Machine Vision and Applications (2011) 22:721–737
DOI 10.1007/s00138-010-0307-7

ORIGINAL PAPER

Lane detection and tracking using a new lane model and distance transform

Jiang Ruyi · Klette Reinhard · Vaudrey Tobi · Wang Shigang

Received: 12 May 2009 / Accepted: 19 November 2010 / Published online: 19 January 2011

Abstract Lane detection is a significant component of driver assistance systems. Highway-based lane departure warning solutions are in the market since the mid-1990s. However, improving and generalizing vision-based lane detection remains to be a challenging task until recently. Among various lane detection methods developed, strong lane models, based on the global assumption of lane shape, have shown robustness in detection results, but are lack of flexibility to various shapes of lane. On the contrary, weak lane models will be adaptable to different shapes, as well as to maintain robustness. Using a typical weak lane model, particle filtering of lane boundary points has been proved to be a robust way to localize lanes. Positions of boundary points are directly used as the tracked states in the current research. This paper introduces a new weak lane model with this particle filter-based approach. This new model parameterizes the relationship between points of left and right lane boundaries, and can be used to detect all types of lanes. 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 an edge map, properties of this distance transform support improved lane detection, including a novel initialization and tracking method. This paper fully explains how the application of this distance transform greatly facilitates lane detection and tracking. 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, and comparisons with other methods.

Keywords Lane detection · Driver assistance · Particle filter · Euclidean distance transform

1 Introduction

Lane detection plays a significant role in driver assistance systems. Typically, lane detection is used for localizing lane boundaries in the given road images, and can help 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.

Lane detection and tracking have been widely studied for driving on a highway [7,17] or an urban road [22], for single [7,25] or multiple [2,18] lanes, with [4] or without [26] marks, based on region (texture [31] or color [9]) or edge [21] features. Various shape models have been applied to describe borders of a lane, such as *piecewise linear segments* [21], *clothoids* [7,17], *parabola* [12], *hyperbola* [27,16], *splines* [25,26], or *snakes* [26,30]. Several lane detectors have been implemented and named in literatures, such as GOLD [4], SCARF [9], RALPH [23], MANIAC [11], and LANA [15,16]. For more complete and detailed reviews of lane

Electronic supplementary material The online version of this article (doi:10.1007/s00138-010-0307-7) contains supplementary material, which is available to authorized users.

J. Ruyi · W. Shigang (✉)
The Department of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China
e-mail: wangshigang@sjtu.edu.cn

J. Ruyi
e-mail: jiangruyi@sjtu.edu.cn

K. Reinhard · V. Tobi
The Department of Computer Science, The University of Auckland, Auckland, New Zealand
e-mail: r.klette@auckland.ac.nz

V. Tobi
e-mail: tvau003@aucklanduni.ac.nz

models and lane detection methods, please refer to [7,4,19, 14], especially the work in [19], which provides an up-to-date table listing driving conditions, lane detector models and difficulties. 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, parallelity of 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, 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 marks, or unpaved roads).

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

This paper is organized as follows: Sect. 2 describes a new lane model. Lane detection using this new lane model, including low-level image processing as well as particle filter framework, is introduced in Sect. 3. Two lane tracking methods focusing on efficiency or robustness are discussed in Sect. 4. Experimental results and comparison of methods are given in Sect. 5. Finally, conclusions are provided in Sect. 6.

2 A new lane model

The lane model used in the paper is illustrated in Fig. 1; the 3D view also contains the xy-coordinate system in the *ground manifold*. In this paper we assume a *ground plane*.

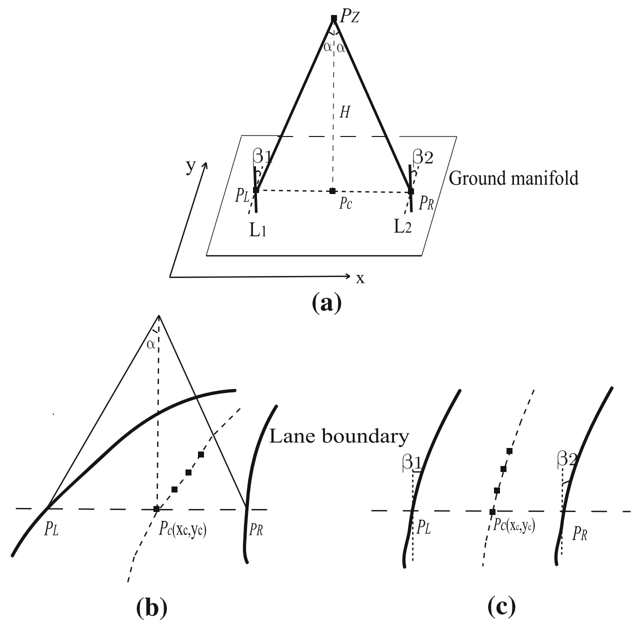


Fig. 1 Lane model as used in this paper. **a** 3D lane view; boundaries are drawn in *bold*. **b** Perspective 2D lane view in the input image. **c** Bird's-eye image of the lane. Slope angles β_1 and β_2 are shown in the 3D view and the bird's-eye image; the zenith angle α in the 3D and the projective view. See text for further explanations

(Wedel [28] proposes cubic B-splines for modeling the ground manifold.)

Five parameters x_c, y_c, α, β_1, and β_2 are used to model opposite points P_L and P_R, located on the left and right lane boundaries, respectively. $P_C = (x_c, y_c)$ is the *centerline point* of a lane in the ground plane. α is the *zenith angle* above P_C, defined by an upward straight line segment between P_C and the *zenith* P_Z 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 lane W at points P_L and P_R can be easily calculated as

$$W = 2H \cdot \tan(\alpha) \tag{1}$$

β_1 and β_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, β_1 and β_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. Note that as constraints of lane marks between left and right lane boundaries are fully parameterized and utilized, the model is applicable only to two-boundary lane detection situation.

This model does not use any assumption about global lane geometry, and applies to all kinds of lanes. For example, this

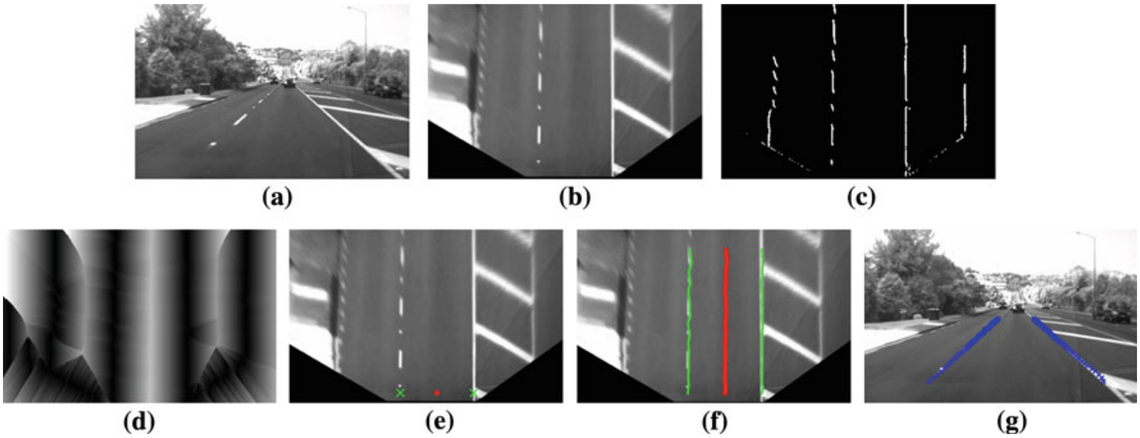


Fig. 2 The overall work flow of lane detection. **a** Input image. **b** Bird's-eye image. **c** Edge map. **d** Distance transform. **e** Initialization. **f** Lane detection results, shown in the bird's-eye image. **g** Lane detection results, shown in the input image

also allows us to detect a lane with varying width, even in a single image. As β_1 and β_2 are calculated separately, a lane may also have an unparallel left and right boundaries. Lane detection and tracking methods, using this model, are discussed in Sects. 3 and 4.

3 Lane detection using a particle filter

For lane detection using a weak model in a single image, particle filter is a good solution to track points along lane boundaries and has been tested by Sehestedt [22]; As simply coordinates of boundary points were used as the state vector to be tracked by Sehestedt [22], it will result into a countable state space, and model no internal relationship between boundary points. Below we show that our lane model provides a more preferable tracking state for particle filter, and models explicitly the internal relationship of left and right lane boundary points. 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 Fig. 2 by an example.

The algorithm starts with mapping the perspective input image into a bird's-eye view, using a homography defined by four vertices of a rectangle in the bird's-eye view. An edge detection method, as introduced in [4] 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 the isolated edge points or small blobs, a *row orientation distance transform* (RODT) is applied; see Sect. 3.1.3 for a specification of this transform. The resulting distance map allows us to design a novel initialization method for finding the initial boundary points. These points are used to initialize the parameters of the particle filter, for tracking further boundary points through the whole image; Also, the RODT map facilitates lane detection and tracking very much based on its advantageous properties. After smoothing boundary pairs using sliding mean, and mapping back into the input image, a lane is finally detected.

3.1 Low-level image processing

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

3.1.1 Bird's-eye mapping

As in [14], a four-point correspondence is used for the mapping from an input image into a bird's-eye image. The mapping is achieved by selecting four points when calibrating the camera, and 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 is proved to be useful for detecting discontinuous lane marks 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. See Fig. 3 for an example.

3.1.2 Edge detection and noise removal

We recall an edge detection method as introduced in [4]. Vertical edges with black–white–black pattern 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:

$$
\begin{aligned}
B_{+m}(x, y) &= b(x, y) - b(x+m, y) \\
B_{-m}(x, y) &= b(x, y) - b(x-m, y)
\end{aligned}
\tag{2}
$$

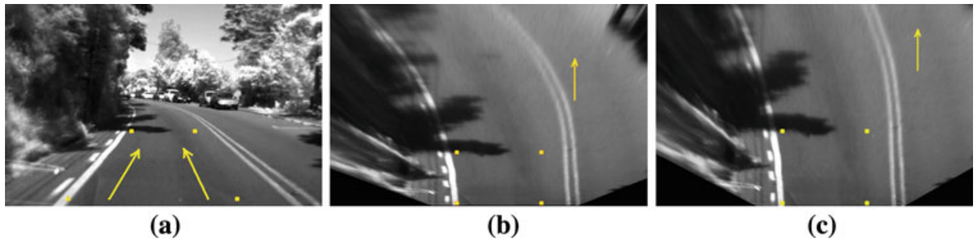


Fig. 3 **a** Input Image. **b** and **c** are bird's-eye images based on different distance definitions. Four-point correspondence (points shown in *yellow*) is established by calibration; the driving direction is indicated by the *arrows*

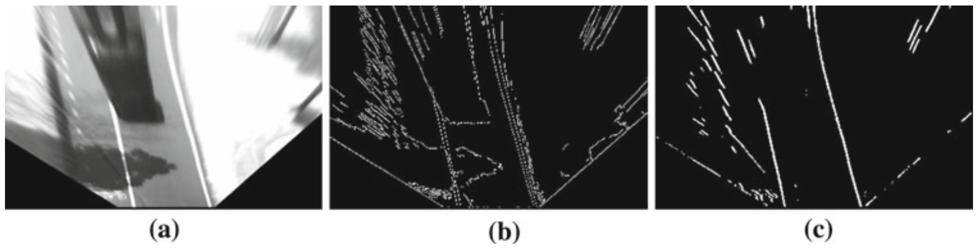


Fig. 4 Edge detection. **a** Bird's-eye image. **b** Edge detection using Canny operator. **c** Edge detection following [4]

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

$$r(x, y) = \begin{cases} 1, & \text{if } B_{+m}, B_{-m} > 0, B_{+m} + B_{-m} \geq T \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

This edge detection method has the following properties. First, m can be adjusted to fit various width 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 detection at lane marks. In the presence of shadows, the brightness of lane marks may be affected, but generally it will still maintain superiority relationship with their horizontal neighbors. Thus, the edge detection method can be used under different illumination conditions. Finally, horizontal edges (unlikely to be lane marks) are not detected. For an example of edge detection, see Fig. 4. In our experiment, we set $m = 5$, $T = 30$.

The edge detection as introduced above may generate some isolated small blobs (including single point) besides the edges of real lane marks. These noisy blobs will greatly affect the result of the following distance transform (see Fig. 6, distance transform will be discussed in Sect. 3.1.3). In order to remove such noise, a specified operation is applied. The general idea is first to find the isolated blobs, and then set them to zero (non-edge). Two small square windows (inner and outer) are used (see Fig. 5) with the same center. And the outer window is larger in width with a gap of 1 pixel than the inner window. The isolated blobs can be detected by moving at the same time these two windows through the whole edge map, and comparing the sums of edge values within them. Two equal sums means that the gap between two windows contains no edge points, and the edge blobs in the inner window are detected as isolated and set to zero. In our experiment, we set the width of outer window to 10 pixels, considering the minimum size of possible lane mark blobs. For computational efficiency, an integral image of the edge map is used to calculate the sums inside small windows.

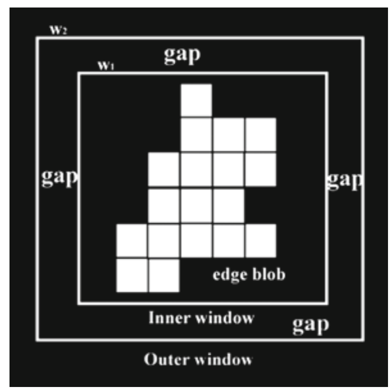


Fig. 5 Detection of the isolated blobs in the binarized edge map. The square inner window and outer window with a gap of 1 pixel ($w_2 - w_1 = 2$) move at the same time through the edge map. When the gap between the inner and outer window contains no edge, an isolated blob is detected in the inner window

3.1.3 Distance transform

Distance transform applied to a binary edge map I labels each pixel with distance to the nearest edge pixel. For all pixels p of I, distance transform determines

$$\mathrm{d}t(p) = min_u\{\mathrm{d}(p, q_u) : I(q_u) = 0 \wedge 1 \leq u \leq U\} \tag{4}$$

where $\mathrm{d}(p, q_u)$ denotes a metric, and u lists all U pixels in the image I. The values for $\mathrm{d}t(p)$ depend on the chosen metric. Edges are obviously labeled by value 0, and shown as black in the generated distance map.

Among various distance transform, Euclidean distance transform (EDT) is a common choice. The original EDT uses Euclidean metric for measuring the distance between

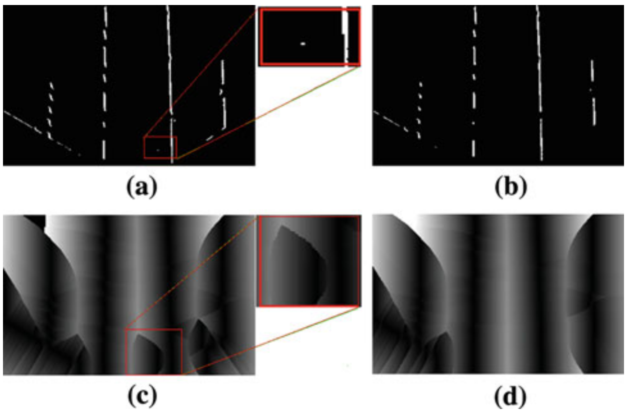


Fig. 6 Effect of isolated noisy point in edge map for distance transform. **a** Noisy edge map with an isolated edge point in the middle of lane. **b** Denoised edge map. **c** RODT of (**a**). (d) RODT of (**b**)

pixels. We substitute $\mathrm{d}(p, q_u)$ in Eq. 4 by the Euclidean metric $\mathrm{d}_e(p, q_u)$ as follows:

$$\mathrm{d}_e(p, q) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{5}$$

for points $p = (x_1, y_1)$ and $q = (x_2, y_2)$. For more details of distance transform and EDT, please refer to [24]. Felzenszwalb [8] proved that a 2D EDT can be calculated by two 1D EDTs, and this greatly improves the computational efficiency. A modified EDT was proposed in [29], called *Orientation Distance Transform* (ODT). This divides the Euclidean distance into two contributing components in row and column direction. (Note that the use of a four- or eight-distance transform would *not* lead to the same row and column components; however, practically there should not be a big difference with respect to the given context.) The *row* part of ODT, named RODT in this paper, labels each pixel with a distance value to the nearest edge point in row direction. Moreover, RODT 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.

The RODT of an edge map offers various benefits. Generally, as a distance transform, every pixel in RODT map indicates where is its nearest edge point. Thus, more information about a lane (e.g. information of the centerline or road boundary, to be indicated in Sect. 3) is provided by distance transform when compared with edge map. For example, a (non-edge) pixel on the centerline of a lane will have a local maximum in distance to lane boundaries. This information is of great usefulness to find or go towards edge points, and will be fully utilized in lane detection and tracking methods introduced in the following sections.

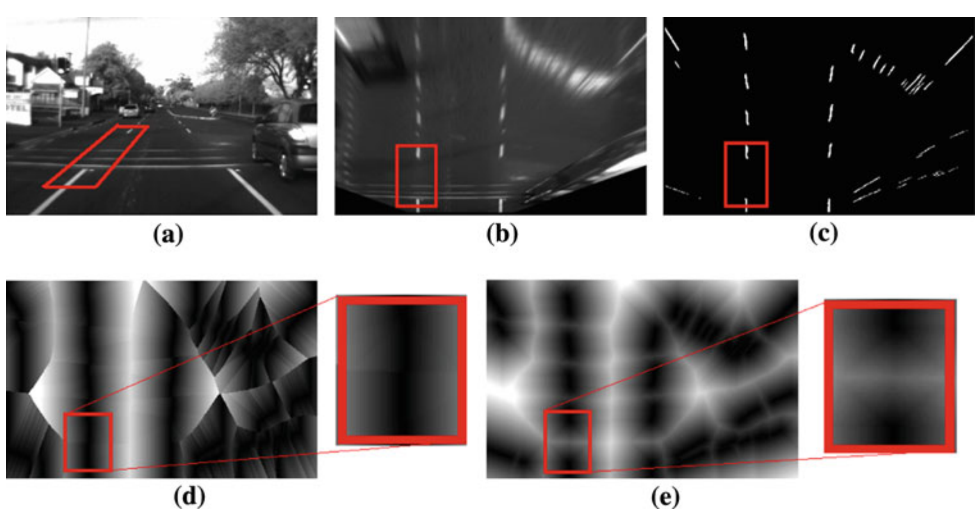
Moreover, compared with EDT, RODT brings several advantageous properties in lane detection situation. First, the initialization of lane detection becomes much easier (to be discussed in Sect. 3.2). Second, discontinuous lane marks will make almost no difference from continuous ones in RODT, but not true in EDT, as illustrated in Fig. 7. This also provides algorithms the ability to find the approaching lanes in advance.

Distance transform is sensitive to some isolated points or blobs in lane detection situation. As indicated in Fig. 6, an edge point in the middle of a lane will greatly change the distance value for the surrounding pixels. So a denoising method on the edge map as introduced in Sect. 3.1.2 is necessary, and proves to be useful.

3.2 Lane detection initialization

The aim of initialization is to find an initial value (e.g., the x-coordinate of points P_L and P_R in a selected image row) for the specified model. In [22], a clustered particle filter is used 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* y_{c_0} (near to

Fig. 7 EDT and RODT on discontinuous lane marks. **a** Road image. **b** Bird's-eye view. **c** Binarized edge map after noise removal. **d** RODT. **e** EDT. **d**, **e** have been contrast adjusted for better visibility. Note RODT shows no gap for the discontinuous lane marks inside the rectangle, while EDT still contains some gap



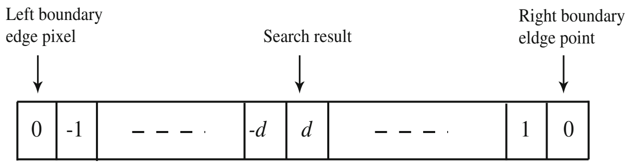


Fig. 8 Illustration of the search procedure in the start row of the distance map. Note that the distance values are signed, as described in Sect. 3.1.3

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 Fig. 8). When such a pixel is found, the left and right boundary points in the start row are instantly known using distance values of the found pixel and of its left neighbor. If there's some noisy edge points in the middle of a lane, which effects the distance value for searching the initial boundary points, then some useful assumptions can be used to complete this procedure. One of such assumptions can be the prior knowledge of the approximate width of a lane. Actually from experiment, we find that the detection result will converge to the correct lane boundary even with a moderately wrong initial position, due to particle filter as well as RODT.

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 α_0 are initialized using the detected left and right start points, while β_{1_0} and β_{2_0} are simply set to zero. Moreover, the initial width of lane W_0 can be easily calculated, and will be used in particle filter later.

3.3 Particle filter for lane detection

Particle filter is widely used for lane detection and tracking, such as in [22,27]. This section discusses particle filtering with 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 Δ (e.g. 10 pixels), starting at the *start row* in the bird's-eye image. For the application of a particle filter, two models [10] are discussed in the following.

3.3.1 The dynamic model

In the diffusion step of particle filtering, new particles are drawn from the conditional state density function from the former steps according to a dynamic model:

$$X_n^i \sim p(X_n | X_{n-1}^i, \omega_{n-1}^i) \tag{6}$$

In our situation, a linear dynamic model is available as:

$$X_n^i = A \cdot X_{n-1}^i + \mathbf{w}_n \tag{7}$$

where matrix $A_{4\times4}$ is the deterministic part of the dynamic model, and standard normal vector $\mathbf{w}_n$ is the stochastic component. We simply take A as being the identity matrix, because of the assumed smoothness of lane boundaries.

3.3.2 The observation model

In the measurement step, each particle is assigned a weight according to some measured features $\mathbf{z}$ in the observation model:

$$\omega_n^i = p(\mathbf{z}_n | X_n = X_n^i) \tag{8}$$

In our situation, the observation model incorporates information from RODT and smoothness to gain the robustness of the algorithm.

Based on RODT information, it is reasonable to assume that points on lane boundaries will have small distance values, while those on the centerline of a lane will have large ones. In terms of our lane model, points (x_{c_n}, y_{c_n}) will have large distance values, and L_1 and L_2 will coincide with short lines of pixels which all only have small distance values (The length of L_1 and L_2 are 20 pixels in our experiment).

Tracking step n is identified by $y_{c_n} = (y_{c_0} + n \cdot \Delta)$, $n = 0, 1, \ldots, N$. Here, N is determined by the forward-looking distance, and is 40 in our experiment after calibration. We calculate the lateral position of the left boundary points of the lane from the particle filter, with $X_n^i(x_{c_n}^i, \alpha_n^i, \beta_{1_n}^i, \beta_{2_n}^i)$ for the ith particle.

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

$$P_L^i = x_{c_n}^i - H \cdot \tan \alpha_n^i \tag{9}$$

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

$$S_{L_1}^i = \sum_{j=-L_1/2}^{L_1/2} \left| d\left(P_L^i + j \cdot \sin \beta_{1_n}^i,\ y_{c_n} + j \cdot \cos \beta_{2_n}^i\right)\right| \tag{10}$$

Here, $d(\cdot, \cdot)$ is the signed distance value from RODT. $S_{L_2}^i$ can be calculated in an analogous way. And the distance value for the centerline point $(x_{c_n}^i, y_{c_n})$ is simply $d(x_{c_n}^i, y_{c_n})$. Finally, for the observation model from RODT, we can calculate from $S_{L_1}^i$,$S_{L_2}^i$ and $d(x_{c_n}^i, y_{c_n})$ separately. But from experiment, it is found that more robustness can be achieved using the following model, as sometimes lack of lane marks or noisy edge points in the middle of a lane can make the distance value all very large or small:

$$\omega_{rodt}^i = b_1 \exp\left(-c_1 \cdot \frac{S_{L_1}^i + S_{L_2}^i}{|d(x_{c_n}^i, y_{c_n})|}\right) \tag{11}$$

where constants $b_1, c_1 > 0$, and are determined by the importance ratio between ω^i_{rodt} and ω^i_{smooth} (see Eq. 12). In our experiment, they are set to $b_1 = 1, c_1 = 5$.

Additionally, the smoothness of lane width and angle parameters from the lane model is used to improve robustness. Assuming Gaussian distribution of lane width during lane detection in one image, with mean W_0 (from initialization) and standard deviation aW_0 ($a = 0.1$ in our experiment), and Gaussian distribution of the tracked angles $(\alpha_n, \beta_{1_n}, \beta_{2_n})$ of lane model, with mean $(\alpha_{n-1}, \beta_{1_{n-1}}, \beta_{2_{n-1}})$, and a predefined same standard deviation (σ, σ, σ) ($\sigma = 0.1$ in our experiment), we have

$$\omega^i_{smooth} = \frac{1}{aW_0\sqrt{2\pi}} \exp\left(-\frac{(W^i - W_0)^2}{2a^2W_0^2}\right) \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\alpha^i_n - \alpha_{n-1})^2}{2\sigma^2}\right) \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\beta^i_{1_n} - \beta_{1_{n-1}})^2}{2\sigma^2}\right) \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(\beta^i_{2_n} - \beta_{2_{n-1}})^2}{2\sigma^2}\right) \tag{12}$$

The final observation model is given by the factors

$$\omega^i_n = \omega^i_{rodt} \cdot \omega^i_{smooth} \tag{13}$$

With the models as introduced above, the complete particle filter adopted to detect lane marks inside a frame is shown in Algorithm 1.

1 **States:** $X_n = \{x_{cn}, \alpha_n, \beta_{1n}, \beta_{2n}\}$, $n = 0, 1, \ldots, N$;
2 **Initialization:** Randomly generate M particles $\{X^i_0, i = 1, \ldots, M\}$, with X_0 determined as Sect. 3.2;
3 **for** $n = 1 : N$ **do**
4 -Diffusion: draw $X^i_n \sim p(X_n | X^i_{n-1}, \omega^i_{n-1})$ according to the dynamic model in Eq. 7;
5 -Measurement: assign each particle a weight ω^i_n according to the observation model by Eq. 11,12,13;
6 -Particle with the largest weight is selected:

$$X_n = arg \max_{X^i_n} \{\omega^i_n, i = 1, \ldots, M\}$$

7 **end**

Algorithm 1: Particle filtering of lane marks in a frame

3.4 Post-processing

While using a weak lane model, we don't have a global shape for the detected lane, but pairs of left and right lane boundary points from the tracked state. A sliding mean is applied to smooth the detected boundary points as

$$P'_{L_n} = P'_{L_{n-1}} + T((P_{L_n} - P_{L_{n-s}})/s, d_0) \tag{14}$$

where function $T(d, d_0)$ is

$$T(d, d_0) = \begin{cases} d_0 & \text{if } |d| > d_0 \\ d & \text{if } |d| \le d_0 \end{cases} \tag{15}$$

We use values $s = 2$, $d_0 = 10$ in our experiment.

Projection of the smoothed point pairs from the bird's-eye view to the input image, a two-boundary lane is detected.

4 Lane tracking

Lane tracking uses information from the previous results to facilitate the current detection. Generally, there are two aims to utilize the previous information: one is to improve computational efficiency of the current detection from a priori knowledge; another is for robustness, as there is more information available by combining the current with the previous. 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 as compared to others (such as on an urban road), depending on road conditions and quality of lane marks. Difficulties as discussed in Sect. 1 mainly happen when detecting a lane in some challenging situation. In conclusion, when performing lane tracking, we will pay more attention to computational efficiency for less challenging situations, but emphasis more on 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

Efficient lane tracking method is fit to situations characterized by good road conditions and good quality of lane marks (such as driving on a highway). It simply uses previously detected lane boundary points, adjusts them by the ego-vehicle's motion model, and then refines them according to the distance values from RODT on the current bird's-eye edge map. It should be noted that neither kalman filter nor particle filter is used for the tracking purpose.

When a lane is detected as in Sect. 3, it is reasonably represented 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 (or similarly mapped to the original image).

Tracking a lane through an image sequence is then simplified as tracking these two point sequences. Sequences $\{P^{(t)}_{L_n}\}$ and $\{P^{(t)}_{R_n}\}$, detected in frame t, are already partly 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. The detection process of $\{P^{(t+1)}_{L_n}\}$ and $\{P^{(t+1)}_{R_n}\}$ at

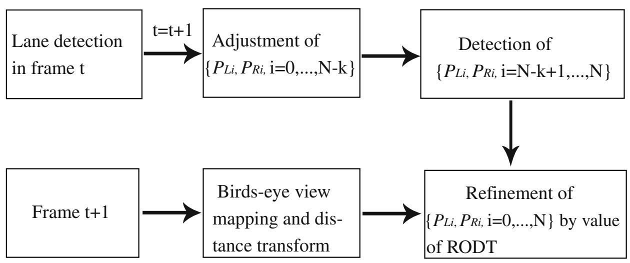


Fig. 9 Efficient lane tracking scheme

time $t+1$ is composed of three steps: adjustment caused by the driven distance and the variation in yaw angle, new points detection, and refinement specification according to the values of RODT in the bird's-eye edge map.

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

$$P_{L_n}^{(t+1)} = P_{L_{(n+k)}}^{(t)}, \qquad P_{R_n}^{(t+1)} = P_{R_{(n+k)}}^{(t)} \tag{16}$$

Here, $n = 0, 1, \ldots, N-k$, and k is determined by the driven distance between time t and $t+1$, and is usually a small number. Information from various navigation sensors (e.g., odometry, GPS, speedometer and so on) can be applicable to obtain k. In practice, if no sensor data is available at all, then we can simply set k to 1 or 2, as further steps in the tracking will adjust the final results. Furthermore, points

$$\{P_{L_n}^{(t+1)}, P_{R_n}^{(t+1)} : n = 0, 1, \ldots, N-k\} \tag{17}$$

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

$$P_{L_n}^{(t+1)} = P_{L_{n-1}}^{(t+1)}, \qquad P_{R_n}^{(t+1)} = P_{R_{n-1}}^{(t+1)} \tag{18}$$

For further refinement, those predictions $\{P_{L_n}^{(t+1)}\}$ and $\{P_{R_n}^{(t+1)}\}$ from the previous results 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

$$\begin{aligned} P_{L_n}^{(t+1)} &= P_{L_n}^{(t+1)} + d(P_{L_n}^{(t+1)}, y_{c_n}), \quad n = 0, 1, \ldots, N \\ P_{R_n}^{(t+1)} &= P_{R_n}^{(t+1)} + d(P_{R_n}^{(t+1)}, y_{c_n}), \quad n = 0, 1, \ldots, N \end{aligned} \tag{19}$$

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

The described efficient lane tracking scheme is summarized in Fig. 9. The main feature of this method is its great computational efficiency. Experiments proved that the computation time needed for those three steps of efficient lane tracking is almost ignorable. 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, always caused by some noisy non-boundary edge points (see experimental results as shown in Fig. 13).

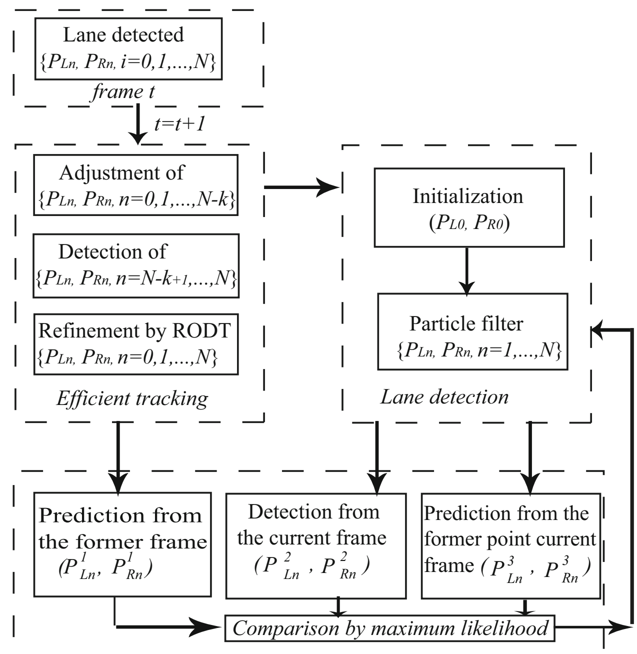


Fig. 10 Robust lane tracking scheme

4.2 Robust lane tracking

Urban roads differ from highways by an increased complexity of environments possibly of relevance for accurate lane detection. In such situations, robustness is of dominant importance. A scheme for robust lane tracking is illustrated in Fig. 10. This process is also composed of three main steps: first, a similar process as what we do in efficient lane tracking; The only difference is that instead of using Eq. 19, we use the following:

$$\begin{aligned} P_{L_n}^{(t+1)} &= P_{L_n}^{(t+1)} + T(d(P_{L_n}^{(t+1)}, y_{c_n}), d_0), \quad n = 0, 1, \ldots, N \\ P_{R_n}^{(t+1)} &= P_{R_n}^{(t+1)} + T(d(P_{R_n}^{(t+1)}, y_{c_n}), d_0), \quad n = 0, 1, \ldots, N \end{aligned} \tag{20}$$

where function $T(d, d_0)$ is as Eq. 15. This modification prevents the boundary points from diverging caused by imperfect road conditions. Second, lane detection as described in Sect. 3 with initialization from the first step. Also, a different W_0 is used, which is the mean width of the lane detected at frame t. Third, comparison of three possible pairs of lane boundary points using maximum likelihood. Note that the first step will be done once for all for time $t+1$ from t, but the last two steps will do iteratively in lane detection for every tracking step n in particle filter frame. As the first two steps have already been discussed in Sect. 4.1, as well as lane detection already in Sect. 3, only the third step, the comparison procedure, remains to be described in this section.

As a result of the first step, 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 point sequences for frame t). We group them into pairs

$$\left(P^1_{L_n}, P^1_{R_n}\right) \tag{21}$$

for the following process. In lane detection, when a pair of points is detected by the particle filter on the left and right boundaries at the nth tracking step, we index this pair as

$$\left(P^2_{L_n}, P^2_{R_n}\right) \tag{22}$$

Finally, the third pair

$$\left(P^3_{L_n}, P^3_{R_n}\right) \tag{23}$$

consists of the point pair 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 from particle filter in the current frame. The third pair $(P^3_{L_n}, P^3_{R_n})$ is a smooth prediction in the current frame from the result of $(n-1)$th step. 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 $p(\mathbf{z}|P^k_{L_n}, P^k_{R_n})$, with $\mathbf{z}$ 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:

$$P^* = arg \max_{k=1,2,3} p(\mathbf{z}|P^k_{L_n}, P^k_{R_n}) \tag{24}$$

The determination of the likelihood function uses information about the similarity of the width p_{width} of the lane, as well as values p_{rodt} of the distance transform.

Similarly as in lane detection, the width of lane for frame $t+1$ follows Gaussian distribution with mean W (the mean width of lane in frame t) and variance aW (a takes the same value as in lane detection), then we have $(k = 1, 2, 3)$

$$p^k_{width} = \frac{1}{aW\sqrt{2\pi}} \exp\left(-\frac{(P^k_{R_n} - P^k_{L_n} - W)^2}{2a^2W^2}\right) \tag{25}$$

Also, the values of the RODT at the points of those pairs are used to evaluate the possibility of finding lane boundaries as $(k = 1, 2, 3)$:

$$p^k_{rodt} = b_2 \exp(-c_2 \cdot (d(P^k_{R_n}) + d(P^k_{L_n}))^2) \tag{26}$$

where constants $b_2, c_2 > 0$, and are determined by the importance ratio between p_{width} and p_{rodt}. We set them to $b_2 = 1, c_2 = 0.001$ in our experiment.

The final value of the likelihood function is calculated by

$$p(\mathbf{z}|P^k_{L_n}, P^k_{R_n}) = p^k_{width} \cdot p^k_{rodt}, \qquad k = 1, 2, 3 \tag{27}$$

The comparison of $p(\mathbf{z}|P^1_{L_n}, P^1_{R_n})$, $p(\mathbf{z}|P^2_{L_n}, P^2_{R_n})$, and $p(\mathbf{z}|P^3_{L_n}, P^3_{R_n})$ will select one pair with the largest likelihood value as being the final detection result at the nth step of the lane detection procedure.

5 Experiments

Experiments were conducted on images and sequences recorded with our test vehicle "HAKA1" of the *.enped-a..* project, as well as on Caltech lane dataset from [1] for comparison. The camera in "HAKA1" is installed behind the windscreen, and pointing forward. So the optic axis is parallel with the ground plane. Images sampled from the camera are in gray scale, with a size of 640*480. As we don't need the internal and external parameters of the camera for lane detection, the only thing to do in calibration is to find the homograph matrix for bird's-eye mapping as introduced in Sect. 3.1.1. As the distortion caused by lens is almost ignorable, no rectification of images is needed in our experiment.

When running the algorithm on the testing sequences, we uses several prior knowledge of lanes. First, we confine the width of lanes to be within some range in bird's-eye view. In this experiment, the range is 200–400 pixels, which is very wide. Also, we assume that the left boundary of lane will be in the left part of the bird's-eye image, while the right boundary in the right part. This simple assumption is used to deal with lane-crossing situation. When situations as out of range in width or wrong place of lane boundaries are happened in lane tracking, a simple re-initialization of lane detection is applied for the following frames. Second, there are many situations with no or very less lane marks on road, where lane detection is meaningless. Mean distance value of the lane boundary pairs in bird's-eye view is calculated to illustrate the quality of the lane detected. The value above a threshold (here 30) triggers a re-initialization as well.

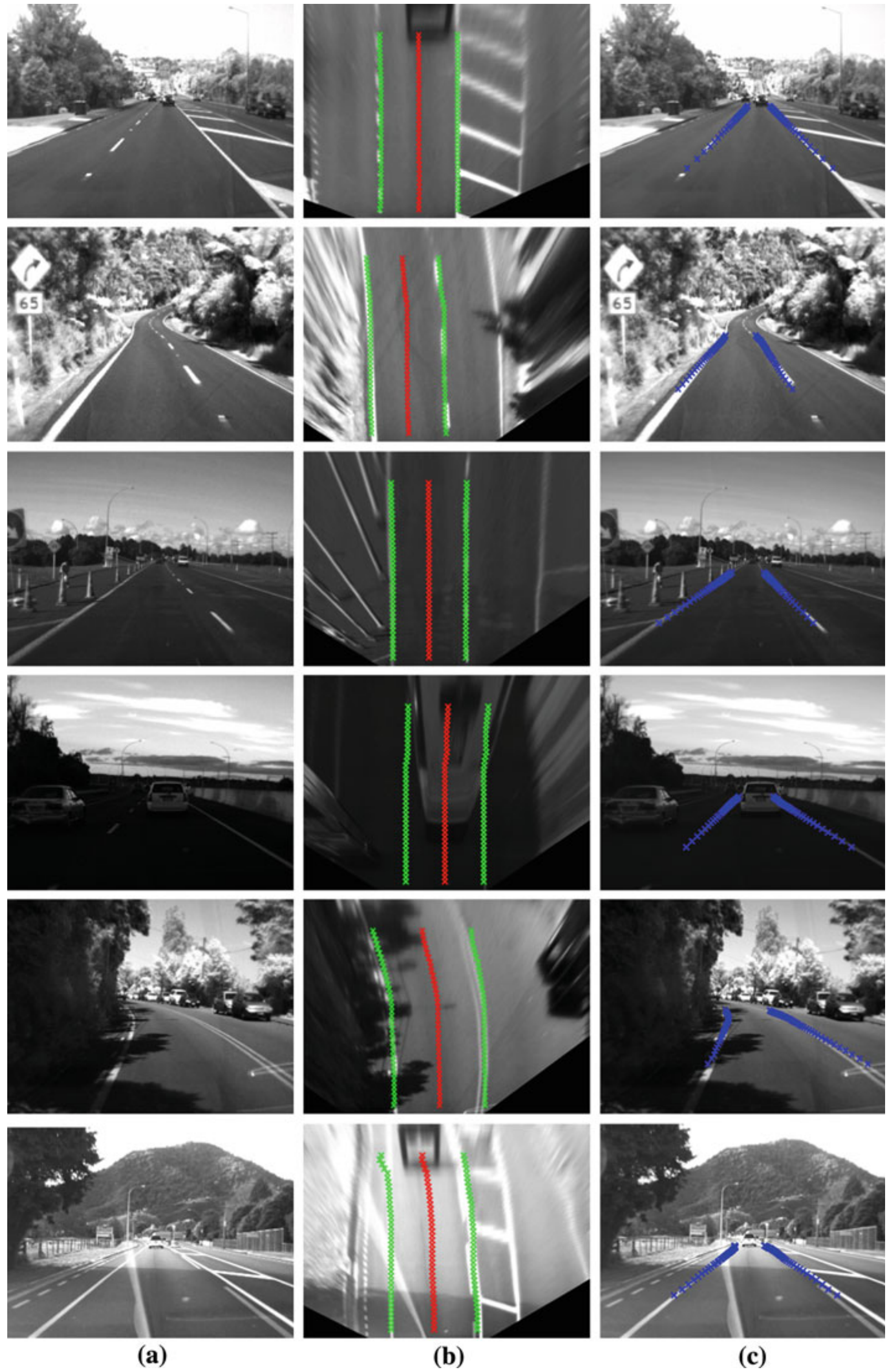


Fig. 11 Experimental results for lane detection. **a** Input images. **b** Lanes detected in the bird's-eye image. Note that the middle (*red*) lines are the centerlines of a lane. **c** Lanes detected in input images

Finally, we take robust lane tracking as the default lane detection method for a sequence. Also, we provide experiment results of using efficient lane tracking on highways. Note that the constants used in the following experiment are assigned the same values, as stated in the context once the constants are introduced.

5.1 General experiment

Six sequences (150 frames each) from both highway and urban roads are used to test the vitality of our lane detection method. Different scenarios are considered. Frames from every sequence are shown in Fig. 11 for a general idea of

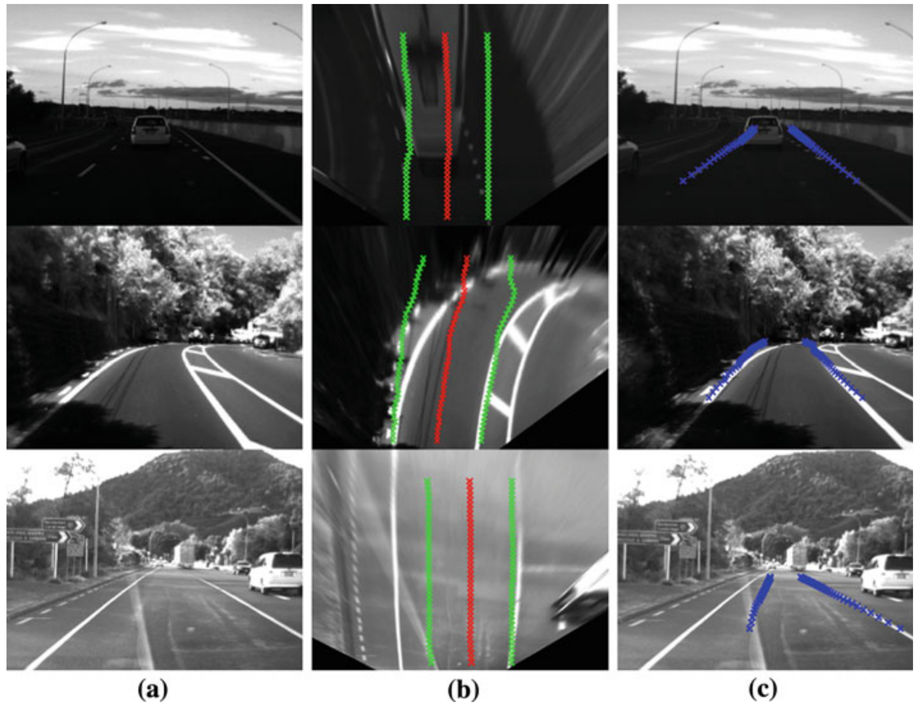


Fig. 12 Three typical failures of lane detection caused by misleading lane marks (*first row*), noisy edges (*second row*), as well as wrong initialization (*third row*)

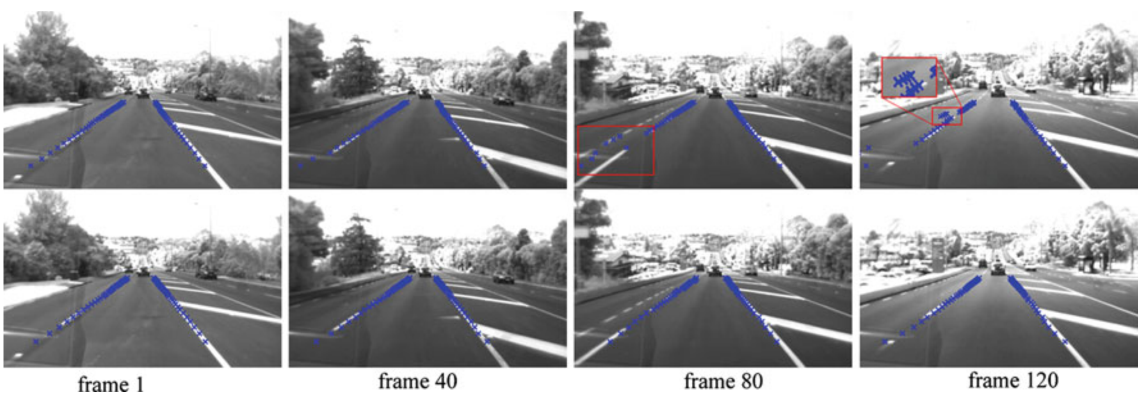


Fig. 13 Experimental results using efficient (*the first row*) and robust (*the second row*) lane tracking on a sequence with high-way like (relatively good) road situation. Note that results by efficient method have outliers (marked by a *rectangle*) sometimes, while almost perfect by robust method

road conditions. Typically, the first four sequences are on highways, but under various environment and driving conditions. The vehicle vibrates up and down (changes in pitch angle) in the second sequence, which causes the width of lane in bird's-eye view changes a lot. In the third sequence, the vehicle drives over a construction site; In the fourth sequence, the front view is blocked by a car. The last two sequences are on challenging urban environments with curved and straight road (more urban road situations will be shown in Caltech lane dataset later). Among all these sequences, there are various illumination conditions, bright or dark, as well as large area of shadows.

The detection results for the first three sequences are almost perfect, showing the effectiveness of the algorithm on relatively good road conditions. While for the fourth sequence, a leading car blocks part of the lane to be detected, which attracts the lane boundaries towards the leading car. Also a misleading design of lane marks causes several wrong detections as shown in the first row of Fig. 12. However, our assumption of lane width corrects this misdetection several frames later. The failures of detection in the last two sequences (see Fig. 12) are mainly caused by complex environments.

Experimental results using efficient lane tracking method are illustrated in Fig. 13 with comparison to the results by robust lane tracking. The efficient lane tracking works quite acceptable in good road situation except for several outliers sometimes.

The particle filter used here proves to be efficient for lane boundary detection. The effect of particle number with respect to the sum of absolute error and computation time is provided in Fig. 14. It is obvious that the more particles

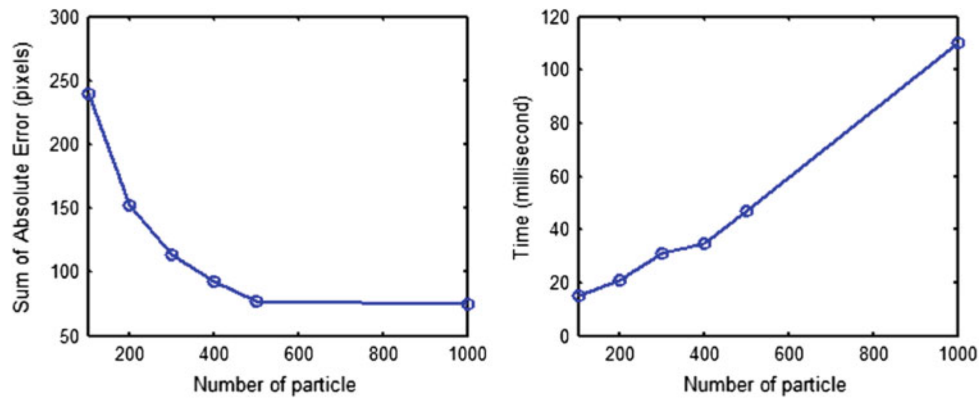


Fig. 14 The sum of absolute error (in pixels) and calculation time for particle filters with different particle number

Table 1 Computation time of lane detection and tracking

Procedures	Lane detection	Efficient tracking	Robust tracking
Time (s)	0.062	0.035	0.063

Table 2 Computation time of steps in low-level processing

Steps	Bird's-eye view	Edge detection	Noise removal	RODT	Total
time (s)	<0.001	<0.001	0.015	0.016	0.031

used, the more accurate and smooth results obtained. For all the sequences, we set the particle number 300. The smoothing using sliding mean as a post-processing also contributes to the final smoothness of the lane detected.

The computation time for lane detection and tracking, as well as low-level image processing on an off-the-shelf PC (Intel dual-core E5300, 2.5 Ghz), implemented in C with OpenCV, are provided in Tables 1 and 2. It can be seen that noise removal and RODT take the most of time in low-level image processing, while lane detection only and robust lane tracking consumes similar time. And not surprising, efficient lane tracking uses the minimum computation time. Generally, lane detection using our method is capable to reach real-time (e.g. over 10 fps) with careful implementation or even special hardware.

5.2 Comparison with other methods

In order to compare our method with others, we use a public testing dataset collected by Aly [1]. The dataset contains four color clips of challenging urban road images captured by a camera installed on a vehicle. Totally 1224 frames are considered, with the number of frames for each clip shown in Table 3. Different road curvatures, lighting conditions and traffic situations are considered in the clips. Along with the dataset, Aly also provides a MATLAB tool for manually labeling lane marks as ground truth, as well as lane detection results using method described in [1].

Three algorithms are considered in performance on the dataset. **Method 1** is the algorithm described by Aly [1]; **Method 2** is what is discussed in this paper; and **Method 3** is from Sehestedt [22]. The selection of **Method 1** is because it is in the category of lane detection using strong model. While **Methods 2** and **3** are using weak models. Meanwhile, **Methods 1** and **3** utilize color information, but **Method 2** adopts a color-to-gray conversion. Though two types of lane detection, two-boundary lane and multi-boundary lane are both discussed in [1], we use only two-boundary lane detection results in **Method 1** for comparison. Meanwhile **Method 3** is also confined to two-boundary lane detection, though clustered particle filter adopted makes it capable for multiple lane detection. Moreover, it should be noted that the detection results are independent splines in **Method 1**, independent left and right point sequences in **Method 3**, but pairs of left and right lane boundaries in our method. Finally, we adopt the same standard as [1] for a good detection.

Table 3 Detection analysis on Caltech lane dataset

Clip	Frames	Detection rate			False positive	False negative	
		Method 1 (%)	**Method 2** (%)	**Method 3** (%)	**Method 1** (%)	**Method 2** (%)	**Method 3** (%)
1	250	97.2	97.4	97.8	3.0	1.3	1.7
2	406	96.2	91.1	89.4	38.4	5.7	8.1
3	336	96.7	97.8	92.2	4.7	1.2	7.0
4	232	95.1	97.3	96.2	2.2	1.4	2.9

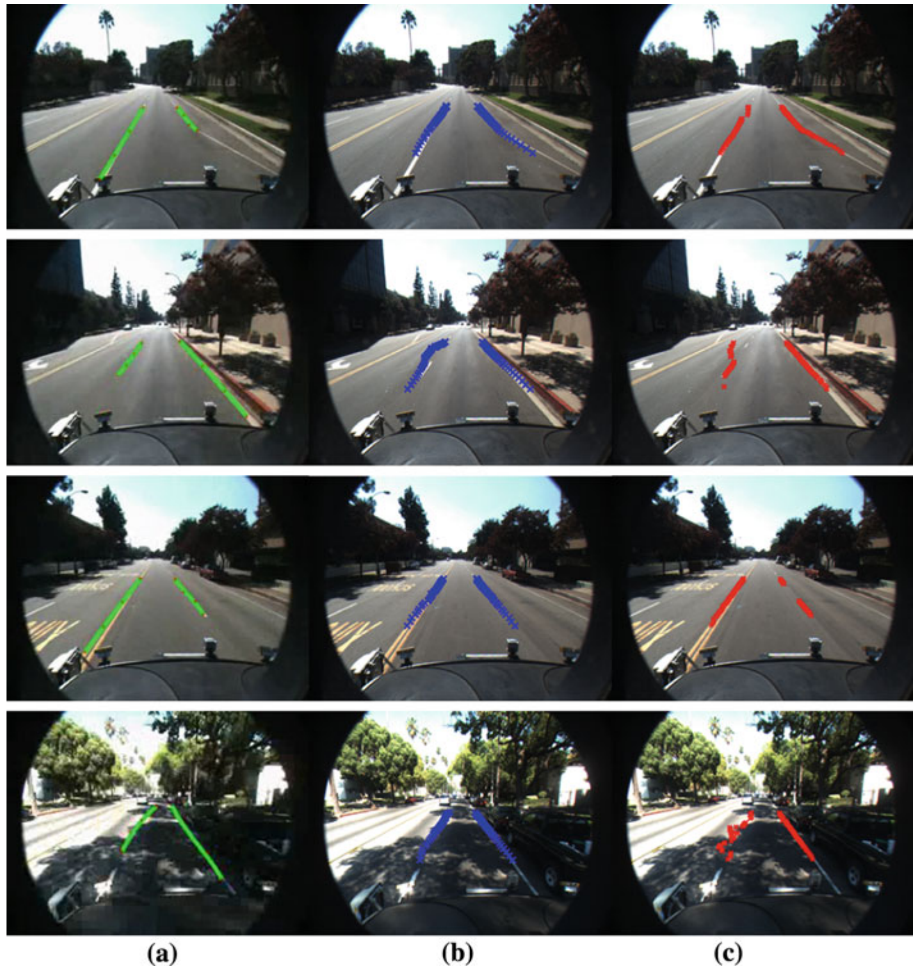


Fig. 15 Lane detection results from three algorithms. For detailed description, see the context. **a** Lane detection by **Method 1** (Aly [1]). **b** Lane detection by **Method 2** (this paper). **c** Lane detection by **Method 3** (Sehestedt [22])

Detection results on four typical frames from the dataset are presented in Fig. 15 to illustrate the different performance of the algorithms and reveal their intrinsic properties in detecting lanes. In the first frame, the right lane boundary is with a shape that cannot be modeled as a spline, which prevents **Method 1** from detecting it completely with strong spline model. When using weak models and detecting points by points, **Methods 2** and **3** have no problem for any shape of lane boundaries. In the second frame, there are many distracting noise edges around the left lane boundary, and **Methods 2** and **3** both are distracted somehow, as no global model is available to correct this distraction. The third frame shows the benefit brought by RODT as introduced in this paper, considering the discontinuous lane marks in the right boundary. Though edges are not available in the gap, global model as adopted in **Method 1** helps to connect two lane mark parts together, while weak-model-based **Method 3** cannot. With RODT, **Method 2** not only detects discontinuous lane marks coherently, but also reasonably extends detected lane marks to form complete lane boundary. In the fourth frame, **Method 2** shows its robustness over **Method 3** on some extremely noisy environment. Though both methods are using weak models, **Method 2** is more robust to the effect of noisy edges, especially in noisy shadow area. Note that **Method 2** is strictly a two-boundary algorithm, while **Method 1** and **3** are generally multi-boundary (including one-boundary) algorithms. With these difference, they will behave differently in large curvatures with two or one boundaries visible, as shown in Fig. 16.

The quantitative analysis of the three methods is shown in Table 3. In the table, “false positive rate” is calculated for **Method 1**, as it uses line detection as a pre-processing to detect all possible lane marks. While **Methods 2** and **3** replace it with “false negative rate”, because our weak lane models aim to find the existing left and right lane boundaries in front of the vehicle. High detection rate with high false positive rate, as **Method 1** performs on Clip 2, means that detection result contains correct lane marks as well as many other outliers. The low detection rate for **Method 3** on Clip 2 is due to extremely noisy road conditions with complex shadows at the beginning of the clip. From the table, **Method 2** is comparable with **Method 1** based on the overall performance, even a bit better. And **Method 2** is more robust than **Method 3** for dealing with difficult road situations.

Fig. 16 **Method 2** in large curvature situation (Frames are from Zu [14]). **a** Large curvature with two lane boundaries available, **Method 2** works, as well as will **Methods 1** and **3**. **b** Large curvature with only one lane boundary visible, **Method 2** will fail, but **Method 1** and **3** will still work, as they are generally a multi-boundary algorithm

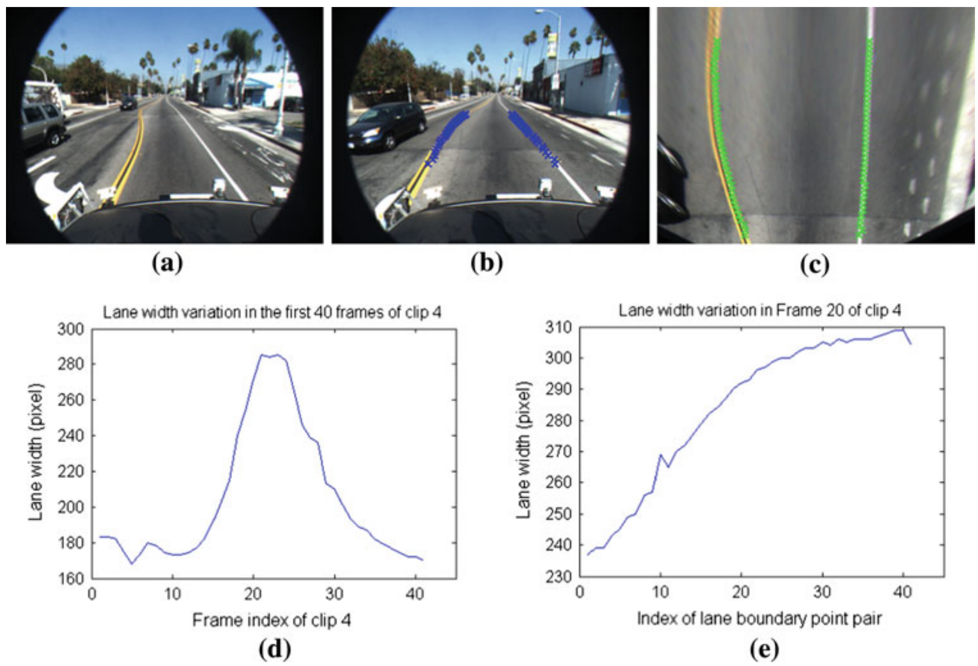


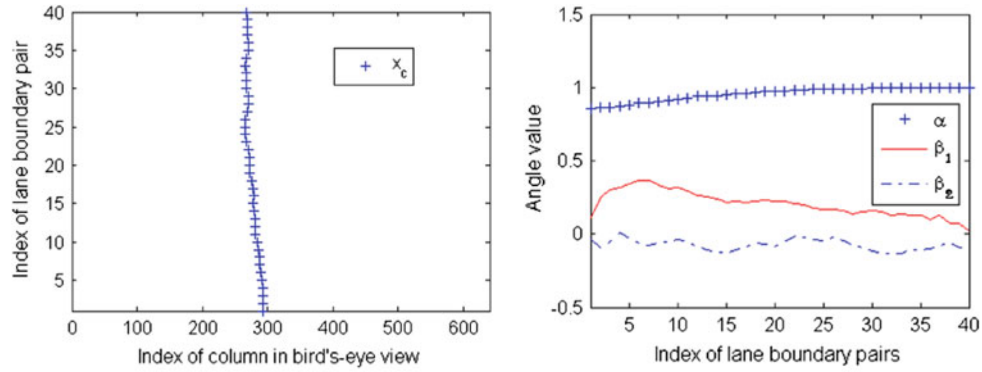
Fig. 17 Effectiveness of the new lane model to the variation of lane width in a single image and sequence. **a** Frame 15 in clip 4 to illustrate the lane ahead with variational width. **b** Frame 20 in clip 4 with lane detected using robust lane tracking. **c** Bird's-eye view of frame 20 with lane detected. **d** Lane width variation in the first 40 frames of clip 4. Note the width for frame is the mean width of lane boundary pairs calculated in bird's-eye view. **e** Lane width variation in bird's-eye view of Frame 20

5.3 Width variation in the new lane model

As stated in Sect. 2, the new lane model introduced in this paper can be used to deal with a lane with variational width, as well as unparallel lane boundaries. A typical example from clip 4 of the Caltech data set is used to illustrate the property of this lane model. There's a lane with changing width in the first 40 frames of the clip 4 as shown in Fig. 17(a). The variation of lane width among these 40 frames as detected using robust lane tracking is shown in Fig. 17(d). Meanwhile, the variation of lane width within frame 20 (Fig. 17 b, c)is shown Fig. 17(e). Furthermore, the specific state values of particle filter for frame 20 are also provided in Fig. 18, which clearly demonstrates the different values for angle β_1 and β_2, as well as the changing of x_c and α. This proves the capability of the new lane model in dealing with lanes with variational width and unparallel boundaries.

5.4 RODT for lane without lane marks

Distance transform used in this paper provides some other useful information besides centerline compared with edge map. In the first example as shown in Fig. 19, left lane (or road) boundary is totally occluded by parked cars. However, there are many edges from cars or any other objects along left lane boundary in the bird's-eye image. With these edges, RODT gives a distance map as if there are some real edges along this boundary, as well as large distance value in the centerline points. A lane is finally detected reasonably using distance information. The second example in Fig. 19



(a) **(b)**

Fig. 18 State values tracked by particle filter for frame 20 as in Fig. 17. **a** Values of x_c tracked. **b** Values of α, β_1 and β_2 tracked. Note that α is always positive, but β_1 and β_2 are signed, with positive as lane boundary turns to *left*, and negative to *right*

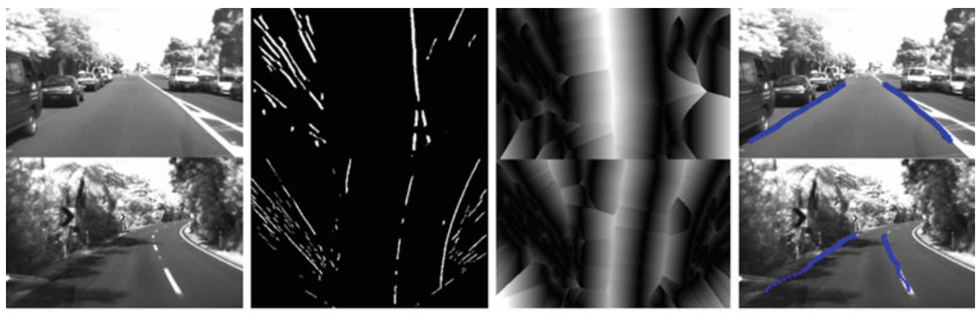

(a) **(b)** **(c)** **(d)**

Fig. 19 Two examples of usefulness of RODT in detection of lane boundary. **a** The input image. **b** The edge map. Note that the edges in the left boundary is far from perfect. **c** RODT of (**b**). The left boundary (*in black*) and the centerline (*in white*) are clear. **d** Lane detection result

has a similar difficulty in detecting edges of left lane boundary. This experiment proves that lane boundaries can be detected in RODT if there are other abundant edges along them.

6 Conclusions

This paper introduced a new weak lane model, 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, provide information for detection of the border or centerline of a lane, find initial values for the particle filter, and adjust the tracking results conveniently.

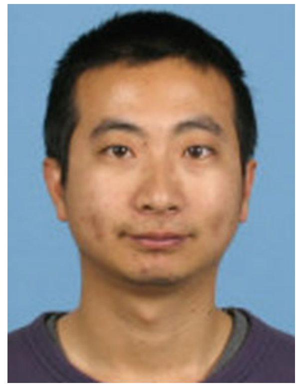
Acknowledgments This work is supported by the National Natural Science Foundation of China under Grant 50875169.

References

1. Aly, M.: Real time detection of lane marks in urban streets. IEEE symposium on intelligent vechicles, pp. 7–12 (2008)
2. Aufrére, R., Chapuis, R., Chausse, F.: A model-driven approach for real-time road recognition. Mach. Vis. Appl. **13**, 95–107 (2001)
3. Broggi, A.: Robust real-time lane and road detection in critical shadow conditions. Intetnational Symposium on Computer Vision, pp. 353–358 (1995)
4. Bertozzi, M., Broggi, A.: GOLD—a parallel real-time stereo vision system for generic obstacle and lane detection. IEEE Trans. Image Process. **7**, 62–81 (1998)
5. Bellino, M., de Meneses, Y.L., Ryser, P., Jacot, J.: Lane detection algorithm for an onboard camera. SPIE Photonics Automob. **5663**, 102–111 (2005)
6. Danescu, R., Nedevschi, S., Meinecke, M.M., To, T.B.: Lane geometry estimation in urban environments using a stereovision system. International Conference on Intelligent Transport System, pp. 271–276 (2007)
7. Dickmanns, E.D., Mysliwetz, B.D.: Recursive 3-D road and relative ego-state recognition. IEEE Trans. Pattern Recog. Mach. Intell. **14**, 199–213 (1992)
8. Felzenszwalb, P.F., Huttenlocher, D.P.: Distance transform of sampled functions. Cornell Computing and Information Science, Technical Report TR2004-1963 (2004)
9. Grisman, J.D., Thorpe, C.E.: SCARF: A color vision system that tracks roads and intersections. IEEE Trans. Robot. Autom. **9**, 49–58 (1993)

10. Isard, M., Blake, A.: Condensation: conditional density propagation for visual tracking. Int. J. Computer Vis. **29**, 5–28 (1998)
11. Jochem, T.M., Pomerleau, D.A., Thorpe, C.E.: MANIAC: a next generation neurally based Autonomous road follower. International Conference on intelligent autonomous systems (1993)
12. Jung, C.R., Kelber, C.R.: A lane departure warning system based on a linear-parabolic lane model. IEEE Symposium on Intelligent Vehicles, pp. 891–895 (2004)
13. Kim, Z.: Realtime lane tracking of curved local road. IEEE Intelligent Transport System, pp. 1149–1155 (2006)
14. Kim, Z.: Robust lane detection and tracking in challenging scenarios. IEEE Trans. Intell. Transp. Sys. **9**, 16–26 (2008)
15. Kreucher, C., Lakshmanan, S.: LANA: a lane extraction algorithm that uses frequency domain features. IEEE Trans. Robot. Autom. **15**, 343–350 (1999)
16. Kreucher, C., Lakshmanan, S.: A frequency domain approach to lane detection in roadway images. International Conference Image Processing, pp. 31–35 (1999)
17. Khosla, D.: Accurate estimation of forward path geometry using two-clothoid road model. IEEE Symp. Intell. Veh. **1**, 154–159 (2002)
18. Lipski, C., Scholz, B., Berger, B., Linz, C., Stich, T.: A fast and robust approach to lane marking detection and lane tracking. IEEE Southwest Symposium on Image Analysis, pp. 57–60 (2008)
19. McCall, J.C., Trivedi, M.M.: "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation". IEEE Trans. Intell. Transp. Sys. **7**, 20–37 (2006)
20. Muad, A.M., Hussain, A., Samad, S.A.: Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system. IEEE TENCON, pp. 207–210 (2004)
21. Nasirudin, M.A., Arshad, M.R.: A feature-based lane detection system using hough transform method. International Symposium on Intelligent Transport System, pp.166–169 (2007)
22. Sehestedt, S., Kodagoda, S., Alempijevic, A., Dissanayake, G.: Efficient lane detection and tracking in urban environments. European Conference on Mobile Robots, pp. 126–131 (2007)
23. Pomerleau, D.: RALPH: Rapidly adapting lateral position handler. IEEE Symposium on Intelligent Vehicles, pp. 506–511 (1995)
24. Klette, R., Rosenfeld, A.: Digital Geometry. Morgan Kaufmann, San Francisco (2004)
25. Wang, Y., Shen, D., Teoh, E.K.: Lane detection using Catmull-Rom spline. IEEE International Conference on Intelligent Vehicles, pp. 51–57 (1998)
26. Wang, Y., Teoh, E.K., Shen, D.: Lane detection and tracking using B-Snake. Image Vis. Comput. **22**, 269–280 (2004)
27. Wang, Y., Bai, L., Fairhurst, M.: Robust road modeling and tracking using condensation. IEEE Trans. Intell. Transp. Sys. **9**, 570–579 (2008)
28. Wedel, A., Franke, U., Badino, H., Cremers, D.: B-spline modeling of road surfaces with an application to free space estimation. IEEE Transactions on Intelligent Transport System, special issue for IV'08 (2008)
29. Wu, T., Ding, X.Q., Wang, S.J., Wang, K.Q.: Video object tracking using improved chamfer matching and condensation particle filter. SPIE-IS & T Electronic Imaging. **6813**, 0.41–04.10 (2008)
30. Yagi, Y., Brady, M., Kawasaki, Y., Yachida, M.: Active contour road model for smart vehicle. Int. Conf. Pattern Recog. **3**, 3819–3822 (2000)
31. Zhang, J., Naqel, H.: Texture-based segmentation of road images. IEEE International Conference on Intelligent Vehicles, pp. 260–265 (1994)
32. Zhou, Y., Xu, R., Hu, X., Ye, Q.: A robust lane detection and tracking method based on computer vision. Meas. Sci. Tech. **17**, 736–745 (2006)

Author Biographies



Jiang Ruyi received the Bachelor of Mechanical Engineering from Zhengzhou University China in 2005. Currently, he is a Ph.D. candidate in Shanghai Jiao Tong University, China. His interests include: vision-based road modeling, visual SLAM, and robotics.

Klette Reinhard is professor at The University of Auckland, chairing work in various projects in multimedia imaging, such as vision-based driver assistance, automated track reading, calculation of shortest Euclidean paths, or non-photorealistic rendering. He was an associated editor of IEEE PAMI in 2002-2008, and is currently on the editorial board of IJCV. In 2004 he published a book, jointly with the late Azriel Rosenfeld, on geometric algorithms for digital picture analysis, and in 2008 a book, jointly with Fay Huang and Karsten Scheibe, on panoramic imaging. He supervised 16 Ph.D. students and about 100 MSc students so far, and he has more than 200 publications in peer-reviewed journals or conferences. He was program co-chair of ACCV 2010.

Vaudrey Tobi completed his Bachelor of Engineering in Engineering Science achieving 1st class honours from the University of Auckland in 2004. He submitted his Ph.D. to the University of Auckland in September 2010. His interests include: stereo and optical fusion, creating "robust" solutions for varying lighting conditions and noise, parallel computing for real-time performance.

Wang Shigang received the Ph.D. degree in HuaZhong University of Science and Technology (HUST), Wuhan, China in 1993, as a postdoctoral fellow in the Department of Mechanical Engineering at HUST from 1993 to 1995. Working in the School of Mechanical Engineering, SJTU since 1995. Prof. Wang is now a chief-professor of the director of Institute of Mechatronics and Automation Technology. He is also the director of Research Center for Microelectronic Equipment at SJTU. Prof. Wang's research interests include Machine Vision and Pattern recognization, Robotics, Design and Control of Complex Mech-Electrical System etc.