Machine Vision and Applications (2016) 27:175–191
DOI 10.1007/s00138-015-0735-5

ORIGINAL PAPER

Vision-based approach towards lane line detection and vehicle localization

Xinxin Du[1] · **Kok Kiong Tan**[1]

Received: 25 June 2014 / Revised: 27 September 2015 / Accepted: 1 November 2015 / Published online: 19 November 2015

Abstract Localization of the vehicle with respect to road lanes plays a critical role in the advances of making the vehicle fully autonomous. Vision based road lane line detection provides a feasible and low cost solution as the vehicle pose can be derived from the detection. While good progress has been made, the road lane line detection has remained an open one, given challenging road appearances with shadows, varying lighting conditions, worn-out lane lines etc. In this paper, we propose a more robust vision-based approach with respect to these challenges. The approach incorporates four key steps. Lane line pixels are first pooled with a ridge detector. An effective noise filtering mechanism will next remove noise pixels to a large extent. A modified version of sequential RANdom Sample Consensus) is then adopted in a model fitting procedure to ensure each lane line in the image is captured correctly. Finally, if lane lines on both sides of the road exist, a parallelism reinforcement technique is imposed to improve the model accuracy. The results obtained show that the proposed approach is able to detect the lane lines accurately and at a high success rate compared to current approaches. The model derived from the lane line detection is capable of generating precise and consistent vehicle localization information with respect to road lane lines, including road geometry, vehicle position and orientation.

Electronic supplementary material The online version of this article (doi:10.1007/s00138-015-0735-5) contains supplementary material, which is available to authorized users.

✉ Xinxin Du
a0107257@nus.edu.sg

Kok Kiong Tan
eletankk@nus.edu.sg

[1] Department of Electrical and Computer Engineering, National University of Singapore, 117576 Singapore, Singapore

Keywords Lane line detection and tracking · Vehicle localization · Road shape modelling and interpretation · Sequential RANSAC

1 Introduction

Recent advances in autonomous vehicles have drawn much attention and interests from researchers, media and the general public. Localization of the vehicle with respect to road lanes is one of the fundamental functions to be enabled in these vehicles to assimilate them into our everyday lives. Vision-based lane line detection is a common approach to address such localization problems. The vehicle position and orientation with respect to the road can be derived from the detection results.

A vast amount of research work has been done in this domain since a few decades ago [24]. However, it is yet to be completely solved and has remained as a challenging problem due to the wide range of uncertainties in real traffic road conditions, which may include shadows from cars and trees, variation of lighting conditions, worn-out lane markings and other markings such as directional arrows, warning words and zebra crossings.

A detailed comparison on existing lane detection approaches is provided by Hillel et al. in [10]. Most of them share two common steps: (1) Lane line candidate extraction using different image features such as edges [14,18] and colour [23] [22], or using machine learning methods such as support vector machine [12], boost classification [6,7]. (2) Model fitting to straight lines [11] or parametric curves [15,21], which can be used to derive vehicle pose for localization.

Some algorithms may also include a third tracking step to impose temporal continuity, where the detection result in the current frame is used to guide the next search through filter mechanisms, such as Kalman filter [2,18] and particle filter [4,12].

A common disadvantage of a feature-based lane line candidate extraction is the sensitivity to noise because it is mainly done through threshholding gradient magnitude. However, edges of shadows and surrounding objects tend to have higher gradient values while poor lighting conditions result in lower gradient values at the lane line boundaries. Thus, gradient threshholding alone is not feasible without other complex adaptation mechanisms. Another disadvantage is that it is limited to a local view [13]. A feature point is detected with only two pixels in the image without considering its connectivity and similarity to the neighbouring pixels.

The machine learning-based approach is subject to the selection of training data and off line training results. To fulfill a good detection, the training set has to contain enough samples under a variety of scenarios. If new situations occur during the test, the approach may fail.

To resolve these issues, López in [16] introduced a newly defined feature ’ridge’, which is claimed to be more suitable than other features (such as edge or colour) to this problem. The noise pixels are then filtered based on ridgeness magnitude and ridge orientation. In model fitting, the author adopted RANSAC to fit a pair of lane lines (right and left lane lines) simultaneously based on the given camera height and pitch angle.

However, the algorithm suffers from the following shortcomings:

1. The noise filtering mechanism is not robust. The author assumed ridge orientation was always along lane line direction and he used this feature to filter out some pixels. But this is not true in the presence of shadows, uneven distribution of lane line painting, worn-out lane lines etc.
2. The filtering based on a fixed ridgeness threshold is not suitable, which leads to under-filtering and over-filtering issues in some situations.
3. Due to the nature of the model fitting method in [16], when the lane exists on only one side, the algorithm will not work properly.
4. The fitted model accuracy is sensitive to the camera viewing angle. Braking/acceleration and road surface changes change the pitch angel. To compensate the variation, the author assigned a series of discrete values within a certain range to pitch angle. But the results due to this variation is still clearly lingering especially when the actual pitch angle is outside the prescribed range.
5. The hyperbola road model itself is not able to provide an accurate fitting for a straight road.

Motivated by these unresolved yet important shortcomings, a more robust lane detection approach based on ’ridge’ identification and incorporation of the modified sequential RANSAC in the model fitting procedure is proposed in this paper to solve the autonomous vehicle localization challenge. The main contributions of the paper include:

1. Implementation of an effective noise filtering mechanism based on adaptive threshold after ridge detection, thus increasing the processing speed and generating improved results.
2. Removal of the effects from camera pitch angle variation on model fitting. In the proposed approach, the model fitting does not rely on having the pitch angle fixed a priori, instead the pitch angle is back calculated from the fitted model.
3. Lane detection on one side of the road even if the other side is missing.
4. Incorporation of a modified version of sequential RANSAC algorithm in model fitting to capture every single lane line in the image independently.
5. Fitting multiple road models simultaneously, including straight line and hyperbola, in the quest for the best matching results.

The paper is organized as follows: Sect. 2 provides a brief introduction to the concept of a ridge. Section 3 will focus on the filtering of noise pixels after the ridgeness thresholding. The modified version of sequential RANSAC for model fitting is elaborated in Sect. 4. Section 5 will illustrate the experimental validation results and the conclusions are drawn in Sect. 6.

2 Ridgeness

The ridge of a grey-level road image refers to the centre axis of the elongated and bright lane lines. The concept can be visualised by considering the image as a landscape with intensity represented along the z axis or height [16]. The intensity increases as we get closer to the centre axis of lane lines, which forms the shape of a ridge as illustrated in Fig. 1. Moreover, ridgeness quantifies how well the pixel neighbourhood resembles a ridge. At the centre axis of the lane line, its neighbourhood at both sides contributes to the formation of the ridge; therefore, it will definitely have a higher ridgeness

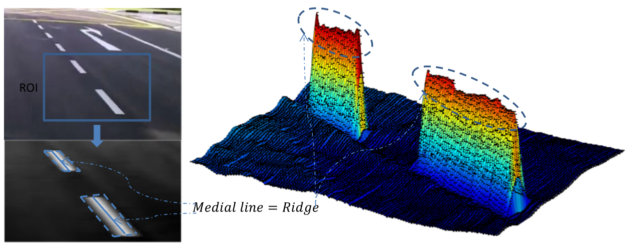

Fig. 1 Concept visualisation of ridge

value. This observation can enable the detection on lane lines by a simple thresholding method. This is also one of the root factors explaining why ridgeness is a more robust feature than edge or colour as it takes its neighbourhood pixels into account instead of just two pixels.

Ridgeness is a loosely defined terminology and there exist different versions of mathematical definitions. In this paper, we adopt the version of López in [16] to compute ridgeness since the approach based on it has been proven to be largely effective.

First, the original grey-level image $L(x)$ is convoluted (*) with a 2D Gaussian filter G_{σ_d}. $L(x)$ is taken as the intensity value (I) in HSI colour space as it has clear advantages than H or S or other colour spaces (such as RGB) [22].

$$L_{\sigma_d}(\mathbf{x}) = G_{\sigma_d}(\mathbf{x}) * L(\mathbf{x}) \tag{1}$$

G_{σ_d} is anisotropic Gaussian kernel with covariance matrix $\sum = diag(\sigma_{dx}, \sigma_{dy})$ where σ_{dy} is constant and σ_{dx} increases with row number which is set equal to half of the lane line width. It depends on the camera focal length and pitch angle φ.

The gradient vector field at each pixel along row (u) and column (v) direction is computed as

$$\mathbf{w}_{\sigma_d}(\mathbf{x}) = (\partial_u L_{\sigma_d}(\mathbf{x}), \partial_v L_{\sigma_d}(\mathbf{x}))^T \tag{2}$$

A 2×2 matrix $s_{\sigma_d}(\mathbf{x})$, similar to Hessian matrix, is computed by dot product $(\cdot)$ of gradient vector for each pixel:

$$s_{\sigma_d}(\mathbf{x}) = \mathbf{w}_{\sigma_d}(\mathbf{x}) \cdot \mathbf{w}^T_{\sigma_d}(\mathbf{x}) \tag{3}$$

The structure tensor field is computed by convoluting each $s_{\sigma_d}(\mathbf{x})$ matrix with another Gaussian filter G_{σ_i}:

$$S_{\sigma_d \sigma_i}(\mathbf{x}) = G_{\sigma_i}(\mathbf{x}) * s_{\sigma_d}(\mathbf{x}) \tag{4}$$

The eigenvector $\mathbf{w'}_{\sigma_d \sigma_i}(\mathbf{x})$ is obtained corresponding to the highest eigenvalue of $S_{\sigma_d \sigma_i}$. There exists explicit solution for eigenvector and eigenvalue to speed up the computation.

Projecting $\mathbf{w'}_{\sigma_d \sigma_i}(\mathbf{x})$ onto $\mathbf{w}_{\sigma_d}(\mathbf{x})$ as:

$$p_{\sigma_d \sigma_i}(\mathbf{x}) = \mathbf{w'}_{\sigma_d \sigma_i}{}^T(\mathbf{x}) \cdot \mathbf{w}_{\sigma_d}(\mathbf{x}) \tag{5}$$

Define a new vector field $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$ at each pixel as:

$$\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x}) = sign(p_{\sigma_d \sigma_i}(\mathbf{x}))\mathbf{w'}_{\sigma_d \sigma_i}{}^T(\mathbf{x}) \tag{6}$$

The ridgeness is then defined by the positive value of divergence of $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$.

$$\widetilde{k}_{\sigma_d \sigma_i}(\mathbf{x}) = -div(\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})) \tag{7}$$

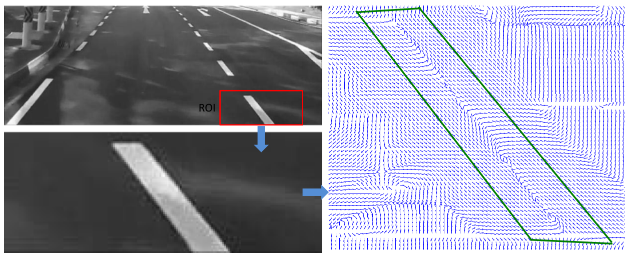

Fig. 2 Illustration on ridge orientation $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$

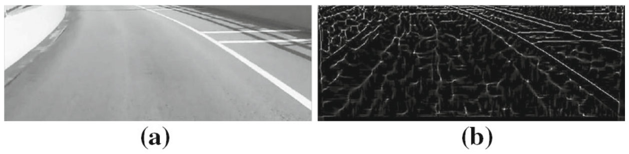

Fig. 3 Original image and its corresponding grey-level image based on ridgeness value. **a** Original, **b** Ridgeness

Figure 2 illustrates the $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$ field orientation of ROI in the original image. The green box highlights the corresponding lane line boundaries. It shows that due to the tree shadows, $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$ orientation deviates from lane line direction, especially for the pixels along the lane line medial axis.

Figure 3 provides an example of the grey-level image based on ridgeness value. It is clear that the medial axis of the lane line is brighter than the rest, which enables the following processing algorithms.

For more detailed explanation on the terminologies and parameter setups, interested readers may refer to [16].

3 Noise filtering mechanism

As mentioned previously, the original approach in removing noise pixels is not robust and effective since a fixed threshold is implemented and $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$ is not always along lane line direction. Here, we propose an adaptive thresholding mechanism based on ridgeness value, which avoids checking $\widetilde{\mathbf{w}}_{\sigma_d \sigma_i}(\mathbf{x})$ direction.

Based on the fact that lane line medial axis always has higher ridgeness value, it can be selected by thresholding. To have an adaptive threshold, the ridgeness histogram is needed.

The image size is 480×640, but the image processing takes effect on the bottom half of the image only to save processing time. And based on the camera setup, only the bottom half contains useful information for lane line detection while the top half mainly consists of sky and road portions that are too far to see. The number of pixels consisting of the longest lane medial axis for one line must be less than $240 + 640 = 880$. Since lane lines exist on both sides and sometimes, double white lane lines may be used on both sides, the number of pixels for lane medal axis must be less

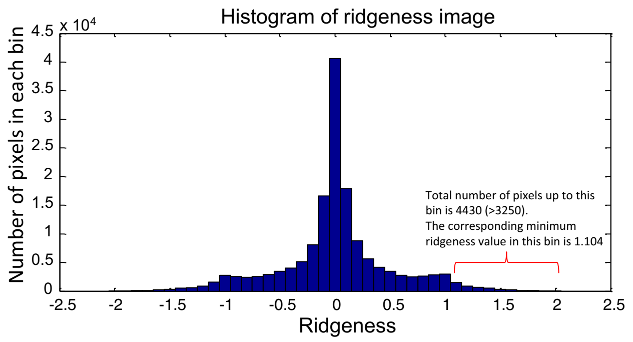

Fig. 4 Histogram based on the ridgeness image

than $4 \times 880 = 3520$. Therefore, 3520 can be a conservative estimation of the maximum number of lane line medial axis pixels. In other words, the number of pixels resulting from lane line detection must be less than 3520.

After the ridgeness calculation, the histogram of the ridgeness grey-level image can be extracted, varying from -2 to 2 with a bin size of 0.1. Each bin is summed in descending order until the number of pixels exceeds 3520. The corresponding minimum ridgeness value in that bin will be set as the threshold. Figure 4 shows the histogram plot based on Fig. 3b and the summing up process.

After thresholding, as shown in Fig. 5c, almost all the lane line medial axis pixels are extracted. But some of the medial axis are disconnected and a lot of noise pixels still exist, resulting from shadows and irregularities on the pavements.

To compensate for these, first a morphographic ‘bridge’ operation to connect pixels with gap of one pixel is applied and then followed by a connected component labelling operation. The corresponding component is removed if its number of pixels is less than a prescribed threshold number. This threshold can be worked out based on the minimum number of pixels required to form a lane line segment medial axis. For breaking lane lines, according to our on-field measurement, the shortest segment is about 1 m. Based on the camera nominal pitch angle, its intrinsic parameters and assuming a segment at the furthest distance in the camera view (bottom half), the minimum number can be calculated approximately. In our setup, the threshold number is 6.

Labels are further removed if the average intensity is less than a threshold as highlighted in [22]. The threshold is set conservatively low to cater for the case when lane lines are obscured by strong shadows or the illumination condition is poor. The number implemented here is 150.

Figure 5 illustrates how the lane line candidate pixels are selected from the original image systematically using the proposed noise filtering mechanism. The red pixels in Fig. 5d indicate the effects from bridge connection. Both quantitative and qualitative comparisons between the original approach in [16] and our proposed one are presented in Sect. 5. They show that the proposed approach can outperform the original one most of the time.

4 Model fitting with modified sequential RANSAC

4.1 Lane line model construction

Many lane line models have been proposed in the literature, ranging from straight lines to spline and conical curves. For example, in [9], circle arc model was implemented and the best fitting circle was found on the road surface using a Hough transform. There is no clear conclusion on which one is the best. However intuitively, assuming the road surface to be flat and lane lines to be parallel, there should exist an explicit relationship between the real road lane line geometry and the projected lane line in the image subject to the camera position and orientation. This kind of relationship can provide useful information to autonomous vehicle navigation system.

Leveraging on this consideration, we adopt the road model proposed by Guiducci [8]. The simplified version provided

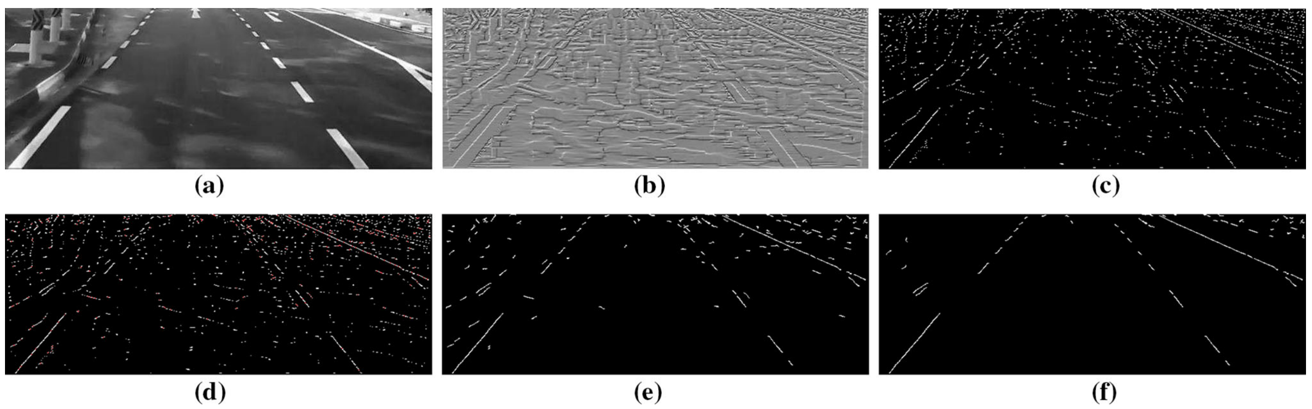

Fig. 5 Proposed noise filtering mechanism. **a** Original, **b** ridgeness, **c** ridgeness threshold, **d** bridge connection, **e** min. structure removal, **f** intensity check

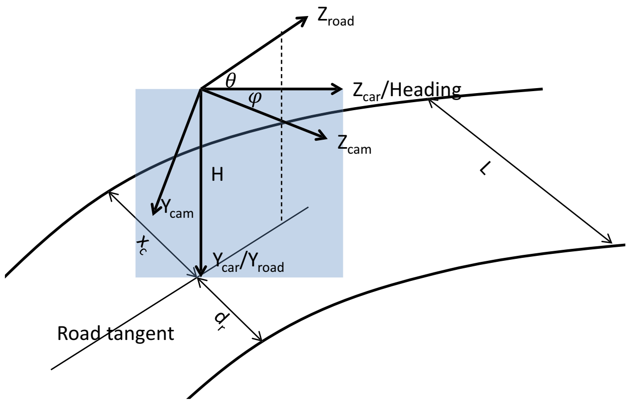

Fig. 6 Road model parameters illustration

in [16] and [1] is shown in (8) and (9) for the left and right lanes, respectively.

$$u_l = E_u\left(\frac{\theta}{\cos\varphi} + \frac{\cos\varphi}{HE_v}x_c(v_l + E_v\tan\varphi) + \frac{E_vHC_0/\cos^3\varphi}{4(v_l + E_v\tan\varphi)}\right) \quad (8)$$

$$u_r = E_u\left(\frac{\theta}{\cos\varphi} - \frac{\cos\varphi}{HE_v}d_r(v_r + E_v\tan\varphi) + \frac{E_vHC_0/\cos^3\varphi}{4(v_r + E_v\tan\varphi)}\right) \quad (9)$$

To fully understand the model, let us define three coordinate systems attached to global (or road), car and camera, respectively, with the same origin at the camera principal point. θ and φ are camera yaw and pitch angle with respect to global system (Fig. 6).

(u, v) defines the horizontal and vertical pixel count of one pixel to the image principal point. E_u and E_v refer to the camera horizontal and vertical focal lengths in the unit of pixel/m. All these parameters (principal point, E_u and E_v) are camera dependent and can be obtained uniquely through calibration. Interested readers may refer to Bouguet's toolbox on the calibration process [3].

H is the height of the camera to the road surface, which can be measured in advance.

φ is the camera pitch angle. Ideally, this is a fixed value and can be measured in advance as well. However, in reality, it varies when the car is slowed/stopped with the brake, accelerating, running on uneven road, etc. The value is so critical to the final fitting result that it cannot be treated as a constant. To compensate for the variation, in [16], the author assigned a series of discrete values within a certain range. However, this approach is sensitive to quantization noises. In this paper, we will consider φ as an unknown and will show that it can be back-estimated instead accurately.

θ defines the angle between car heading direction and road tangent.

C_0 is the lateral curvature of real road with unit of m^{-1}. If the road is straight, $C_0 = 0$ and (8) and (9) represent a line.

x_c and d_r represent the distances from the car to the left and right road lane lines. $L = x_c + d_r$ is the lane width.

For each side of the road model, there are four unknown parameters to be determined through model fitting, namely φ, θ, C_0 and $x_c (or\ d_r)$. Although there are parallelism relationships between left and right lanes, we will ignore this relationship and take them to be independent for the first pass model fitting and only merge/fuse the results based on this relationship in a latter process when both left and right lanes are confirmed to exist. This is to cater to the case when the lane line only exists on one side.

To facilitate the model fitting process, the road model can be further simplified to the form of a hyperbola with A, B, C and D as unknowns.

$$u = \frac{A}{v - D} + Bv + C \quad (10)$$

In the matrix form (Cr is symmetrical):

$$P^T \cdot Cr \cdot P = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & -0.5 & 0.5D \\ -0.5 & B & 0.5E \\ 0.5D & 0.5E & F \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0, \quad (11)$$

where $E = C - BD$, $F = A - CD$, $Cr \cdot P$ represents the tangent of hyperbola at point P. This matrix form will be very useful in model fitting.

4.2 Model fitting

As mentioned, we will fit model independently for left and right lanes. Therefore, the image is deliberately separated into right and left parts. For simplicity, we choose the vertical center axis of the image as the separation line. If the connected component is across the center axis, it will be classified according to the number of pixels on each side. If it has more pixels on the left, then it belongs to the left part and vise versa.

To determine one model, four pixels are required. However, most of the time, there will be much more than four candidate pixels. Moreover, some are outliers and multiple models may exist. For such a multiple model fitting problem with outliers, there are several popular techniques in the literature, such as sequential RANSAC, multiRANSAC, residual histogram analysis, J-linkage, kernel fitting and energy minimization PEARL. As concluded by Fouhey [5] in his study, sequential RANSAC is a strong first choice due to its effectiveness.

As the name implies, sequential RANSAC implements the RANSAC algorithm a number of times until all models

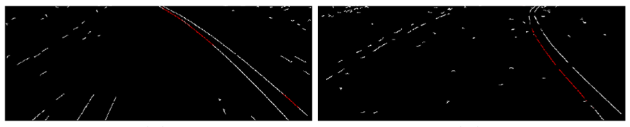

Fig. 7 Inaccurate fitting examples (*red* pixles) from conventional sequential RANSAC

are found or a certain number of iterations has been reached. Once one model is determined, all its supporting data points will be eliminated from the data set and RANSAC is run through the remaining data points to look for another model. However, when the previous model is not correctly selected, the subsequent models will be affected as some of the data points are eliminated wrongly. This occurs more often when double lane lines are used at road bending. Figure below illustrates examples of inaccurately fitted model (highlighted as red) using sequential RANSAC.

To resolve this critical issue and better fit the lane line detection, we propose the following modified sequential RANSAC algorithm by adding a fusion step:

1. Randomly select four points from data set.
2. Create the model and find all its supporting data points.
3. Fuse the new model with previous models. If there are common data points in the new model and the previous ones, only the model with more data points will be kept while the other is discharged. If no common data point exist, the model will be taken as new. This valid because there should not be any intersections between lane lines.
4. Repeat step 1–3 for a certain number of iterations. No data point is eliminated as we do not want this model to affect the consequent fitting results.
5. Up till this step, we have obtained models without any intersection. To expedite the process, eliminate all the supporting data points to these models and repeat step 1–3 with the remaining data set for a certain number of times.

For example in Fig 7a, in one iteration, the inadequate model represented by the red pixels is identified. In another, the subsequent iteration, the accurate model represented by the inner line, is also identified. Then it will be fused with the inadequate model. Since they share some data points (the red points at the top of the image), only the accurate model is kept and the inadequate model is discharged because it has less data points.

But if using conventional sequential RANSAC, once the inadequate model is identified before the accurate model, all the red pixels will be removed from the data pool. Almost half of the inner line data points are removed wrongly: even the accurate model can be identified later, it will not be kept if the number of remaining inner line data points is less than that of red data points.

The results from this modified sequential RANSAC are several non-intersecting models from both sides. The next step is to pair up models and select the best one from all the possible pairs.

Before explaining the pairing mechanism, we would like to elaborate more on step 2, which is the core of RANSAC. Although any four arbitrary points (not on the same line) can generate one unique hyperbola, not every result can describe the road properly. Before searching for supporting data points in step 2, we need to validate the model first. The explicit form of $A,\ B,\ C$ and D is shown in (12)–(15):

$$A = C_0 H E_u E_v / (4\cos^3(\varphi)) \tag{12}$$

$$B = x_c E_u \cos(\varphi)/(H E_v) \tag{13}$$

$$C = \theta E_u / \cos(\varphi) + x_c E_u \sin(\varphi)/H \tag{14}$$

$$D = -\tan(\varphi) E_v \tag{15}$$

The first validation is on D. Its value varies within the range determined by the physical limits of φ. The change of φ is a direct result from the car suspension systems. When its front suspension system is fully compressed and the rear one is fully released, the camera points most downwards and φ is maximum. When the front suspension system is fully released and the rear one is fully compressed, the camera points least downwards or even upwards and φ is minimum. The nominal pitch angle φ_n can be calibrated when the car is not moving and not loaded. By referring to the vehicle catalog, we can calculate the maximum angle that the car frame can be tilted, which corresponds to the two extreme conditions. For the testing vehicle used, the maximum angle is $\sim 3^\circ$ or $\sim$0.035 rad. Therefore, the range of φ is approximated as $[\varphi_n - 0.035,\ \varphi_n + 0.035]$.

With the valid D, A is further validated based on the road curvature. The absolute value of A has to be less than the value derived from the maximum allowable $|C_0|$ for a feasible and safe turning. For a normal sedan, the minimum feasible turning radius is $\sim$10 m limited by its steering angle. Therefore, in this paper, the maximum allowable $|C_0|$ is set as 0.1 m^{-1}.

B and C are related to the vehicle pose. Since the vehicle can be at any locations on the road, they should not be constrained. In summary, we have defined a subclass of valid hyperbolas based on camera pitch angle limits and road curvature limit.

If it is not able to pass the validation on D or A, the next iteration will begin. Some weird fitting results (highlighted in black) without adding these constraints are shown in Fig. 8.

As mentioned, two base models, hyperbola and straight line, will be fitted simultaneously. This is done in step 2 as well. The four randomly selected points can generate one unique hyperbola and six lines. Out of these seven models, only the valid model with most supporting data points will be

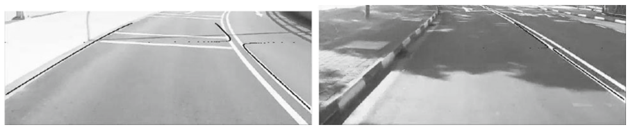

Fig. 8 Wrong fitting results without constraint on hyperbola centre

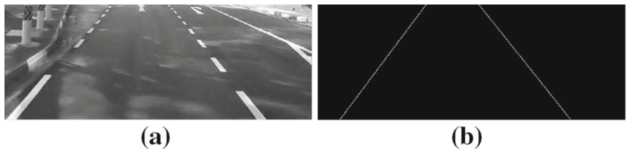

Fig. 9 Perfect model for a straight road. **a** Original, **b** perfect line extraction

kept. Supporting data points will be determined purely based on Sampson's distance d_s [20]. If its $d_s <$ a threshold, the point P will be classified as a supporting point.

$$d_s = \frac{(P^T \cdot Cr \cdot P)^2}{4\left((Cr \cdot P)_1^2 + (Cr \cdot P)_2^2\right)}, \tag{16}$$

where Cr is defined in (11) and $(Cr \cdot P)_n$ refers to the nth element of the vector. For straight lines, $A = 0$.

The reasons why a straight line is needed include the following:

1. To increase the successful rate in finding a valid model. Fig. 9 illustrates a perfect lane line extraction for the straight road with width of 3.4 m. By randomly selecting 4 points on one side, only about 9 % of the 1000 trials is able to provide a valid hyperbola model. It indicates that the rate of fitting is very low by using hyperbola only even under the perfect situation, not to mention situations when there exist noise pixels from the lane line extraction. The original approach [16] suffers the same issue.
2. To increase the accuracy of the model, Fig. 10a plots the number of supporting points for each of the successful fitted models in ascending order. Fig. 10b illustrates their corresponding row-wise maximum deviations. These two figures indicate that out of the 9 % successful trials, only half provides acceptable results in terms of number of supporting points and maximum deviation. The maximum deviation mainly occurs at the top part of the image as shown in Fig. 10d. Even among those successful trials with small row-wise maximum deviations (from trial 60 onwards), the derived parameters from fitted models are not consistent. Fig. 10c plots the D value derived from the corresponding trials. Among the successful trials, it varies from −1000 to 1000 while the ground truth value is 42.8. As shown in (18)–(22), all the localiza-

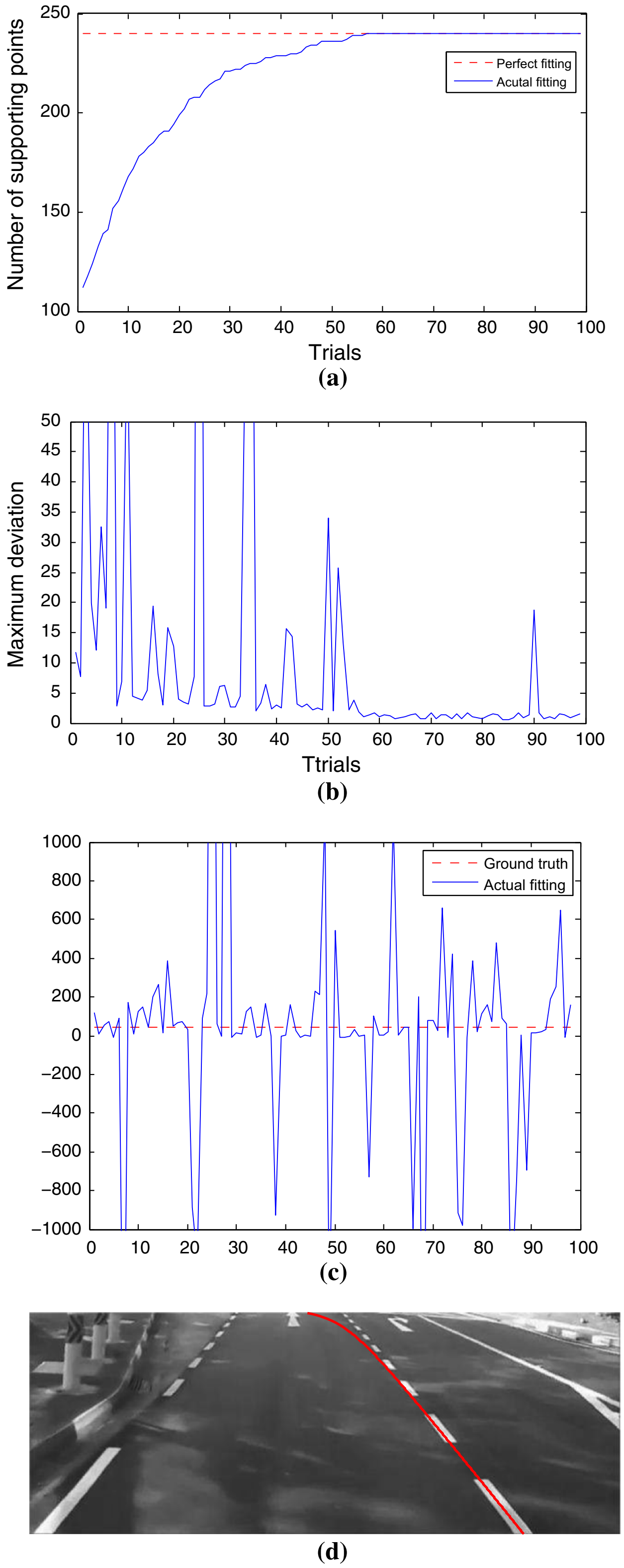

Fig. 10 Hyperbola model accuracy for a straight line. **a** Supporting points, **b** row-wise maximum deviation, **c** derived D from fitted models, **d** example of row-wise deviation

tion information is directly related to D. If this value is not accurate enough, then the whole set of information becomes invalid.

The numbers presented above may vary from trial to trial and line to line, but it provides an idea of how a straight line model helps in fitting a straight road and increasing the accuracy.

4.3 Pairing and parallelism reinforcement

In the case that lane lines on both sides exist, a pairing step is implemented right after all single models are captured. The best pair is determined based on the number of supporting data points and whether the lane width estimated from the model is within a typical road lane width (∼3.4 m).

Equations (8), (9) will give the solution for lane width L. However, most of the time, the results from independent fitting will not follow the parallelism relationship exactly. In turn, we will get a non-constant L. To compensate this and to get the model as accurate as possible, the two paired models need to be fine tuned and adjusted according to the parallelism relationship defined by (8) and (9). This fine-tuning process is called parallelism reinforcement.

It can be derived from (8) and (9) that if one lane model follows (10), then the other lane model will be

$$u = \frac{A}{v - D} + B'v + C + D(B - B') \tag{17}$$

Equations (10) and (17) indicate that to fine tune a pair of lane models, five points are required to determine the five unknowns (A, B, B', C and D). However, the pair of lane models are derived from eight points in model fitting step (four for each side). To balance the contribution from both sides, three out of the four points from one side and two out of the other four points from the other side will be chosen. In total, there will be 48 combinations to fine tune this pair of lane lines. Out of these 48 combinations, the one with the most supporting data points will be the final model for this pair.

Just to highlight, the five unknowns cannot be solved in the form of linear algebras. The set of equations consists of high-order polynomials with multiple variables. Fortunately after tedious conversions and substitutions, it can be downgraded to one-third order polynomial equation with variable D only. Close form solutions are available in the literature to get the three roots explicitly [19]. The real root with the most supporting points will be selected.

After finalizing all pair models, the one which has the most supporting data points with lane width L within the prescribed range will be used to describe the lane where the vehicle is running.

The corresponding vehicle localization information can be calculated as follows:

$$\varphi = \arctan(-D/E_v) \tag{18}$$

$$x_c \text{ or } d_r = BHE_v/(E_u \cos\varphi) \tag{19}$$

$$C_0 = 4A\cos^3\varphi/(HE_uE_v) \tag{20}$$

$$\theta = (C - E_u x_c \sin\varphi/H)\cos\varphi/E_u \tag{21}$$

$$L = x_c + d_r \tag{22}$$

Note that even the lane line exists on one side only, the localization information is still able to be calculated using the same formula above. The only missing information is lane width L.

5 Experiment validation and results

The following tests were carried out on a computer running Window7 OS with Intel i5 CPU processor (3.30 GHz). The algorithm was programmed in MATLAB with mex-C functions and was not optimized to run parallel threads. The average processing time for one image is 0.12 s, of which 20 % is for ridgeness calculation, 17 % for noise filtering, 57 % for model fitting and 6 % for pairing and parallelism reinforcement. The processing speed is sufficient for a vehicle moving at normal speed (∼70 km/h). Because the look-ahead distance in the image is approximately 30 m, the vehicle only moves 2–3 m during one image processing time.

5.1 Noise filtering

The filter results in Sect. 3 using the proposed method and the original method are compared both qualitatively and quantitatively. For the quantitative comparison, only 350 images from 3 video clips are used due to the difficulties in obtaining the ground truth. In each video, the images are sampled consecutively at 10 Hz. The three video clips contain different challenging scenarios, such as breaking lines, dense traffic and worn-off lane lines. The detailed challenges or noise sources are tabulated in Table 1.

Two images from each video clip are shown in Fig. 11 as a qualitative comparison. Column (b) contains the results from the original approach [16] and column (c) contains results from the proposed one.

The first row shows an example on simple images in which little challenging scenarios exist. Both approaches are able to identify the right double while lines. For the left boundary, it is debatable whether it can be treated as lane marking or not. The original approach cannot identify it but the proposed one can.

The second row indicate that both approaches are able to remove the noise from the horizontal speed regulation strip.

Table 1 Noise sources contained in each video clip

Video	Noise sources
1	Other markings, horizontal speed regulation lines
2	Other markings, tree/car shadow, breaking lines, dense traffic, worn-off markings
3	Other markings, tree/car shadow, breaking lines, horizontal speed regulation lines

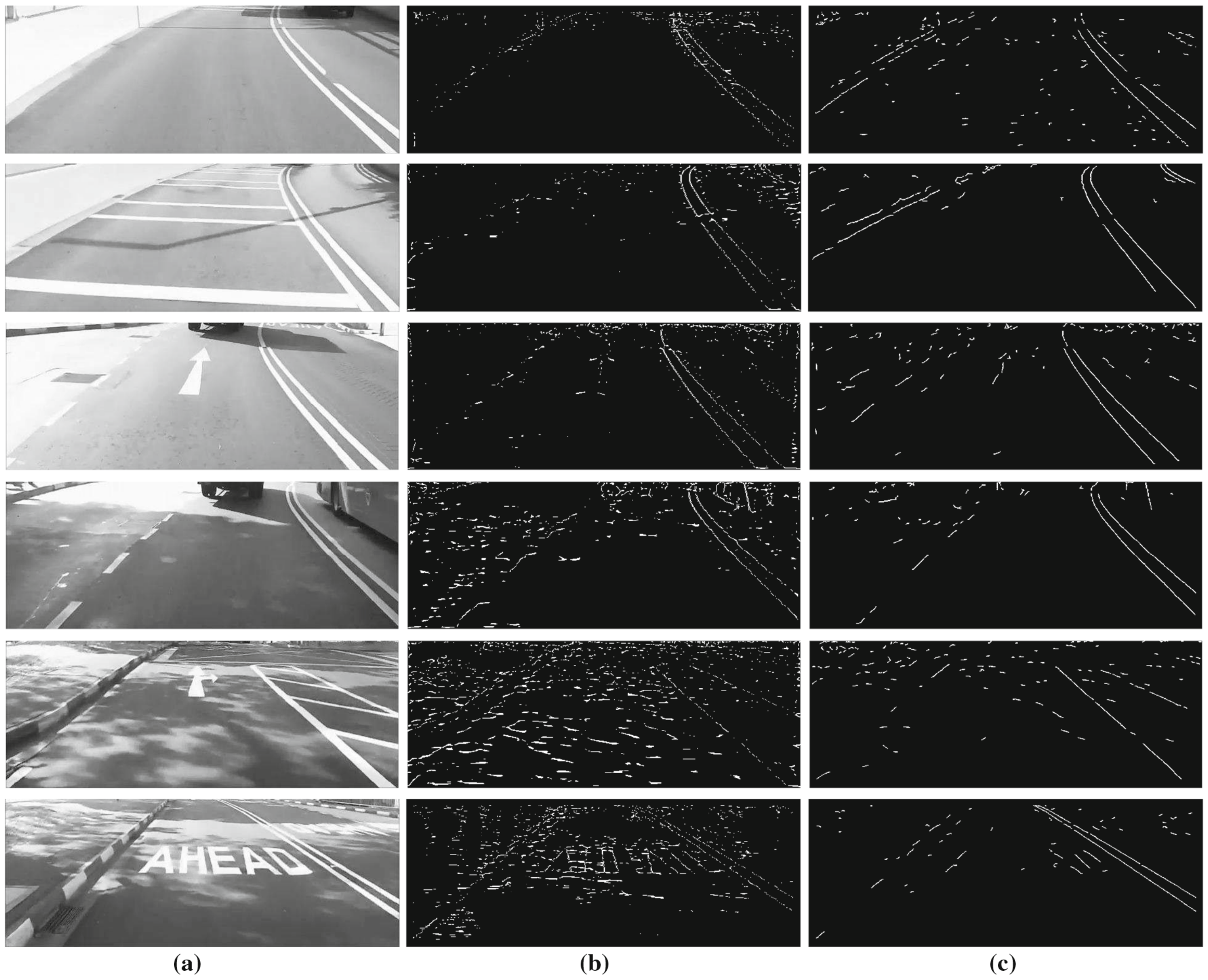
(a) **(b)** **(c)**

Fig. 11 Noise filtering mechanism comparison (Col **a** Original images. Col **b, c** Image processing results from original approach [16] and the proposed one)

The third row depicts the results on worn-off lane markings (left lane line). The proposed approach is able to identify the three lane line segments counting from bottom on the left but the original one almost fails.

In the fourth row, the nearby vehicle will lead to false detection in the original approach as shown on the right top corner of the image. The non-uniform road color on left side of the image leads to more wrong detections as well in the original approach.

The last two rows illustrate the performances of the two approaches when there are strong tree shadows and letters in the image. The original approach is not able to remove the noise pixels effectively.

From the qualitative comparison, we can conclude that the proposed approach performs similar to the original approach on simple road scenarios (e.g. speed regulation strips). But when the road surface becomes more erratic, non-uniform and complex (e.g. tree shadows), the proposed approach is more capable of removing noise pixels.

In the quantitative comparison, the ground truth is labelled manually according to the 350 images. One example is shown in Fig. 12c. Three ratios, commonly used in ROC (Receiver

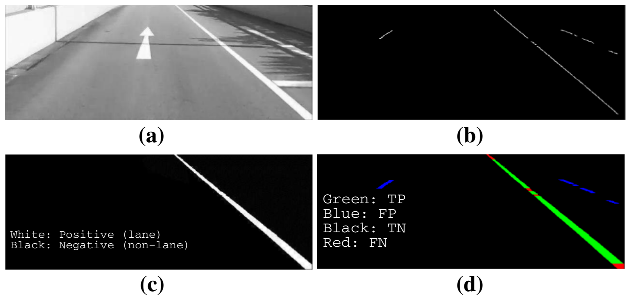

Fig. 12 Illustration on ROC counting (*TP* true positive, *FP* false positive, *TN* true negative, *FN* false negative). **a** Original image, **b** ridge detector, **c** ground truth, **d** ROC

operating characteristic) curve, are introduced for the quantitative evaluation.

$$\text{True positive rate (TPR)} = \frac{\text{TP}}{P} \tag{23}$$

$$\text{False positive rate (FPR)} = \frac{\text{FP}}{N} \tag{24}$$

$$\text{Accuracy (ACC)} = \frac{\text{TP} + \text{TN}}{P + N}, \tag{25}$$

where TP is the number of lane pixels labelled correctly as lane line (true positive); FP is the number of non-lane pixels erroneously labelled as lane line (false positive); TN is the number of non-lane pixels correctly labelled as non-lane line (true negative); P and N are the number of lane line and non-lane line pixels.

A lower FPR and a higher TPR indicate better detection results as it is closer to the perfect result with FPR = 0 and TPR = 1.

Note the way of counting TP, FP and TN. Due to the fact that the result from ridge detector is the lane line medial axis instead of the whole lane area, if one pixel after the ridge detector is classified as TP, all its connected pixels forming the width of the lane on the same row in the ground truth image will be counted as TP. If one pixel is classified as FP in the ridge image, it will be expanded along row direction first. The width of expansion equals to 2 times of its corresponding σ_{dx} (defined in Sect. 2, Eq. 1). For example, if the FP pixel is at the last row and its corresponding σ_{dx} is 12, then the number of FP pixels is approximated to be $2 \times 12 = 24$. TN number equals to the number of N minus FP pixels. Fig. 12 illustrates this process in detail. In Fig. 12d, Green represents TP pixels, Blue represents FP, Black represents TN and Red FN.

As mentioned before, the first filtering step is an adaptive threshold on ridgeness values. To evaluate its performance, we compare its TPR and FPR values to those resulted from a series of fixed threshold as shown in Fig. 13. The blue dots are obtained by setting the ridgeness threshold to the corresponding fixed values and keeping the remaining filtering steps the same.

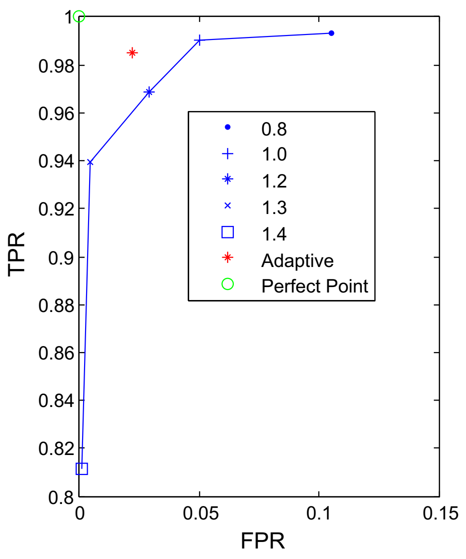

Fig. 13 Average TPR and FPR comparison between ridgeness adaptive threshold and fixed-value threshold

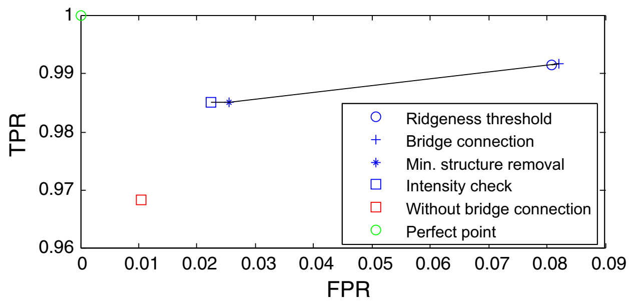

Fig. 14 Average TPR and FPR after each filtering step

The figure shows that the resulting point defined by (FPR, TPR) is closer to the perfect point (0, 1) when using the proposed adaptive threshold. This indicates that adaptive threshold is able to improve the noise filtering performance as compared to fixed-value threshold.

To further analyse the impact from each individual filtering step, after running each step, the corresponding ROC values of each image are logged and the average FPR and TPR values over the 350 images after each step are plotted in Fig. 14 (blue indicators).

As can be seen from the figure, the minimum structure removal has the greatest impact in reducing the FPR, which means the noise pixels are largely removed by this step. The intensity check step can further improve FPR without sacrificing TPR significantly.

The bridge connection step seems to have little impact on FPR and TPR. But it is necessary. To verify this, another filtering test on the same set of images is carried out by removing this step but keeping the rest unchanged. The resulting

(FPR, TPR) is shown as the red square in Fig. 14. The FPR is further reduced as compared to the blue square, but the TPR is reduced more significantly, resulting in a longer distance to the perfect point. The bridge connection step prevents some true positive points from being removed by the minimum structure removal step.

To further evaluate the performance of the proposed noise filtering mechanism, we also compare its ROC values with those obtained from the original approach [16]. Figure 15a–c illustrate the comparison results frame by frame, where the black dashed lines mark the separation of different video clips.

For the first video, both approaches achieve similar performances as the road conditions are relatively simple. But when more challenging scenarios (e.g. tree shadows) come into the image as shown in the second and third video, the proposed algorithm will scarify the TPR by a little and maintain a low FPR as compared to the original approach. This trade-off strategy will improve the model fitting efficiency significantly as to be shown later in this section. In other words, the proposed algorithm will remove the noise pixels more effectively. This observation is similar to that from the quantitative comparison.

Overall, both methods achieve a similar TPR (the proposed is slightly worse by 0.2 % only), but the proposed method has much better performance in FPR (lower by 3 %) and ACC (better by 3 %).

Figure 15d presents the (FPR, TPR) distribution for each image. It is clear that the result cluster based on the proposed method is closer to the perfect point (0, 1) than that from the original method.

To further analyse the impact of the noise filtering mechanism on the model fitting process, define the following variables: ω is the probability of choosing an inlier (TP) each time a single point is selected from the lane line pixel candidates (TP + FP), p is the probability that the RANSAC process produces a useful result, n is the number of points needed to estimate model parameters and k is the number of iterations for the RANSAC process. The following relationships hold:

$$\omega = \mathrm{TP}/(\mathrm{TP}+\mathrm{FP}) \tag{26}$$

$$1 - p = (1-\omega^n)^k \tag{27}$$

Let the subscript p refer to the proposed method and o refer to the original method, based on (23–25) and (26–27); we can derive

$$\frac{k_p}{k_o}\cdot\frac{\lg(1-p_o)}{lg(1-p_p)} = \frac{\lg\left(1-\left[\frac{\mathrm{TPR}_o\cdot P}{\mathrm{TPR}_o\cdot P+\mathrm{FPR}_o\cdot N}\right]^{n_o}\right)}{\lg\left(1-\left[\frac{\mathrm{TPR}_p\cdot P}{\mathrm{TPR}_p\cdot P+\mathrm{FPR}_p\cdot N}\right]^{n_p}\right)} \tag{28}$$

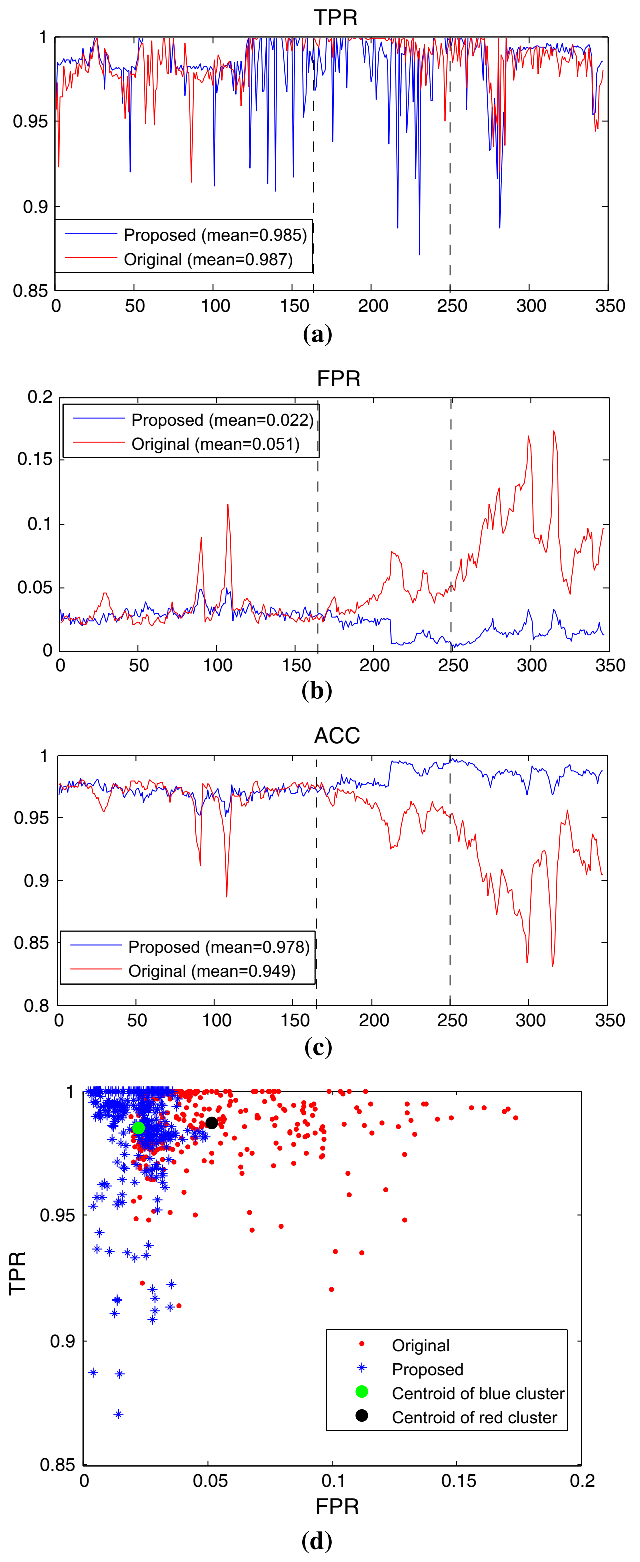

Fig. 15 ROC comparison between the proposed and original algorithms based on 350 images. **a** TPR (true positive rate), **b** FPR (false positive rate), **c** ACC (accuracy), **d** TPR vs. FPR distribution

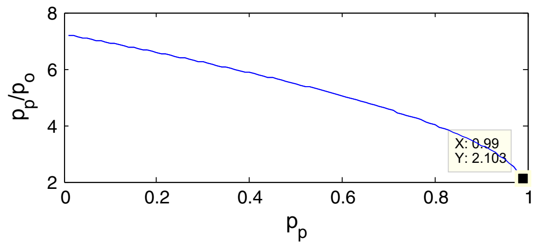

Fig. 16 RANSAC probability analysis of the proposed (p_p) and the original (p_o) approach when running the same number of iterations

To facilitate the analysis, we will use average values based on the 350 images to substitute to the right-hand side of (28). If assuming hyperbola, we can have

$$\frac{k_p}{k_o} \cdot \frac{\lg(1-p_o)}{lg(1-p_p)} = 0.1382 \tag{29}$$

where on average, $P = 3672$, $N = 149928$, $\text{TPR}_o = 0.987$, $\text{FPR}_o = 0.051$, $\text{TPR}_p = 0.985$, $\text{FPR}_p = 0.022$, $n_o = n_p = 4$.

From (29), statistically, to achieve the same probability p from RANSAC fitting ($p_o = p_p$), the number of iterations required in the proposed approach is only 13.8 % of the original one. Furthermore, if running the same number of iterations ($k_p = k_o$), the probability of the proposed approach to get a useful model is always higher than that of the original one as shown in Fig. 16, where x-axis is p_p, varying from 0.01 to 0.99 and y-axis is p_p/p_o. When the proposed approach has the probability of 99 % to find a useful model, the original approach only has 47 %.

From the ROC analysis above, we can conclude that without sacrificing the true positive detection rate very much (worse by 0.2 % only), the proposed filtering mechanism removes the noise pixels much more effectively than the original method and thus it improves the detection accuracy and RANSAC fitting efficiency.

5.2 Modified sequential RANSAC

For a better lane line fitting results, especially for double lines, we modified the conventional sequential RANSAC algorithm by adding in a fusion step. To evaluate the modified RANSAC algorithm, we compared it with the conventional one based on the first video clip. The first half of the video contains single line marking and the second contains double-line marking.

The fitting results from both algorithms were then compared with the ground truth. Their row-wise maximum pixel deviation from the ground truth is depicted frame by frame in Fig. 17.

For the first half of the video, both algorithms perform similarly in fitting single line markings; not much difference is observed. But for the second half, the modified RANSAC

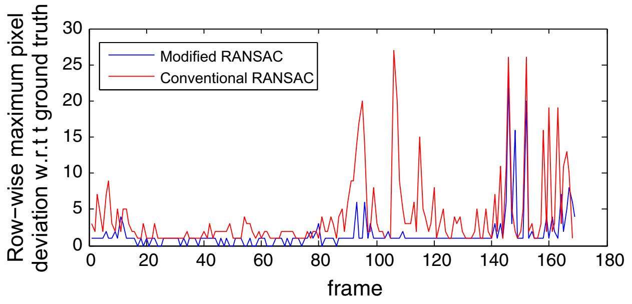

Fig. 17 Line fitting comparison between modified and conventional RANSAC in terms of maximum deviations

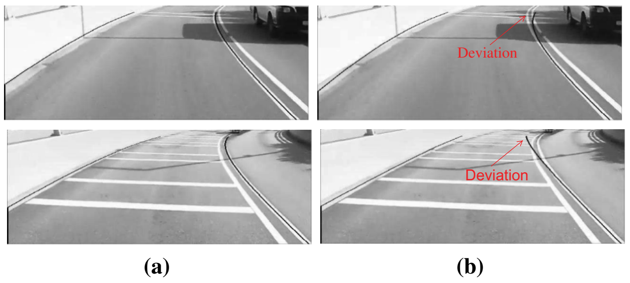

Fig. 18 Comparison of fitting results between modified (column **a**) and conventional (column **b**) RANSAC

algorithm obtains more accurate results when fitting double line markings. The average maximum deviation is only 2.26 pixels for the modified algorithm and 5.63 pixels for the conventional one. It is more often that the conventional algorithm results in large fitting deviations. Two examples are provided in Fig. 18. The major deviations of the conventional algorithm are at the top of the images.

5.3 Simulation

As mentioned previously, the main intended application of this project is to generate vehicle localization information with respect to the road lane and thus enable autonomous driving. This information can be calculated from equation (18)–(22) based on the fitted model.

To verify the model fitting accuracy, we first tested our algorithm on the simulator proposed in [16]. The simulator simulates a sequence of road images (resolution 480 × 640) captured by an onboard camera when changing its orientation (θ and pitch angle), position (x_c) and road geometry (L and C_0). The total number of images is about 2000. Then the proposed approach is tested with the simulated road images and all the unknown parameters for each image are back estimated from the fitted model. At the end, the estimated parameters are compared with the corresponding ones set up in the simulator to evaluate the accuracy. Figure 19 is one of the images generated by the simulator. The corresponding parameters are given beside the figure.

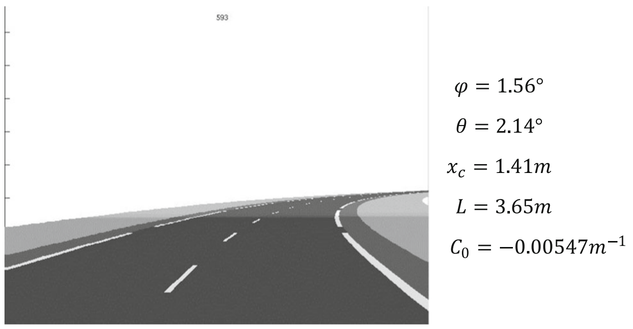

Fig. 19 Simulated road with corresponding parameters

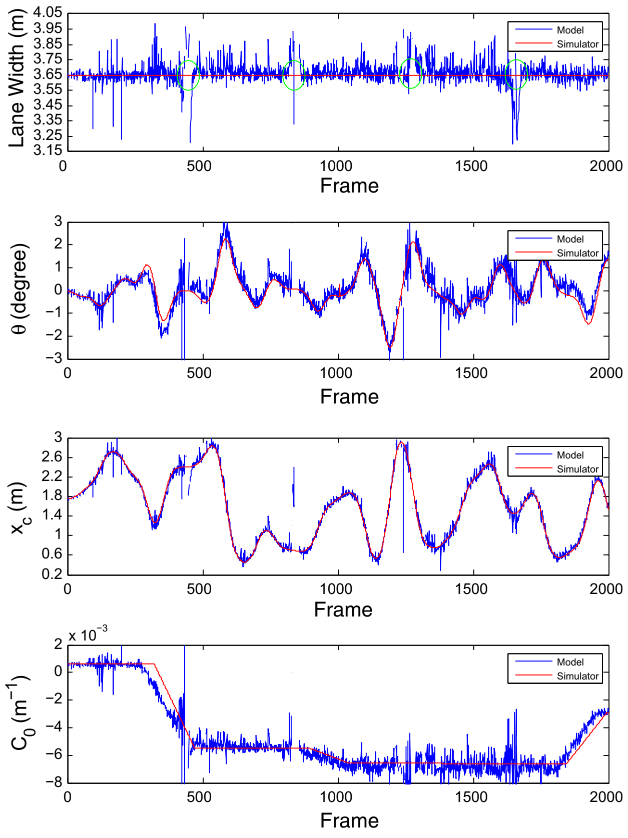

Fig. 20 Model accuracy evaluation

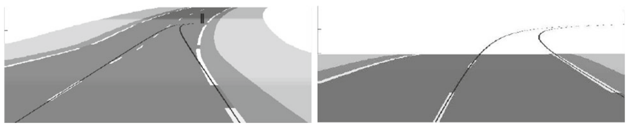

Fig. 21 Inaccurate fitting examples at non-flat road surface

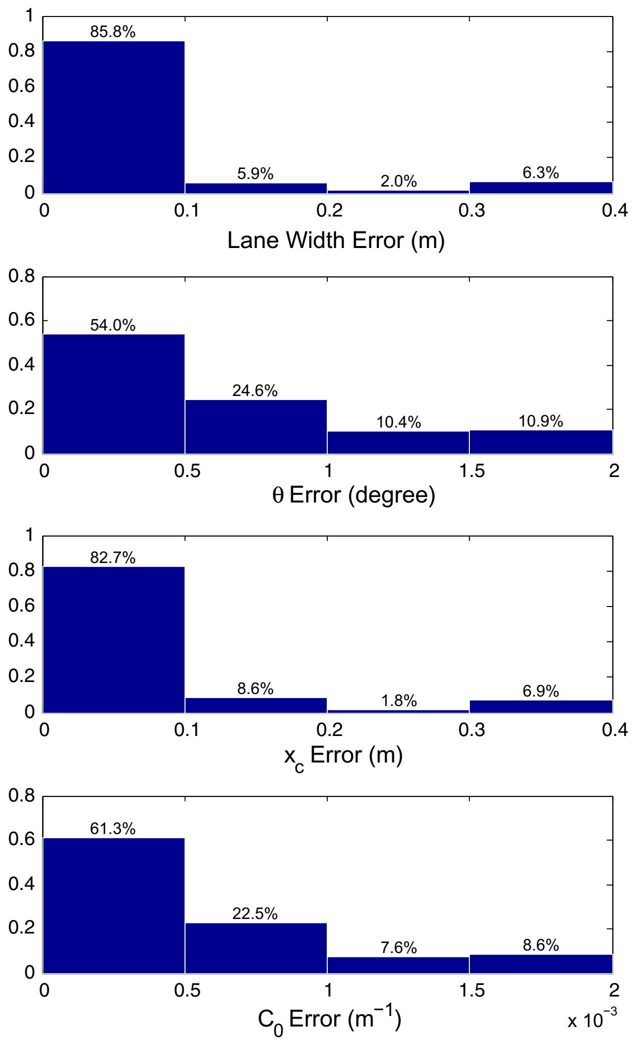

Fig. 22 Model absolute error distribution

Figure 20 illustrates the comparison results of L, θ, x_c and C_0 between the estimated model value (blue) and the simulator ground truth (red). Most of the time, all the estimated values are quite close to their true values. The most prominent errors occur at the locations where road surface is not flat (highlighted by green circles), which violates the assumptions on model construction in Sect. 4.1. The lane line markings in the image do not form hyperbolas or straight lines. Figure 21 shows two fitting examples at location 417 and 827, representing down-hill and up-hill situations, respectively.

The absolute error distributions for the corresponding parameters are shown in Fig. 22. Most of the time (∼90 %), the absolute error for all the parameters is very small.

Compared to the original approach under four cases in Table 2, in terms of root-mean square error, L, θ and x_c are always better. C_0 is a little bit worse than the best performance but better than the remaining three cases. n refers to the number of discrete values assigned to the pitch angle in the original approach. For a fair comparison, we only compare our results to the original one without causal median filter.

Table 2 Lane model parameters comparison between proposed and original approach (root-mean square error)

Parameter	Proposed	Original			
		$n = 1$	$n = 3$	$n = 7$	$n = 41$
L (m)	0.070	0.215	0.222	0.210	0.215
θ (degree)	0.94	1.09	1.75	1.54	1.56
x_c (m)	0.116	0.25	0.26	0.26	0.26
C_0 (m^{-1})	0.0029	0.0027	0.0050	0.0045	0.0046

Another important parameter related to localization is the camera pitch angle, of which the simulation result is illustrated in Fig. 23. Top one is the proposed method while bottom is original with $n = 41$ [16]. Zooming into the figures, we can see that the pitch angle from proposed method is closer to ground truth despite several spikes arising while the original approach provides a coarser results around the true values due to the discrete noise. This indicates that pitch angle can be back-calculated/estimated more accurately than assigning a series of discrete values in advance and searching for the best matching. The root mean square error for the proposed approach is 0.1052 while it is 0.1249 for the original approach.

The simulation results show that the proposed method, even without a priori accurate information of the camera pitch angle, is still able to find a suitable model and generate more accurate geometrical information related to vehicle localization with respect to road lane. In addition, we also show that pitch angle can be back estimated from the model and the value is more precise than the original approach.

5.4 Real-world environment performance

In the literature, most authors tested their algorithms on highways, where the road structure is well constructed most of the time and the environment is more confined and more predictable. To further challenge the capabilities of our algorithm, we extended the test of the proposed approach on normal roads around NUS (National University of Singapore) instead of highways. The situation is more challenging and erratic as shown in Fig. 26.

However, similar to other works, here we face the main difficulties in obtaining precise ground truth. Theoretically, to generate the ground truth, all the five parameters (L, C_0, θ, x_c and φ) mentioned above need to be measured directly at each frame. This is almost impossible by just using any manual measuring instruments.

In [25], the author provides a method of approximating these values based on high-precision GPS and accelerometers, but it requires a lane-level digital map in place which is still a research issue in itself. In [17], a side camera is incorporated to measure the lateral offset from the lane line and the side camera measurement serves as the ground truth for the front camera. It is capable of providing the ground truth for x_c and θ, which are the most essential parameters for the vehicle localization. Therefore, we made use of this setup for ground truth generation.

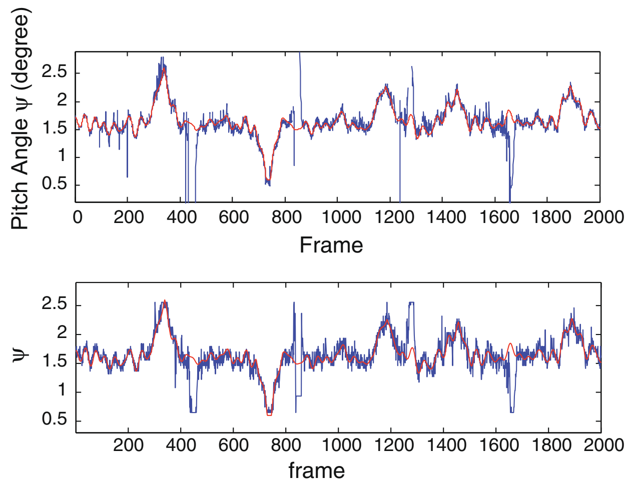

Fig. 23 Pitch angle comparison between proposed (*top*) and original approach (*bottom*). *Red line* represents the simulator ground truth and *blue line* is the estimation results

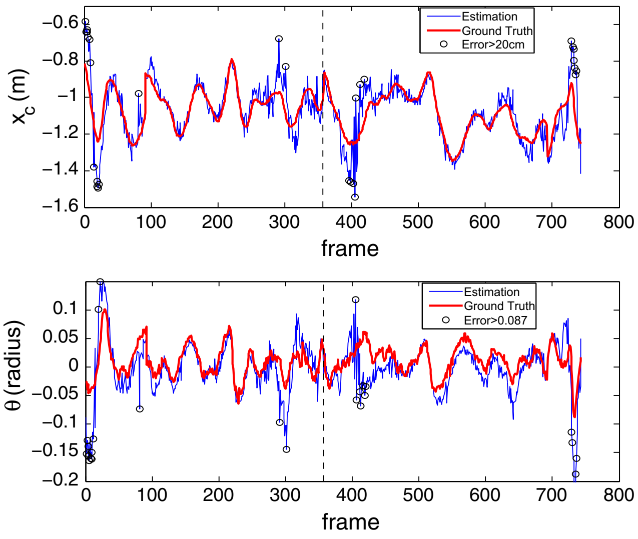

Fig. 24 Evaluation on model accuracy for real world testing

To eliminate the over fitting concern of the proposed algorithm, in this experiment setup, we changed the camera from the Logitech webcam (used in Sect. 5.1) to an industrial CMOS camera. We also purposely offset the camera mounting position from the car centre and misaligned the camera optical axis with the vehicle longitudinal axis (pointing slightly to the side of the car instead of front).

Figure 24 illustrates the comparison results based on two video clips (black vertical dashed line marks the separation

Table 3 Fitting results error statistics

	Mean	Abs mean	STD
θ (rad)	−0.0131	0.0247	0.032
x_c (m)	0.0018	0.0559	0.0811

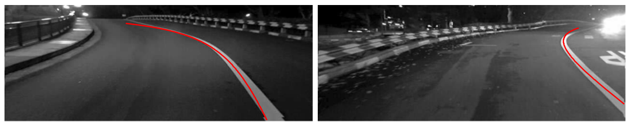

Fig. 25 Inadequate model fitting examples (*red lines*) at non-flat road surface

of the two videos). Both videos were taken at night and contained solid/dash single line markings on one side of the road. The noise sources include non-flat surface, poor illumination conditions, warning letters and glare from the head lights of incoming vehicles. Table 3 shows the corresponding mean error, absolute mean error and the error standard deviation for both parameters.

From the comparison, it is clear that most of the time, the proposed algorithm is able to provide accurate and consistent estimation on vehicle lateral distance x_c to the road boundary and its moving direction θ w.r.t road. The average absolute error in x_c is only 5.6 cm, which is even less than road lane marking width.

The most prominent errors for both parameters, as highlighted by black circles in Fig. 24, occur at the locations where the road surface is not flat and the road is curving. This observation tallies with that from the simulation. Two examples are illustrated in Fig. 25, where the fitted line (red) slightly miss-aligns with the medial axis of the lane line markings at the bottom of the image.

Another observation, similar to the simulation as well, is the non-smoothing estimation, which fluctuates around the ground truth values with small errors. From the statistical analysis in Table 3, the estimation error can be approximated as zero-mean white noise since their mean values are close to zero. This implies that a Kalman filter can be implemented to provide smoother and more consistent results without spikes.

In Fig. 26, we present some quantitative fitting results sampled from the testing sequence under different situations as indicated under each image. Black lines (daytime vision $a \sim j$) and red lines (night vision $k \sim n$) represent the fitted results.

In this section, we demonstrate both qualitatively and quantitatively that the proposed approach works well even with strong disturbances coming into the images. It is capable to provide estimation on vehicle localization with respect to road lane lines, which can be used as a feedback to control an autonomous vehicle to follow the lane.

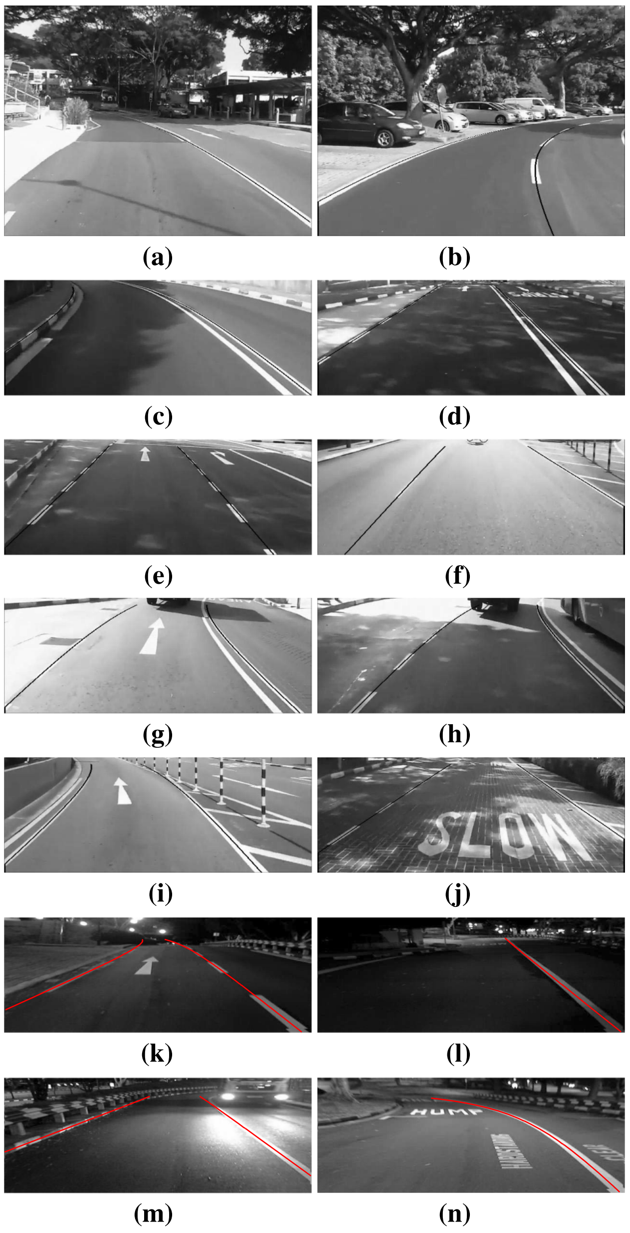

Fig. 26 Fitting results under different situations. $a \sim j$ daytime vision (*black line*) and $k \sim n$ night vision (*red line*). **a** Lane line on one side only **b** Sharp Turn, (**c**) Different Lighting, **d** Shadow, **e** Breaking Line 1, (**f**) Breaking Line 2, **g** Worn-out of lane line, **h** With other vehicles, **i** Different lane line colors, **j** Different Road Surface, **k** Breaking lines, **l** Poor illumination, **m** Glare, **n** Bending

In addition, when lane line exists on one side only (Fig. 26a,l,n), the original approach will fail due to the nature of the algorithm (detecting lane line on both sides simultaneously) but the proposed one is still able to detect the lane line. Under situations with small irregularities on road surfaces (such as d and j), the original approach is not able to remove noise pixels effectively and consequently it will have a higher probability to fail as shown by the probability analysis in Sect. 5.1.

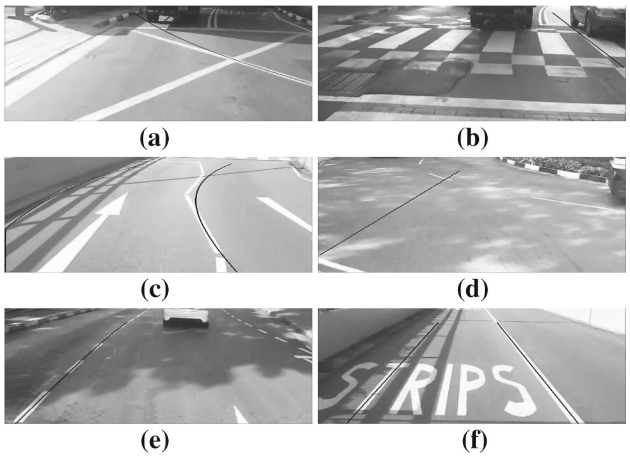

Fig. 27 Situations where both original and proposed approaches do not work properly. **a** Non-stop yellow box, **b** Zebra crossing, **c** Before zebra crossing, **d** Round-about, **e** Non-parallel lane lines, **f** Warning letters

However, there are still extreme situations under which both the proposed and original approaches may not work satisfactorily. These situations include the following:

1. Other markings, which are similar to lane line markings, may result in false detection. For example, the non-stop yellow box (Fig. 27a) and zebra crossing (Fig. 27b).
2. Lane lines do not follow the hyperbola or straight line assumptions. For example, before the zebra crossings, the lane line is in zig-zag shape (Fig. 27c). At round-about, it forms a circle (Fig. 27d)
3. Lane lines are not parallel. For example, at lanes merging location, the lane line width shrinks (Fig. 27e).
4. Sometimes, warning letters on the road will lead to false detection as well (Fig. 27f), especially when lane line exists on one side only.

To address the first and second cases, additional special indicators on the road can be implemented to indicate to the algorithms that the vehicle is approaching these locations so that the results are interpreted accordingly. The false detection resulting from warning letters can be filtered out by adding a tracking step between consecutive images.

6 Conclusion

In this paper, we proposed a robust and reliable vision-based road lane line detection approach which works even under challenging road situations. We demonstrated that the fitted model is capable of providing accurate estimation on vehicle localization information with respect to road lane lines, including the camera pitch angle φ, vehicle heading direction θ, vehicle position to lane line x_c *or* d_r, road width L and road curvature C_0. This information, integrated with depth sensing system, can be implemented in autonomous vehicle navigation.

To further improve the proposed algorithm, a filtering mechanism such as particle filter, incorporated with the vehicle dynamics, will be implemented between consecutive image sequences so that the vehicle pose estimation becomes more consistent and smooth. The similar algorithm will be migrated to stereo vision system to get the depth information. The road curvature estimation will be fused with the information from map and GPS to improve consistency.

References

1. Aufrere, R., Chapuis, R., Chausse, F.: A model-driven approach for real-time road recognition. Mach. Vis. Appl. **13**(2), 95–107 (2001)
2. Borkar, A., Hayes, M., Smith, M.T.: A novel lane detection system with efficient ground truth generation. Intell. Transp. Syst. IEEE Trans. **13**(1), 365–374 (2012)
3. Bouguet, J.Y.: Camera calibration toolbox for matlab. (2010-07). http://www.vision.caltech.edu/bouguetj/calib_doc/
4. Danescu, R., Nedevschi, S.: Probabilistic lane tracking in difficult road scenarios using stereovision. Intell. Transp. Syst. IEEE Trans. **10**(2), 272–282 (2009)
5. David, F., Scharstein, D.: Multi-model estimation in the presence of outliers. Master's thesis, Middlebury College (2011)
6. Fritsch, J., Kuhnl, T., Kummert, F.: Monocular road terrain detection by combining visual and spatial information. Intell. Transp. Syst. IEEE Trans. **15**(4), 1586–1596 (2014)
7. Gopalan, R., Hong, T., Shneier, M., Chellappa, R.: A learning approach towards detection and tracking of lane markings. Intell. Transp. Syst. IEEE Trans. **13**(3), 1088–1098 (2012)
8. Guiducci, A.: Parametric model of the perspective projection of a road with applications to lane keeping and 3d road reconstruction. Comput. Vis. Image Underst. **73**(3), 414–427 (1999)
9. Gyory, G.: Obstacle detection methods for stereo vision as driving aid. In: Advanced Robotics, 2003. International Conference on, IEEE, pp. 477–481 (2003)
10. Hillel, A.B., Lerner, R., Levi, D., Raz, G.: Recent progress in road and lane detection: a survey. Mach. Vis. Appl. **25**(3), 727–745 (2014)
11. Kang, D.J., Jung, M.H.: Road lane segmentation using dynamic programming for active safety vehicles. Pattern Recogn. Lett. **24**(16), 3177–3185 (2003)
12. Kim, Z.: Robust lane detection and tracking in challenging scenarios. Intell. Transp. Syst. IEEE Trans. **9**(1), 16–26 (2008)
13. Li, H., Nashashibi, F.: Robust real-time lane detection based on lane mark segment features and general a priori knowledge. In: Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on, IEEE, pp. 812–817 (2011)
14. Li, Q., Zheng, N., Cheng, H.: An adaptive approach to lane markings detection. In: Intelligent transportation systems, 2003. Proceedings., IEEE, vol. 1, pp. 510–514 (2003)
15. Liu, W., Zhang, H., Duan, B., Yuan, H., Zhao, H.: Vision-based real-time lane marking detection and tracking. In: Intelligent transportation systems, 2008. ITSC 2008. 11th International IEEE Conference on, IEEE, pp. 49–54 (2008)
16. López, A., Serrat, J., Canero, C., Lumbreras, F., Graf, T.: Robust lane markings detection and road geometry computation. Int. J. Autom. Technol. **11**(3), 395–407 (2010)

17. McCall, J.C., Trivedi, M.M.: Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. Intell. Transp. Syst. IEEE Trans. **7**(1), 20–37 (2006)
18. Nedevschi, S., Schmidt, R., Graf, T., Danescu, R., Frentiu, D., Marita, T., Oniga, F., Pocol, C.: 3d lane detection system based on stereovision. In: Intelligent transportation systems, 2004. Proceedings. The 7th International IEEE Conference on, IEEE, pp. 161–166 (2004)
19. Press, W.H.: Numerical recipes in Fortran 77: the art of scientific computing, vol. 1. Cambridge University Press (1992)
20. Sampson, P.D.: Fitting conic sections to very scattered data: an iterative refinement of the bookstein algorithm. Comput. Graph. Image Process. **18**(1), 97–108 (1982)
21. Sivaraman, S., Trivedi, M.M.: Integrated lane and vehicle detection, localization, and tracking: a synergistic approach. Intell. Transp. Syst. IEEE Trans. **14**(2), 906–917 (2013)
22. Sun, T.Y., Tsai, S.J., Chan, V.: Hsi color model based lane-marking detection. In: Intelligent transportation systems conference, ITSC'06. IEEE, pp. 1168–1172 (2006)
23. Tapia-Espinoza, R., Torres-Torriti, M.: A comparison of gradient versus color and texture analysis for lane detection and tracking. In: Robotics symposium (LARS), 2009 6th Latin American, IEEE, pp. 1–6 (2009)
24. Thorpe, C., Hebert, M.H., Kanade, T., Shafer, S.A.: Vision and navigation for the carnegie-mellon navlab. Pattern Anal. Mach. Intell. IEEE Trans. **10**(3), 362–373 (1988)
25. Wang, J., Schroedl, S., Mezger, K., Ortloff, R., Joos, A., Passegger, T.: Lane keeping based on location technology. Intell. Transp. Syst. IEEE Trans. **6**(3), 351–356 (2005)

Xinxin Du received his B.Eng degree in 2011 from Nanyang Technological University (NTU) Singapore. He was an Industrial Engineer in Micron Semiconductor Asia Pte. Ltd. (Singapore) from August 2011 to January 2013, working on production forecasting and system optimization. Since January 2013, he has been with Department of Electrical and Computer Engineering, National University of Singapore (NUS), pursuing a Ph.D degree. His research interests include autonomous electrical vehicle, image processing and motion control.

Kok Kiong Tan received his Ph.D. in 1995 from the National University of Singapore (NUS). Prior to joining the university, he was a research fellow at SIMTech, a national R&D institute spearheading the promotion of R&D in local manufacturing industries, where he has been involved in managing industrial projects in IT and automation. He is currently an associate professor with NUS and his current research interests are in Internet applications, remote control and monitoring, precision motion control and instrumentation, advanced process control and auto tuning and general industrial automation.