** Lane model as used in this paper. (a) 3D lane view; boundaries are drawn in bold. (b) Perspective 2D lane view in the input image. (c) Bird's-eye image of the lane. Slope angles $\beta_1$ and $\beta_2$ are shown in the 3D view and the bird's-eye images; the zenith angle $\alpha$ in the 3D and the projective view. See text for further explanations.

of fixed length $H$, and a line incident with $P_Z$ and either $P_L$ or $P_R$. As the height $H$ is fixed, the width of a road at points $P_L$ and $P_R$ can easily be calculated as

$$2H \cdot \tan(\alpha)$$

$\beta_1$ and $\beta_2$ are the *slope angles* between short line segments $L_1$ and $L_2$ and a vertical line in the ground plane; the two short line segments $L_1$ and $L_2$ are defined by a fixed length and local approximations to edges at lane boundaries (e.g., calculated during point tracking). Ideally, $L_1$ and $L_2$ should coincide with tangents on lane boundaries at points $P_L$ and $P_R$; in such an ideal case, $\beta_1$ and $\beta_2$ would be the angles between tangential directions of lane boundaries at those points and a vertical line. By applying this model, a lane is identified by two lane boundaries, and points are tracked along those boundaries in the bird's-eye image.

This model does not use any assumption about lane geometry, and is applicable to all variates of lanes. For example, this also allows us to detect a lane with varying width, even in a single image. As $\beta_1$ and $\beta_2$ are calculated separately, a lane may also have nonparallel left and right boundaries. Lane detection and tracking methods, using this model, are discussed in Sections 3 and 4.

## 3 Lane Detection Using a Particle Filter

For lane detection in a single image, a particle filter may be used to track points along lane boundaries; see [22] which simply uses coordinates of boundary points as the tracking state. Below we show that our lane model provides more robust tracking. Furthermore, a novel initialization method is adopted based on a distance transform applied to the bird's-eye edge map. The whole procedure of lane detection is illustrated in Figure 2 by an example.

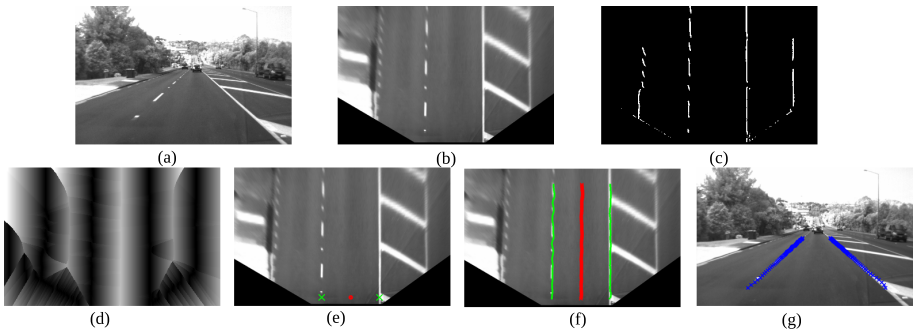


**Fig. 2.** The overall work flow of lane detection. (a) Input image. (b) Bird's-eye image. (c) Edge map. (d) Distance transform. (e) Initialization. (f) Lane detection results, shown in the bird's-eye image. (g) Lane detection results, shown in the input image.

The algorithm starts with mapping the perspective input image into a bird's-eye view (i.e., an orthogonal projection toward the zenith), using a homography defined by four vertices of a rectangle in the bird's-eye view. An edge detection method, as introduced in [3] for lane detection, is then adopted to detect lane-mark-like edges in the bird's-eye image. After binarization of the resulting edge map and denoising of isolated edge points or small blobs, a *real orientation distance transform* (RODT) is applied; see Section 3.1 for a specification of this transform. The resulting distance map allows us to design a novel initialization method for finding the initial boundary point. This point is used to initialize the parameters of the particle filter, for tracking further boundary points through the whole image; a lane is finally detected.

### 3.1 Low-level image processing

Low-level image processing is composed of three steps: bird's-eye-view mapping, edge detection with denoising, and distance transform.

**Bird's-eye-view mapping.** As in [13], a four-points correspondence is used for the mapping from the input image into the bird's-eye image. The mapping is achieved by selecting four points when calibrating the ego-vehicle's camera(s),

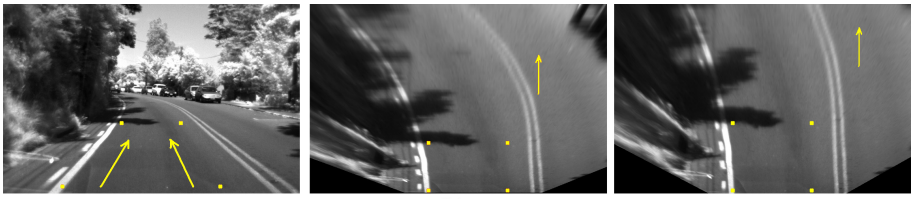


**Fig. 3.** (a) Input Image. (b) and (c) are bird's-eye images based on different distance definitions. Four-point correspondence (points shown in yellow) is established by calibration; the driving direction is indicated by the arrow.

and by using the locally planar ground manifold assumption. One benefit of the bird's-eye image is that the used distance scale can be adjusted by selecting different sets of four corresponding points (i.e., by scaling the "length" of the rectangle). This proved to be useful for detecting discontinuous lane markers as well as for further forward looking situations. Also, lane marks in the bird's-eye image have a constant width, which will be utilized for edge detection.

**Edge detection and noise removal.** We recall an edge detection method as introduced in [3]. Black-white-black edges in vertical direction are detected in the bird's-eye image by a specially designed simple algorithm. Every pixel in the bird's-eye image, with value $b(x, y)$, is compared to values $b(x - m, y)$ and $b(x + m, y)$ of its horizontal left and right neighbors at a distance $m \geq 1$ as follows:

$$B_{+m}(x, y) = b(x, y) - b(x + m, y)$$
$$B_{-m}(x, y) = b(x, y) - b(x - m, y)$$

and finally, using a threshold $T$, the edge map value will be

$$r(x, y) = \begin{cases} 1, & \text{if } B_{+m} \geq 0,\ B_{-m} \geq 0, \text{ and } B_{+m} + B_{-m} \geq T \\ 0, & \text{otherwise} \end{cases}$$

This edge detection method has the following properties. First, $m$ can be adjusted to fit various widths of lane marks. Second, pixels within a lane mark are all labeled as being edge pixels, which is different from gradient-based edge operators (e.g. Canny, Sobel). This greatly improves the robustness in detecting points of lane marks. Third, shadows on the road surface do not influence edge detections at lane marks. Thus, the edge detection method can be used under various illumination conditions. Finally, horizontal edges are not detected. For an example of edge detection, see Figure 4.

Edge detection as introduced above may generate some isolated small blobs (including single points) besides the edges of real lane marks. These noisy blobs will greatly affect the result of the following distance transform (see Fig. 6; the used distance transform will be discussed in Section 3.1). We remove such noise by, first, finding the isolated blobs, second, setting them to zero (non-edge

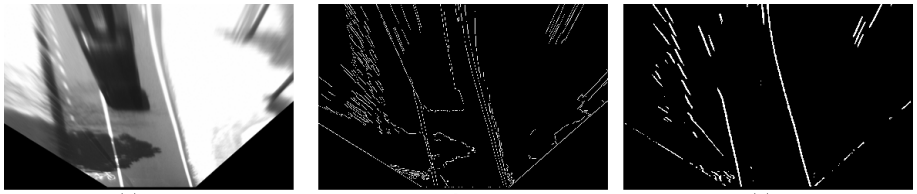


**Fig. 4.** Edge detection. (a) Bird's-eye image. (b) Edge detection using the Canny operator. (c) Edge detection following [3].

pixels). Two centered small windows (inner and outer) are used (see Fig. 5), where the outer window is slightly larger than the inner one in width and height. The isolated blobs can be detected by moving these two windows (concurrently) through the whole edge map, and comparing the sums of edge values within them. If the two sums are equal (i.e., the gap between both windows does not contain any edge point), then edge blobs in the inner window are identified as being isolated, and set to be zero. For computational efficiency, an integral image of the edge map is used for calculating the sum in those small windows.

**Distance transform.** The distance transform applied to the binary edge map labels each pixel with the distance to the nearest edge pixel (see [21] for details). Edge pixels are obviously labeled by 0, and this is shown as black in the generated distance map. Pixels "in the middle of a lane" are supposed to receive large labels, shown as bright pixels in the distance map (see Figure 7).

The Euclidean distance transform (EDT) is in general the preferred option, using the Euclidean metric for measuring the distance between pixels. [7] proved that a 2D EDT can be calculated by two 1D EDTs, and this greatly improves

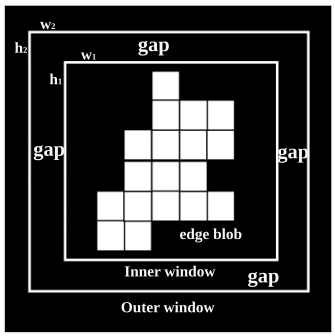


**Fig. 5.** Detection of isolated blobs in the binarized edge map. Inner window and outer window move at the same time through the edge map. When the gap between the inner and outer window contains no edge pixel, an isolated blob is detected in the inner window.

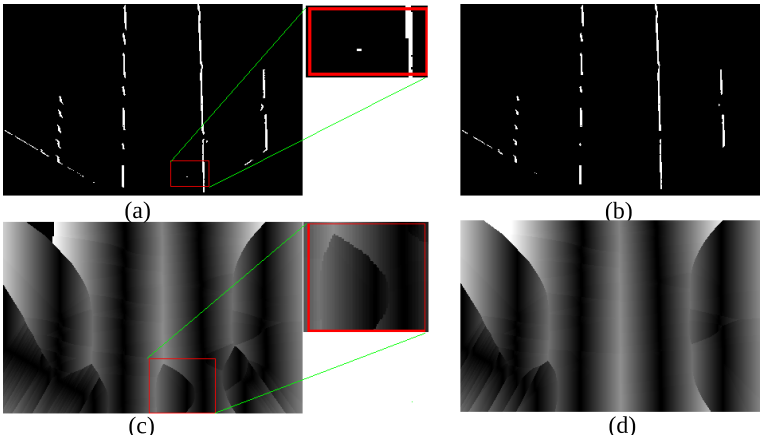


**Fig. 6.** Effect of isolated ("noisy") points in an edge map for the distance transform. (a) Noisy edge map with an isolated edge point in the middle of the lane. (b) Denoised edge map. (c) RODT based on (a). (d) RODT based on (b).

the computation efficiency. A modified EDT was proposed in [27], called *orientation distance transform* (ODT). This divides the Euclidean distance into a contributing component in row and column direction. (Note that the use of a 4- or 8-distance transform would *not* lead to the same row and column components; however, practically there should be not a big difference with respect to the given context.) A complex number is assigned to each pixel by the ODT, with the distance component in the row direction as the real part, and the distance component in the column direction as the imaginary part. Then the magnitude and the phase angle of such a complex number at a non-edge pixel represent the

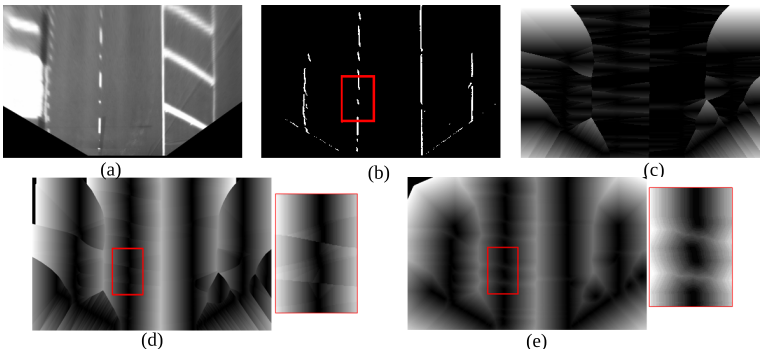


**Fig. 7.** EDT and ODT on a bird's-eye road image. (a) Bird's-eye road image. (b) Binary edge map (the area in the rectangle is a discontinuous lane mark). (c) EDT. (d) Real part of ODT (absolute value). (e) Imaginary part of ODT. (c)(d)(e) have been contrast adjusted for better visibility.

Euclidean distance and the orientation to the nearest edge pixel, respectively. Note that distance component in row direction is signed, with a positive value indicating that the nearest edge point lies to the right, and a negative value if it is to the left. See Figure 7 for an example. The imaginary part is mostly dark because nearest edge pixels are in general in the same row, due to the applied edge detection method. Thus, we decided to ignore the imaginary part.

This paper uses only the Euclidean distance in row direction, and we call this the *real orientation distance transform* (RODT). The RODT of our edge map offers various benefits. First, the initialization of lane detection becomes much easier (to be discussed in Section 3.2). Second, discontinuous lane marks will make almost no difference with continuous ones in the RODT (and this is different to the EDT of the edge map), as illustrated in Figure 7. Third, more information about the lane (e.g. information on centerline or road boundary, to be indicated in Sec. 3) is provided by the distance transform compared with the edge map. Generally, a (non-edge) pixel on the centerline of a lane will have a local maximum in distance to the lane boundaries. Thus, combined with the lane model introduced in Sec. 2, a point with a high distance value is likely to be a centerline point $P_C$. The usefulness of these properties of the RODT will be discussed in the following sections.

The distance transform is sensitive to isolated points or blobs in a lane detection situation. As illustrated in Fig. 6, a single edge point in the middle of a lane would already greatly change the distance value for the surrounding pixels. Thus, a very simple denoising method on the edge map, as introduced in Section 3.1 is necessary, and proved to be sufficient.

### 3.2 Initialization

The aim of the initialization step is to find an initial value (e.g., the $x$-coordinate of a point $P_L$ or point $P_R$ in a selected image row) for the specified model. In [22], a clustered particle filter is used in order to find a start point on a lane boundary. In distinction to this, we fully utilize the distance map to find the first left and right boundary points. In a pre-defined *start row* (near to the bottom) of the bird's-eye image, a search is conducted, starting at the middle of the row, for a pixel which has a positive distance value but a negative distance value at its left neighbor (see Figure 8). When such a pixel is found, the left and right boundary

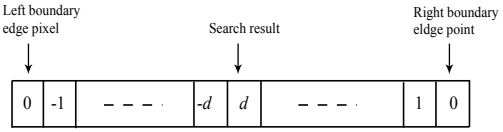


**Fig. 8.** Illustration of the search procedure in the start row of the distance map. Note that the distance values are signed, as described in Section 3.1.

points in the start row are instantly known using the distance value of the found pixel and of its left neighbor.

For the initial state $X_0(x_{c_0}, \alpha_0, \beta_{1_0}, \beta_{2_0})$ of the particle filter, $x_{c_0}$ and $\alpha_0$ are initialized by using the detected left and right start points, while $\beta_{1_0}$ and $\beta_{2_0}$ are simply set to be zero.

### 3.3 Particle filter for lane detection

Particle filters are widely used for lane detection and tracking, such as in [22, 25]. This section discusses particle filtering for our new lane model. The state vector $X = (x_c, \alpha, \beta_1, \beta_2)^T$ to be tracked is defined by the parameters of the lane model, without $y_c$, as $y_c$ will be calculated incrementally by applying a fixed step $\Delta$, starting at row $y_{c_0}$ in the bird's-eye image. For the application of a particle filter, two models [9] are discussed in the following.

**The dynamic model.** The dynamic model $A$ is used to define the motion of particles in the image. The prediction value $\hat{X}_n$ is generated from $X_{n-1}$ by using $\hat{X}_n = A \cdot X_{n-1}$. We simply take $A$ as being the identity matrix, because of the assumed smoothness of the lane boundary.

**The observation model.** The observation model determines each particle's importance factor for re-sampling. Based on the RODT information, it is reasonable to assume that subsequent pixels on the lane boundary will have a distance value zero, and points on the centerline of a lane will have large distance values. In terms of our lane model, points $(x_{c_n}, y_{c_n})$ have large distance values, and $L_1$ and $L_2$ coincide with short lines of pixels which all only have small distance values.

Tracking step $n$ is identified by $y_{c_n} = (y_{c_0} + n \cdot \Delta)$. We calculate the lateral position of the left boundary point of the lane from the predicted state vectors, with $\hat{X}_n^i(\hat{x}_{c_n}^i, \hat{\alpha}_n^i, \hat{\beta}_{1_n}^i, \hat{\beta}_{2_n}^i)$ for the $i^{th}$ particle.

From now on, $P_L$ and $P_R$ only represent the lateral position of boundary points, for simplicity. The left position is calculated as follows:

$$P_L^i = \hat{x}_{c_n}^i - H \cdot \tan \hat{\alpha}_n^i$$

Next, the sum of the distance values along line segment $L_1$ is as follows:

$$S_{L_1}^i = \sum_{j=-L_1/2}^{L_1/2} \left| d\left(P_L^i + j \cdot \sin \hat{\beta}_{1_n}^i,\, y_{c_n} + j \cdot \cos \hat{\beta}_{2_n}^i\right) \right|$$

Here, $d(\cdot, \cdot)$ is the distance value of the RODT. Calculating $S_{L_2}^i$ in the analogous way, we obtain the $i^{th}$ importance factor

$$\omega_{dist}^i = \frac{1}{2\pi\sigma_1\sigma_2} \exp\left(-\frac{(S_{L_1}^i - \mu_1)^2}{2\sigma_1} - \frac{(S_{L_2}^i - \mu_2)^2}{2\sigma_2}\right)$$

For the centerline point $(x^i_{c_n}, y_{c_n})$, the importance factor is equal to

$$\omega^i_{center} = \frac{1}{\sigma_3\sqrt{2\pi}} \exp\left(-\frac{\left(\left|\frac{1}{d(x^i_{c_n}, y_{c_n})}\right| - \mu_3\right)^2}{2\sigma_3}\right)$$

where $\mu_k$ and $\sigma_k$ are constants, for $k = 1, 2, 3$ . The final observation model is given by the factors

$$\omega^i = \omega^i_{dist} \cdot \omega^i_{center}$$

## 4 Lane tracking

Lane tracking uses information defined from previous results to facilitate the current detection. Actually, there are two aims to utilize previous information: one is to improve the computational efficiency of the current detection by utilizing *a priori* knowledge; the other one is robustness, as there is more information available by combining the current and previous state. Efficiency and robustness sometimes cannot be achieved at the same time, and they might be biased due to the given (application) context. Generally, lane detection in some situations (such as on a highway) will be relatively easier compared to others (such as on an urban road), depending on road conditions and quality of lane marks. Difficulties as discussed in Section 1 mainly happen when detecting a lane in some challenging situation. In conclusion, when performing lane tracking, we will pay more attention to the computation efficiency for less challenging situations, but more to robustness for complex road situations. For these reasons, this section introduces two lane tracking methods: efficient lane tracking and robust lane tracking.

### 4.1 Efficient lane tracking

The efficient lane tracking method is for situations characterized by good road conditions and good quality lane marks (such as on a highway). It simply uses previously detected lane boundary points, adjusts them according to the ego-vehicle's motion model, and then offsets them according to values of the RODT on the current bird's-eye edge map.

Note that when a lane is detected (as in Section 3), it is reasonably represented just by two sequences $\{P_{L_n} : n = 0, 1, \ldots, N\}$ and $\{P_{R_n} : n = 0, 1, \ldots, N\}$ of points on its left and right lane boundaries in the bird's-eye image. Here, $N$ is determined by the forward-looking distance.

Tracking of a lane through an image sequence is then simplified as tracking of these two point sequences. Sequences $\{P^{(t)}_{L_n}\}$ and $\{P^{(t)}_{R_n}\}$, detected in frame $t$, are partly already driven through by the ego-vehicle at time $t+1$. The length of this already driven part is determined by the ego-vehicle's motion model (speed and yaw angle). The detection process of $\{P^{(t+1)}_{L_n}\}$ and $\{P^{(t+1)}_{R_n}\}$ at time $t+1$ is composed of three steps: adjustment caused by the driven distance and the

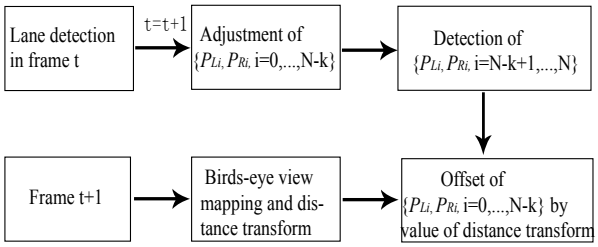


**Fig. 9.** Efficient lane tracking scheme.

variation in yaw angle, new points detection, and offset specification according to the values of the RODT in the bird's-eye edge map.

Because of the driven distance between frames $t$ and $t+1$, it holds (in principle) that

$$P_{L_n}^{(t+1)} = P_{L_{(n+k)}}^{(t)}, \qquad P_{R_n}^{(t+1)} = P_{R_{(n+k)}}^{(t)}, \qquad n = 0, 1, \ldots, N-k$$

Here, $k$ is determined by the driven distance between time $t$ and $t+1$, and is usually a small number. Driven distance is determined from odometry. Furthermore, points

$$\{P_{L_n}^{(t+1)} : n = 0, 1, \ldots, N-k\} \text{ and } \{P_{R_n}^{(t+1)} : n = 0, 1, \ldots, N-k\}$$

are obtained by adding some translation (according to $n$) caused by the variation in driving direction between $t$ and $t+1$.

For the detection of new points $\{P_{L_n}^{(t+1)} : n = N-k+1, \ldots, N\}$ and $\{P_{R_n}^{(t+1)} : n = N-k+1, \ldots, N\}$, note that $k$ is small and we also assume smoothness of lane boundaries. Thus, we simply start as follows:

$$P_{L_n}^{(t+1)} = P_{L_{n-1}}^{(t+1)}, \qquad P_{R_n}^{(t+1)} = P_{R_{n-1}}^{(t+1)}, \qquad n = N-k+1, \ldots, N$$

For further refinement, those predictions $\{P_{L_n}^{(t+1)}\}$ and $\{P_{R_n}^{(t+1)}\}$ from the previous result at frame $t$ are likely to be already located near the true points on the boundaries, as the variation of a lane is usually minor between two subsequent frames. – The adjustment

$$\begin{aligned} P_{L_n}^{(t+1)} &= P_{L_n}^{(t+1)} + d(P_{L_n}^{(t+1)}, y_{c_n}), \qquad n = 0, 1, \ldots, N \\ P_{R_n}^{(t+1)} &= P_{R_n}^{(t+1)} + d(P_{R_n}^{(t+1)}, y_{c_n}), \qquad n = 0, 1, \ldots, N \end{aligned}$$

of all $N+1$ points is finally achieved by information available from values of the RODT of the current bird's-eye edge map.

The described efficient lane tracking scheme is summarized in Figure 9. The main feature of this method is its computational efficiency. Experiments prove that the computation time needed for those three steps of efficient lane tracking

is almost negligible. Thanks to a minor variation between neighboring frames and the RODT, lane tracking results in highway-like situations are very much acceptable, except for rarely occurring outliers, which are always caused by some noisy non-boundary edge points (see experimental results as shown in Figure 15).

### 4.2 Robust lane tracking

Urban roads differ from highways, as they provide increased complexity of environments which results in difficult lane detection. In such situations, robustness is the dominant importance factor. A scheme for robust lane tracking is illustrated in Figure 10. This process is also composed of three main steps: first, adjustment of the result from the previous frame; second, detection of more boundary points (these two steps are the same as we do in the efficient lane tracking); third, lane detection as described in Section 3.

The initialization of the third step (i.e., initialization of the particle filter) is facilitated by the results of the first two steps. The only difference compared to lane detection is that when a pair of points on the left and right lane boundary is calculated from the tracking state as introduced in Section 3, two more pairs of points are generated. Then, a comparison among these three pairs of points is conducted by maximum likelihood, and this produces the final pair of detected points of the left and right boundary. As the first two steps have been discussed already in Section 4.1, as well as lane detection already in Section 3, only the comparison procedure remains to be described in this section.

As a result of the first two steps, we have $\{P_{L_n} : n = 0, 1, \ldots, N\}$ and $\{P_{R_n} : n = 0, 1, \ldots, N\}$ for frame $t+1$ (derived from sequences for frame $t$). We

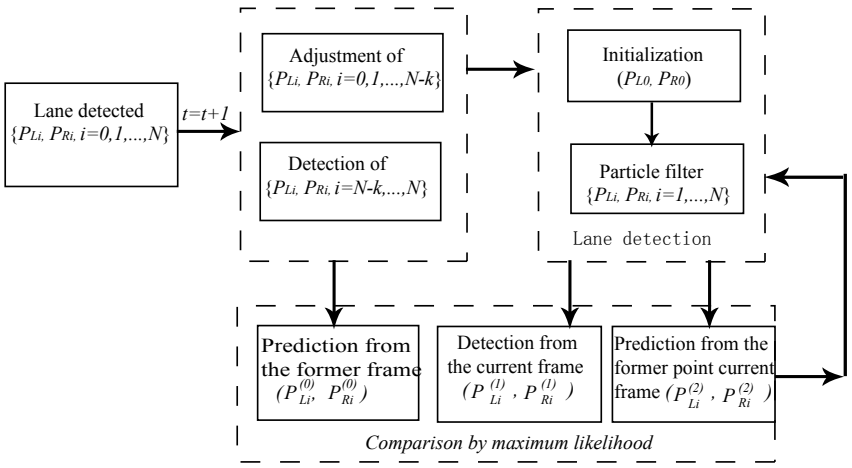


**Fig. 10.** Robust lane tracking scheme.

group them into pairs

$$\left(P^1_{L_n}, P^1_{R_n}\right)$$

for the following process. In lane detection, when a pair of points is detected by the particle filter on the left and right boundary at the $n^{th}$ tracking step, we index this pair as

$$\left(P^2_{L_n}, P^2_{R_n}\right)$$

Finally, the third pair

$$\left(P^3_{L_n}, P^3_{R_n}\right)$$

consists of the points as detected on left and right lane boundary at the $(n-1)^{th}$ step (i.e., for which we decided in the previous comparison step).

The selection of these three pairs of points has a physical meaning. The first pair $(P^1_{L_n}, P^1_{R_n})$ is a prediction from the previous frame. The second pair $(P^2_{L_n}, P^2_{R_n})$ combines the detected points in the current frame. The third pair $(P^3_{L_n}, P^3_{R_n})$ is a prediction in the current frame using the previous point. As these pairs carry information from different sources, their comparison, guided by maximum likelihood, will tell us which one best represents the points on the current lane boundaries.

Maximum likelihood estimation is used to judge the quality of the three pairs of points. A likelihood function $p(z|K)$, with $z$ for observed features, denotes the probability of observing a lane boundary point by the $K^{th}$ pair of points, for $K = 1, 2, 3$. The maximum likelihood estimation is written as follows:

$$P^* = arg \max_{K=1,2,3} p(z|K)$$

The determination of the likelihood function $p(z|K)$ uses information about the similarity of the width $\omega_{width}$ of the lane, smoothness $\omega_{smooth}$ of the lane boundary, as well as values $\omega_{distance}$ of the distance transform.

For the width of the lane, it is reasonable to assume that the width of a lane at the $n^{th}$ tracking step is the same as at the $(n-1)^{th}$. As $(P^3_{L_n}, P^3_{R_n})$ is predicted from the $(n-1)^{th}$ pair of boundary points for smoothness reasons, thus the similarity of width for the pair $(P^3_{L_n}, P^3_{R_n})$ is set to be 1, and the other two pairs are compared with it. Let

$$W^K = P^K_{R_n} - P^K_{L_n}, \qquad K = 1, 2, 3$$

be the width of each pair at the $n^{th}$ tracking step. Then,

$$\begin{aligned}
\omega^1_{width} &= a_1 \cdot \exp\left(-b_1 (W^1 - W^3)^2\right) \\
\omega^2_{width} &= a_2 \cdot \exp\left(-b_2 (W^2 - W^3)^2\right) \\
\omega^3_{width} &= 1
\end{aligned}$$

The smoothness weights of those three pairs are computed in an analogous way. The smoothness of pair $(P^3_{L_n}, P^3_{R_n})$ is set to be 1, and the other two pairs are compared with it. Let

$$C^K = (P^K_{L_n} + P^K_{R_n})/2, \qquad K = 1, 2, 3$$

be the center points at the $n^{th}$ tracking step. Then,

$$\begin{aligned}\omega^1_{smooth} &= a_3 \cdot \exp\left(-b_3(C^1 - C^3)^2\right)\\ \omega^2_{smooth} &= a_4 \cdot \exp\left(-b_4(C^2 - C^3)^2\right)\\ \omega^3_{smooth} &= 1\end{aligned}$$

Third, the values of the RODT at the points of those pairs is also used to evaluate the possibility of finding lane boundaries:

$$\omega^K_{distance} = a_5 \cdot \exp\left(-b_5(d(P^K_{L_n}, y_{c_n}) + d(P^K_{R_n}, y_{c_n}))^2\right), \qquad K = 1, 2, 3$$

Parameters $a_j$ and $b_j$ are constants, for $j = 1, 2, 3, 4, 5$. The final value of the likelihood function is calculated as follows:

$$P^{(K)} = \omega^K_{width} \cdot \omega^K_{smooth} \cdot \omega^K_{distance}, \qquad K = 1, 2, 3$$

The comparison of $P^1$, $P^2$, and $P^3$ will select one pair as being the final detection result at the $n^{th}$ step of the lane detection procedure.

## 5 Experiments

Experiments were conducted on images and sequences recorded with the test vehicle "HAKA1" of the *.enpeda..* project (see Figure 11). The size of images are $752 \times 480$ pixels.

Experimental results for lane detection are shown in Figure 12. Different scenarios are considered. The results show that the lane detection method as introduced in this paper works well under different situations. Note that detected lane boundaries are sometimes locally slightly curved. This is due to the fact that the distance transform of discontinuous lane marks is slightly unaligned in column direction in the bird's-eye image (see Figure 7).

The distance transform as used in this paper provides some other useful information besides a centerline (compared to that available in the edge map). The first example in Fig. 13 shows a case where the left lane (or road) boundary


(a) (b)

**Fig. 11.** (a) The test vehicle "High Awareness Kinematic Automobile 1" (HAKA1). (b) A stereo camera pair on a bar behind the windscreen.

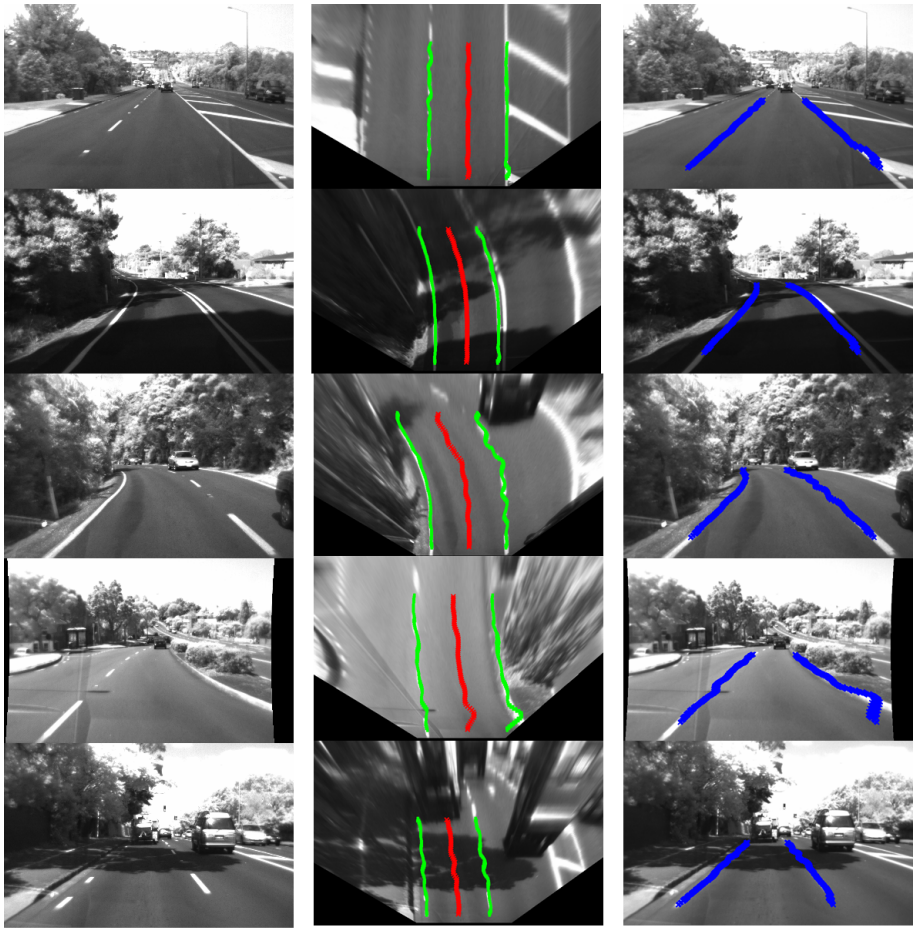


**Fig. 12.** Experimental results for lane detection. (a) Input images. (b) Lanes detected in the bird's-eye image. Note that red lines are the centerlines of a lane. (c) Lanes detected in input images.

is totally occluded by parked cars. However, there are many edges of cars, or of other objects along the left lane boundary in the bird's-eye image. With these edges, the RODT gives a distance map as if there are some real edges along this boundary, as well as large distance values at centerline points. A lane is finally detected reasonably using this distance information. The second example in Fig. 13 illustrates a similar difficulty for detecting edges of the left lane boundary. The experiment proves that lane boundaries can also be detected in this case in the RODT.

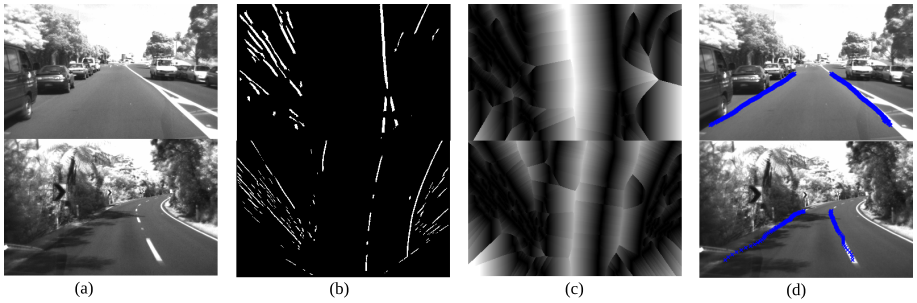


**Fig. 13.** Two examples illustrating the usefulness of RODT for the detection of lane boundaries. (a) The input image. (b) The edge map. Note that the edges in the left boundary are far from perfect.(c) RODT of (b). The left boundary (in black) and the centerline (in white) are clearly visible. (d) Lane detection result.

Some typical failures in lane detection are shown in Fig. 14, and these are due to missing lane information, strong noisy edges, difficult illumination conditions, and so on.

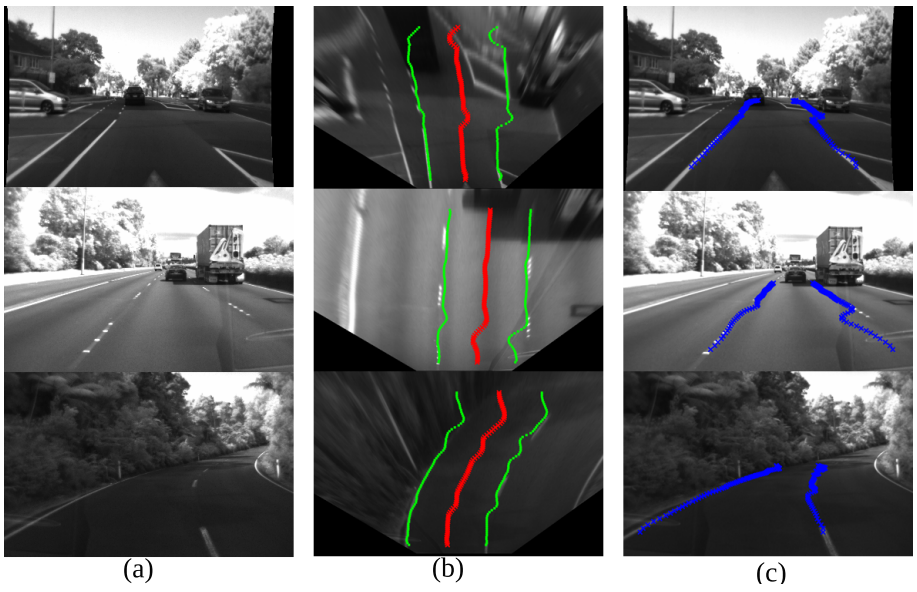


**Fig. 14.** Some failures of lane detection caused by missing information, noisy edges, as well as difficult lighting conditions. The top row shows the effect with large areas of no lane markings. The middle row shows the effect of a huge amount of noise from the internal reflection of the windscreen. The bottom shows the issues with a mixture of low contrast and large distances between lane marks.

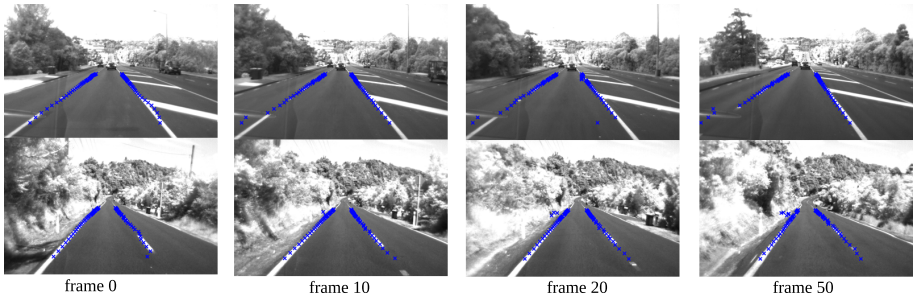


**Fig. 15.** Experimental results using efficient lane tracking.

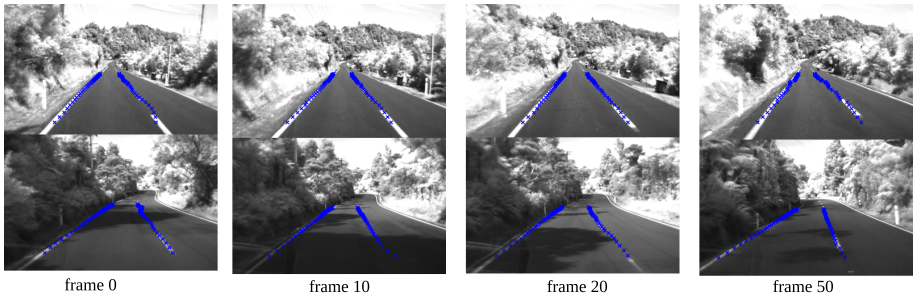


**Fig. 16.** Experimental results using robust lane tracking.

**Table 1.** Computation time of lane detection and tracking.

| Procedures | Lane detection | Efficient tracking | Robust tracking |
|---|---|---|---|
| time (s) | 0.144 | 0.137 | 0.148 |

Experimental results for periods while using the efficient lane tracking method are illustrated in Figure 15. The results are obviously acceptable, except for some sparse outliers at lane boundaries.

Experimental results for periods while using the robust lane tracking method are shown in Figure 16. Compared with efficient lane tracking, this method requires more computation time but proves to be more robust.

The computation time for lane detection and tracking is documented in Table 1 for a general comparison. The experiment was conducted using an off-the-shelf computer (Dual core, 2.1GHz) and OpenCV without any runtime optimization. Actually, the computation time will always be affected by several factors, such as $\Delta$, the forward-looking distance, number of particles, and so on.

Table 1 shows that efficient lane tracking performs relatively faster than the robust tracking approach and the lane detection.

The computation time for the various steps of low-level image processing is shown in Table 2. This table shows that low-level image processing consumes

**Table 2.** Computation time of steps in the low-level processing part.

| procedures | bird's-eye view | edge detection | noise remove | distance transform | total |
|---|---|---|---|---|---|
| time (s) | 0.015 | 0.015 | 0.015 | 0.075 | 0.120 |

most of the computation time of our lane detection and tracking method. Among those processing steps, the distance transform requires most of the computation time.

## 6 Conclusions

This paper introduced a new weak model of a lane, and a possible lane detection scheme using a particle filter. Furthermore, two lane tracking methods were proposed and discussed. They focus either on efficiency or on robustness, and can be applied under different scenarios.

A (simple and easy to calculate) distance transform was used in this paper for lane detection and tracking. It shows that the distance transform is a powerful method to exploit information in lane detection situations. The distance transform can deal with discontinuous lane marks, provides information for the detection of the border or centerline of a lane, finds initial values for the particle filter, and adjusts the tracking results conveniently.

**Acknowledgments**: This work is supported by the National Natural Science Foundation of China under Grant 50875169.

## References

1. R. Aufrére, R. Chapuis, and F. Chausse: A model-driven approach for real-time road recognition. *Machine Vision Applications*, **13**:95–107, 2001.
2. A. Broggi: Robust real-time lane and road detection in critical shadow conditions. In Proc. *Int. Symp. Computer Vision*, pages 353–358, 1995.
3. M. Bertozzi, A. Broggi: GOLD - A parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Trans. Image Processing*, **7**:62–81, 1998.
4. M. Bellino, Y. L. de Meneses, P. Ryser, and J. Jacot: Lane detection algorithm for an onboard camera. In Proc. *SPIE Photonics in the Automobile*, volume 5663, pages 102–111, 2005.
5. R. Danescu, S. Nedevschi, M. M. Meinecke, and T. B. To: Lane geometry estimation in urban environments using a stereovision system. In Proc. *Intelligent Transportation Systems Conf.*, pages 271–276, 2007.
6. E. D. Dickmanns and B. D. Mysliwetz: Recursive 3-D road and relative ego-state recognition. *IEEE Trans. Pattern Recognition Machine Intelligence*, **14**:199–213, 1992.
7. P. F. Felzenszwalb and D. P. Huttenlocher: Distance transform of sampled functions. Cornell Computing and Information Science, Technical Report TR2004-1963, September 2004.

8. J. D. Grisman and C. E. Thorpe: SCARF: A color vision system that tracks roads and intersections. *IEEE Trans. Robotics and Automation*, **9**:49–58, 1993.
9. M. Isard, and A. Blake: Condensation: conditional density propagation for visual tracking. *Int. J. Computer Vision*, **29**:5–28, 1998.
10. T. M. Jochem, D. A. Pomerleau, and C. E. Thorpe: MANIAC: A next generation neurally based autonomous road follower. In Proc. *Int. Conf. Intelligent Autonomous Systems*, 1993.
11. C. R. Jung and C. R. Kelber: A lane departure warning system based on a linear-parabolic lane model. In Proc. *IEEE Symp. Intelligent Vehicle*, pages 891–895, 2004.
12. Z. Kim: Realtime lane tracking of curved local road. In Proc. *IEEE Intelligent Transportation Systems*, pages 1149–1155, 2006.
13. Z. Kim: Robust lane detection and tracking in challenging scenarios. *IEEE Trans. Intelligent Transportation System*, **9**:16–26, 2008.
14. C. Kreucher and S. Lakshmanan: LANA: A lane extraction algorithm that uses frequency domain features. *IEEE Trans. Robotics and Automation*, **15**:343–350, 1999.
15. C. Kreucher, S. Lakshmanan: A frequency domain approach to lane detection in roadway images. In Proc. *Int. Conf. Image Processing*, pages 31–35, 1999.
16. D. Khosla: Accurate estimation of forward path geometry using two-clothoid road model. In Proc. *IEEE Symp. Intelligent Vehicles*, volume 1, pages 154–159, 2002.
17. C. Lipski, B. Scholz, K. Berger, C. Linz, and T. Stich: A fast and robust approach to lane marking detection and lane tracking. *IEEE Southwest Symp. Image Analysis and Interpretation*, pages 57–60, 2008.
18. A. M. Muad, A. Hussain, and S. A. Samad: Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system. In Proc. *IEEE TENCON*, pages 207–210, 2004.
19. M. A. Nasirudin and M. R. Arshad: A feature-based lane detection system using hough transform method. In Proc. *Int. Symp. Intelligent Transportation Systems*, pages 166–169, 2007.
20. D. Pomerleau: RALPH: Rapidly adapting lateral position handler. In Proc. *IEEE Symp. Intelligent Vehicles*, pages 506–511, 1995.
21. R. Klette, and A. Rosenfeld: *Digital Geometry.* Morgan Kaufmann, San Francisco, 2004.
22. S. Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake: Efficient lane detection and tracking in urban environments. In Proc. *European Conf. Mobile Robots*, pages 126–131, 2007.
23. Y. Wang, D. Shen, and E. K. Teoh: Lane detection using Catmull-Rom spline. In Proc. *IEEE Int. Conf. Intelligent Vehicles*, pages 51–57, 1998.
24. Y. Wang, E. K. Teoh, and D. Shen: Lane detection and tracking using B-Snake. *Image Vision Computing*, **22** : 269–280, 2004.
25. Y. Wang, L. Bai, M. Fairhurst: Robust road modeling and tracking using condensation. *IEEE Trans. Intelligent Transportation Systems*, **9**:570–579, 2008.
26. A. Wedel, U. Franke, H. Badino, and D. Cremers. B-spline modeling of road surfaces with an application to free space estimation. *IEEE Trans. ITS*, special issue for IV'08, 2008.
27. T. Wu, X. Q. Ding, S. J. Wang, and K. Q. Wang: Video object tracking using improved chamfer matching and condensation particle filter. In Proc. *SPIE-IS & T Electronic Imaging.* volume 6813, pages 04.1–04.10, 2008.

28. Y. Yagi, M. Brady, Y. Kawasaki, and M. Yachida: Active contour road model for smart vehicle. In Proc. *Int. Conf. Pattern Recognition*, volume 3, pages 3819–3822, 2000.
29. J. Zhang and H. Naqel: Texture-based segmentation of road images. In Proc. *IEEE Int. Conf. Intelligent Vehicles*, pages 260–265, 1994.
30. Y. Zhou, R. Xu, X. Hu, and Q. Ye: A robust lane detection and tracking method based on computer vision. *Measurement Science Technology*, **17**:736–745, 2006.