Computer Communications 73 (2016) 132–143

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

A real-time lane marking localization, tracking and communication system

Abdelhamid Mammeri*, Azzedine Boukerche, Zongzhi Tang

Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Street, Ottawa, Ontario, Canada

ARTICLE INFO

Article history:
Available online 20 August 2015

Keywords:
In-vehicle computing system
Intelligent transportation system
Vision

ABSTRACT

In this paper, we present an in-vehicle computing system capable of localizing lane markings and communicating them to drivers. To the best of our knowledge, this is the first system that combines the Maximally Stable Extremal Region (MSER) technique with the Hough transform to detect and recognize lane markings (i.e., lines and pictograms). Our system begins by localizing the region of interest using the MSER technique. A three-stage refinement computing algorithm is then introduced to enhance the results of MSER and to filter out undesirable information such as trees and vehicles. To achieve the requirements of real-time systems, the Progressive Probabilistic Hough Transform (PPHT) is used in the detection stage to detect line markings. Next, the recognition of the color and the form of line markings is performed; this it is based on the results of the application of the MSER to left and right line markings. The recognition of High-Occupancy Vehicle pictograms is performed using a new algorithm, based on the results of MSER regions. In the tracking stage, Kalman filter is used to track both ends of each detected line marking. Several experiments are conducted to show the efficiency of our system.

1. Introduction

Lane marking localization and tracking is an important component of in-vehicle computing systems. Lane marking localization and tracking systems have attracted an extensive amount of interest from both academia and the automobile industry. Many architectural and commercial systems have been proposed in the literature, for example [1–3]. Lane marking recognition systems, conversely, have received surprisingly little attention. Lane marking such as lines and pictograms, are important tools for communicating regulations and guidelines in order to keep vehicles in the correct lanes.

In this paper, we present a real-time lane marking localization and communication system able to detect, recognize and track pavement markings. To the best of our knowledge, this is the first system that combines Maximally Stable Extremal Region (MSER) with Hough transform to detect and recognize lane markings. Not only does our architecture take advantage of the MSER features of road images, it also refines MSER regions so that MSER fits better with Hough transform. In this paper, lane markings and pavement markings are used interchangeably to designate line and pictogram markings. Line markings are used to indicate a well-defined category of lane markings, i.e., solid or dashed lines that are yellow or white. On the other hand, pictogram markings mean ideograms which convey messages as understandable graphics to drivers, for example High-Occupancy Vehicle (HOV) lanes, or as pictures.

Our system consists of the following stages: preprocessing, detection, recognition, and tracking using Kalman Filter. This framework is distinguished from our predecessors by the following: (1) The computation of the detection stage is carried out using texture information, which is generated by MSER. In contrast to traditional computing methods, MSER feeds more effective and stable lane marking information to Hough transform by exclusively recognizing stable extremal regions. A three-stage refinement algorithm is then introduced to enhance the results of MSER and to filter out undesirable information such as trees and vehicles. We begin by computing the Minimum Bounding Rectangle (MBR) of all MSER regions. We then use a heuristic based on the dimensions of MBRs to filter out non-line-marking blobs. Finally, we use a scanning method to localize lane marking edges. (2) To achieve the requirements of real-time systems, the Progressive Probabilistic Hough Transform (PPHT) is used in the detection stage. Compared to Hough transform, which returns the parameters ρ and θ, PPHT returns two end-points of the detected line-markings. (3) The recognition of line markings is performed in HSV color space to better distinguish between white and yellow colors; it is based on the result of the application of MSER to left and right markings. The recognition of HOV signs is based on the results of MSER regions. We develop an algorithm that

* Corresponding author. Tel.: +16132162452.
E-mail addresses: amammeri@uottawa.ca (A. Mammeri), boukerch@site.uottawa.ca (A. Boukerche), ztang082@uottawa.ca (Z. Tang).

http://dx.doi.org/10.1016/j.comcom.2015.08.010
0140-3664/© 2015 Elsevier B.V. All rights reserved.

filters out unwanted objects and retains HOV signs. (4) We have used Kalman filter in the tracking stage to track both ends of each detected line-marking instead of tracking the parameters (ρ, θ) of Hough transform as done in many research papers, e.g., [4]. The main reason of using Kalman filter to track both ends of each detected line-marking is that it has much lower computational requirements, which fits perfectly with the real-time constraints of our system.

Our paper is structured as follows. In Section 2, we summarize and reference the related works for our research. In Section 3, we describe the preprocessing stage used by our System. We then refine in Section 4 the results of the last stage using our refinement procedure. Next, we present in Section 5 the detection algorithm adopted by our system. Then, in Sections 6 and 7 we described the recognition strategy and the tracking filter used by our system. Several experiments are conducted in Section 8, and we conclude our paper and present our future work in Section 9.

2. Related work

To detect and track pavement markings, most lane detection and tracking systems adopt the following architecture: preprocessing, detection, and tracking [5–7]. In this section, we review some important papers regarding these stages.

Pre-processing. The purpose of this stage is to enhance input frames in order to increase the likelihood of the successful delivery of areas with useful information to subsequent stages [6]. In other words, it allows the extraction of the region of interest (ROI) that contains pavement markings, in order to reduce the computational cost. To get the ROI, three main approaches are used: vanishing point detection, perspective analysis and projective model, and sub-sampling. Vanishing point detection technique is used to determine the ROI in many papers, such as in [2,8,9]. However, this has some strict limitations related to the vanishing points in the image, i.e., straight lanes with constant vanishing points. The second approach, i.e., perspective analysis and projective model, is based on the fact that parallel lane markings in the real world plane intersect at a vanishing point in the image plane. Usually, by analyzing the perspective effect, the detection range can be focused on a certain area, which can be ROI. With a reasonable projection applied between the image plane, the real world plane and the camera plane, the ROI can be extracted. For instance, RALPH (Rapidly Adapting Lateral Position Handler [10]) constructed a very basic projection model to obtain ROI. In [11], a projection model is constructed based on a 2D lane geometric model, which also helps estimate lane model parameters and lane model matching for the refinement stage. Sub-sampling is the third approach used to determine the ROI, as performed in [12,13], in which a predefined or adaptive percentage of the image can be used to determine the size of ROI.

After the generation of ROI, inverse perspective mapping (IPM) is usually deployed on the extracted area. IPM is used to transform an image from a real world plane to a birds-eye-view in order to obtain the desired line candidates straight and parallel, (e.g., [6,14–16]). Moreover, unwanted regions are removed because the remapped image focuses only on the road surface. Segmentation techniques are also used to enhance edges of lane markings and remove excess undesirable blobs. To prepare images for the detection stage, segmentation is used to extract certain features from the input image. Color [17], blobs [18] and edge [19] are three main features which are considered for lane detection segmentation.

Detection. After preprocessing, detection is used to extract lane markings from the ROI using feature extraction methods and refinement approaches. Three main feature extraction approaches are found in the literature: edge-based methods, color-based methods and hybrid (edge and color) methods.

Edge-based methods: The Hough transform is the most commonly technique used to detect lines [19]. However, two drawbacks are reported: high false positive rate and computational complexity [20]. To cope with the high false positive rate, probabilistic Hough transform [17,19] and adaptive random Hough transform [6,7] are employed. Apart from Hough transform and its variants, another edge-based method based on Steerable filter is applied in many research papers such as in [5,16,21,22]. This method yields good results when road markings are clearly painted and consistently smooth. However, Steerable filter does not adapt to heavy traffic where the orientation of lane markings are not always dominant in all directions.

Color-based methods: Unlike edge-based methods, color-based methods are not widely used by researchers, because they are influenced by lightning. The authors in [23] use a color-based method in the HIS color space by computing the cylindrical distribution of color features.

Hybrid methods: Color and shape information has been used to overcome the drawbacks of color-based and edge-based methods. These types of techniques usually combine width, length, and location of lines with gray levels and brightness values of pixels, which improves the extraction results [24].

Tracking. To facilitate following of line markings over time, a tracking stage is usually incorporated. The aim of the tracking stage is the prediction of future line marking positions in the image, and the decrease of false detection. The most common trackers used in lane tracking systems are Kalman and Particle filters. The parameters of Hough transform have previously been tracked using variants of Kalman Filter, such as Extended Kalman Filter used in [25].

3. Pre-processing

Based on the detection scheme proposed in [26], a line marking localization, recognition and tracking system is proposed in this paper. Our system is composed of a pre-processing stage based on MSER computation (Section 3), a detection stage using an improved version of Hough transform (Progressive Probabilistic Hough Transform, outlined in Section 5), a recognition stage that identifies lines and pictogram markings (Section 6) using computational geometry, and a tracking stage using Kalman filter (Section 7).

The purpose of the pre-processing stage is to generate a binarized picture, which contains the desired line information while effectively reducing unwanted information. To the best of our knowledge, the most common method for solving the above problem is to use segmentation based on edge and area information. The key in selecting suitable segmentation methods for line marking detection is to retain line marking pixels while weakening unwanted pixels. Many different masks (e.g., Canny, Sobel or Prewitt) are used to be convolved with gray images. Only a few of them use blob-extraction-based methods (e.g., MSER) to extract desired pixels (e.g. [26]). To select the best segmentation method, comparative experiments have been conducted in [26], focusing on MSER segmentation and edge segmentation.

3.1. Edge-based segmentation

Edge-based segmentation is usually performed on the ROI in gray scale to enhance edges and to obtain pixels that belong to the desired lane markings. Region of interest is always mandatory for edge-based segmentation. Different methods have been used to extract the ROI from the target frame, as performed in [3]. In fact, input images contain lane markings and some unwanted objects such as electrical poles, pedestrians, trees and cars. In order to reduce undesirable objects which might affect the system results, the detection area should be focused only on the road surface.

After obtaining ROI, some classical segmentation methods (for example Sobel, Prewitt, Robert or Canny filters) can be used to detect edge information. It is well known that Canny's operator outperforms other operators such as Sobel in the general edge detection problem. However, the authors in [27] have revealed that, for the purpose of lane detection, Canny filter is very sensitive to irrelevant objects as well as lane markings, which rapidly increases the number of false positives. The Canny operator is also computationally expensive compared to other edge detection algorithms. Conversely, the Sobel operator is less sensitive to noise and less complex than Canny, and it is able to detect the main markings. Hence Sobel is selected as the edge segmentation method to compete with MSER algorithm in [26].

3.2. MSER-based segmentation

One effective area-based segmentation method is Maximally Stable Extremal Region (MSER), which was proposed in [28] and has rarely been used in lane detection. In [18], Sun et al. proposed a method which consists of describing MSER patches using SIFT-based descriptors, followed by a graphical model to localize lane markings. In contrast to [18], which involves an unsupervised learning algorithm and off-line training, our proposed system directly takes advantage of MSER blobs to extract the features of lane markings (lines and pictograms). In addition, our proposed method uses MSER results to set the ROI for detection stage, allows the improvement of MSER and Hough transform results, and increases the detection rate of lane markings.

3.2.1. Background on MSER

For a gray image I which can be described as a mapping: $(x, y) \in Z^2 \to L$, where Z^2 represents a set of pixels with coordinates (x, y), and L represents a set of luminance of pixels ranges that vary from 0 to 255. The term **region** we used here represents a contiguous subset S of the space Z^2 (specifically for 4-neighbourhoods) which satisfied:

$\forall\, p, q \in S;\ p, q \neq \emptyset;\ \exists$ series $\{p, a_1, a_2, a_3, \dots, a_{i-1}, a_i, q\}$, where $a_1, a_2, a_3, \dots, a_{i-1}, a_i \in S$, $s.t. |p - a_1| = 1,\ |a_i - q| = 1,\ \sum_{j=2}^{i} |a_j - a_{j-1}| = i - 1$.

A region S is called an **extremal region** when an arbitrary element in the region satisfies the mapping rule $S \to m \leq l$; where $m, l \in L$, m represents the mapped value in L of an arbitrary element in S, and l is a pre-defined threshold which ranges in [0; 255]. A **stable extremal region** is an extremal region S that does not change a lot while varies.

Let:

$$R(S_l) = \{S_l, S_{l+1}, S_{l+2}, \dots, S_{l+\Delta-1}, S_{l+\Delta}\} \qquad (1)$$

be a branch of trees rooted in S_l and satisfied: $S_l \subset S_{l+1} \subset S_{l+2} \subset \cdots \subset S_{l+\Delta-1} \subset S_{l+\Delta}$.

In order to measure the stability of different extremal region, we use the following equation (as proposed in [28]):

$$q(l) = \frac{card(S_{l+\Delta} - S_l)}{card(S_l)} \qquad (2)$$

where $card(S_l)$ represents the cardinality of a set S (one extremal region). An extremal region S_l can be chosen as a stable extremal region only in case when $q(l)$ of S_l is in the lower level among the entire extremal regions. For certain $\Delta \in L$, **Maximally Stable Extremal Region** can be obtained by choosing the stable extremal region with the smallest $q(l)$ of all stable extremal regions.

As stated above, researchers have employed various methods to attempt to extract the edge information of ROI. Additionally, they tend to use smooth methods (Gaussian Filter, Median Filters, etc.) before edge detection to remove unwanted information, blur the difference inside the region and keep regions with stable luminance. This can result in the omission of details, especially in the edges of regions where luminance changes rapidly. To balance between keeping desired details and removing annoying information, we use MSER for the pre-processing stage. Compared to edge-based segmentation, the biggest advantage of MSER is that it only recognizes the stable extremal region (e.g., lane markings, traffic signs or dark parts of cars), and successfully ignores unpredicted undesirable regions (e.g., potholes and obstacles on the road). A fact that must be noted is that MSER is however more computationally expensive than edge-based detectors. Moreover, sometimes, MSER-blobs contain unwanted details as well as desired pixels (as shown in Figs. 4 and 5). This makes it necessary to refine the results of MSER. In order to improve the time efficiency of the entire system, and based on the scanning method proposed in [26], a novel refinement scheme is proposed in Section 4.

4. Refinement of MSER results

Recall that the recognition system proposed in this paper does not only recognize the line markings, but also recognizes pictogram markings (HOV painted on road surface). To keep the details of line and pictogram markings while eliminating annoying data, three stages are used:

1. Finding the Minimum Bounding Rectangle (MBR) of all MSER blobs within input frames;
2. Using the length-width ratio of each MBR to filter out non-line-marking blobs;
3. Using the scanning method shown in Algorithm [26] to locate line marking edges.

Stages 1 and 2 aim at selecting blobs with similar shapes to line marking blobs (based on the length-width ratio). Stage 3 is used to reduce the input pixels in the next detection stage by creating a one-pixel-width edge for line marking blobs.

4.1. Finding Minimum Bounding Rectangles

Minimum Bounding Rectangle (MBR) is the 2-dimensional form of Minimum Bounding Box, and is defined as the smallest rectangle that contains every point within a given blob [29]. MBR is used in our work to envelop wanted regions (blobs of line and pictogram markings) located inside lanes, as well as unwanted blobs outside lanes. Unwanted blobs are then filter out. Specifically, according to the width-length of each rectangle, we can select appropriate blobs. As a result, rectangles of the minimum area enclosing each input (blobs) must be produced. Figs. 1 and 2 demonstrate the process of generating an MBR. First, for each blob generated previously (Fig. 1), we draw a bounding rectangle which envelops the current blob, and shrink its boundaries until they meet the edge of the contour of the blob (as shown in Fig. 1); then, we record the new rectangle (red rectangle in Fig. 2). Second, we rotate the bounding rectangle by an angle of 10 degrees, and shrink again following the same procedure performed previously. We repeat this process for each angle, increasing by 10 degrees; in this way, we cover the entire 360 degree plan. From the 36 bounding rectangles that have been generated, we select the smallest one that contains every point in the blob, and we eliminate rectangles that do not contain all blobs pixels (which might be smaller than the blob and does not contain all of its pixels). This Algorithm is performed for all blobs found in the previous step.

4.2. Length-width ratio

After MBR-stage, some rectangles enveloping blobs are generated. The length-width ratio of each MBR is then calculated. It is commonly known that line markings are usually much slimmer than other objects (as shown in Fig. 3). In other words, it is easier to extract road lines from blobs, which have an exclusively large length-width ratio. Empirically, we found that the length of a line marking rectangle is usually more than twice the length of its width. This can be used to differentiate the potential line blobs from other objects. During our experiments, we notice that some other slim objects (e.g. trees and

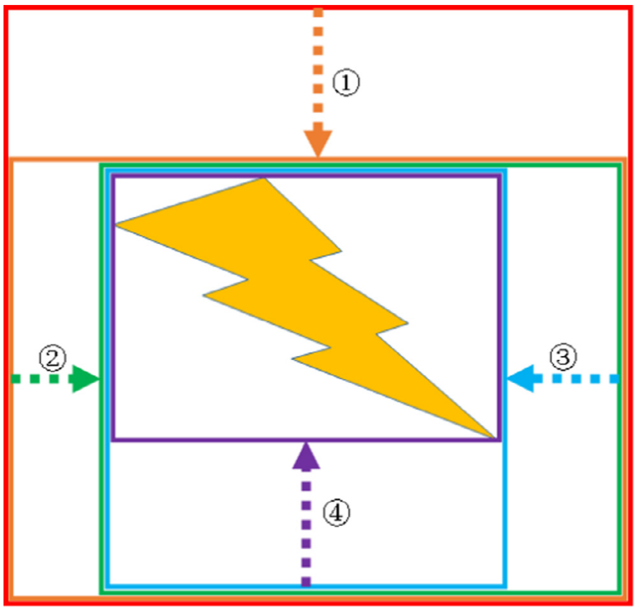

Fig. 1. The shrinking process used to generate a Minimum Bounding Rectangle: step 1 begins at the top of a rectangular box and continues until it reaches an edge point of the closed region; steps 2 and 3 start from the left and right sides, respectively, until edge points are met; step 4 begins at the bottom and stops when edge points are reached.

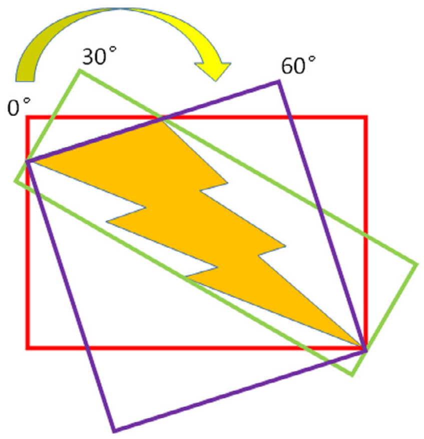

Fig. 2. Shrunken rectangles for every rotation. The rotation increment is 10 degrees.

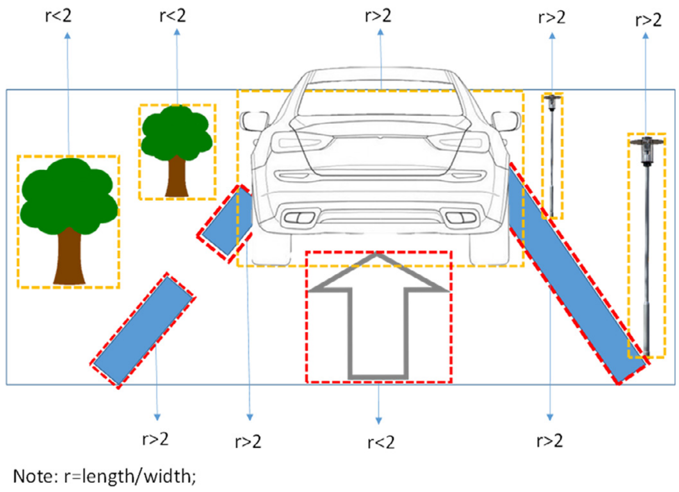

Fig. 3. The length of a line marking rectangle should be more than twice the width.

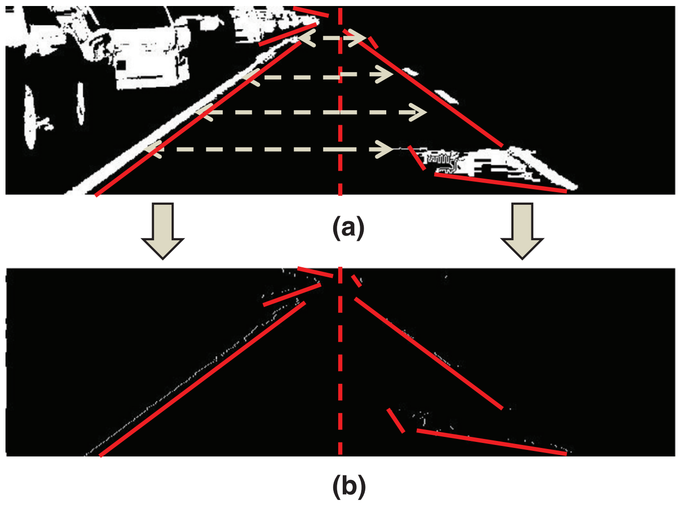

Fig. 4. The scanning method. Red dash line in (Top) and (Bottom) in the middle indicates the “middle column” of the ROI, while dash arrows in (Top) indicate the scanning direction of left and right areas divided by middle column. For every row of each area, the scanning process stops when the first white pixel is reached by the arrow. Red solid lines depicts the entire contour after refinement. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

electrical poles in Fig. 3) can be erroneously extracted as line marking blobs. To address this, a scanning method called MSER Refinement is applied afterwards to exclude those slim outliers.

4.3. MSER refinement

Empirically, we find that MSER-blobs contain considerable annoying details as well as desired pixels, as shown in Fig. 4. This makes it necessary to refine the results of MSER segmentation stage. Based on the fact that the number of objects located between the left and right line markings are lower than those outside the line markings (as shown in Fig. 4), it is reasonable to say that line marking blobs are located near the middle column (as the red dash line in Fig. 4) compared to other blobs located outside lane boundaries. As a usual fact, areas between line markings are mainly road surface, which has very weak luminance compared to other objects in gray scale images. Since MSER only extracts the stable extremal region, undesirable points within left and right line markings can be eliminated from MSER-blobs. This is different from edge detection that extracts features of both stable extremal regions and unwanted regions.

Hence, we propose a scanning method for the binarized picture as described in Algorithm 1. Starting from the middle pixel of each row, scanning is performed in left and right directions, respectively. MSER blobs are drawn in white, while the non-MSER area is kept black. The scanning of each image row stops when we find the first white pixel in left and right areas, respectively. The output of the proposed scanning method is MSER blobs shrunken into pieces of lines, which are actually partial contours of MSER blobs (as shown as Fig. 4). More importantly, because the scanning process starts from the middle column, these contours only belong to blobs that are close to the middle column in left and right areas. To the most extent, this method actually depicts the contours of line-marking-blobs which are near to middle column. Moreover, the proposed scanning rule results in selected contours that are one pixel in width, which weakens annoying blobs and makes long lines more easily recognized by Hough transform. However, the proposed scanning method has two drawbacks.

First, even though annoying points within areas between left and right line markings can hardly form MSER-blobs, real scenarios occasionally have MSER-blobs located between left and right line markings (coming from cars or an area of erosion on the road, as shown

Algorithm 1: Scanning Refinement of MSER.

```
1  Input: Binarized images with MSER blobs
2  Output: Refined contours of MSER blobs
3  x and y: coordinates of a pixel point (x, y) in the binarized image
4  width and height: the width and height of binarized image
5  P(x, y): pixel value of the point (x, y)

6  if Scanning for left area then
7      for y = 0 to height do
8          for x = $\frac{width}{2}$ to 0 do
9              if $P(x, y)! = 0$ then
10                 $x - -$;
11                 continue;
12             else
13                 $y + +$;
14                 break;
15 else
16     for y = 0 to height do
17         for x = $\frac{width}{2} + 1$ to width do
18             if $P(x, y)! = 0$ then
19                 $x + +$;
20                 continue;
21             else
22                 $y + +$;
23                 break;
```

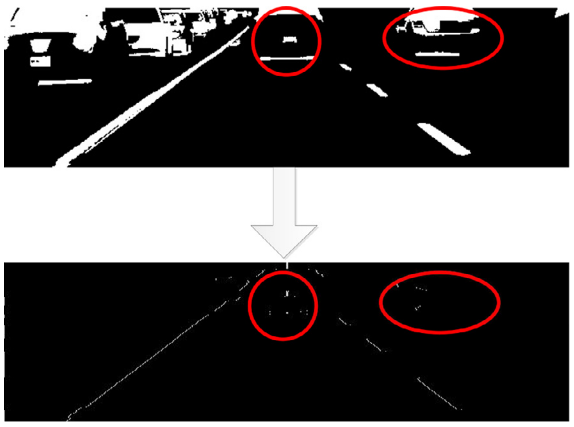
Fig. 5. Drawbacks of the proposed scanning rule: blobs between line markings and outside the current lane.

as red circular area in Fig. 5). Our proposed scanning method might inevitably take the contour of those annoying blobs as line marking candidates, and then feed those pixels together with real line marking pixels to PPHT. To eliminate unwanted blobs, as shown in Fig. 4, we proceed as follows. We know that the scanning method only selects at most two pixels in a row (one pixel per area), which results in selected line candidates that are only one pixel in width. This dramatically weakens the contour of annoying blobs between line markings, and makes the continuous contours of line marking blobs more prominent. On the other hand, PPHT can further remove the contours of MSER-blobs by thresholding length and angle of detected line segments (see Section 5).

The second drawback of the proposed scanning method is that, for dashed lines, blobs outside line boundaries (described as a red ellipse area in Fig. 5) have white pixels in rows between dashes. This might bring annoying contours for those rows. Experimentally, we find that PPHT can handle the above issues by thresholding the length and angles of line candidates, as shown by Eq. 4 and described in further detail at the end of Section 5. That is, by an appropriate thresholding, lines located in irrelevant regions can hardly be selected as line markings.

5. Line markings detection using PPHT

After the refinement stage, binary edge maps are produced with refined one-pixel-width line candidates. PPHT is used as the line detection technique in this stage. Hough transform (HT) was proposed in [30] and is usually used to detect lines and circles; it has been used as the core method of lane marking detection in [3] and [31]. The core formula of HT is

$$\lambda = x\cos(\theta) + y\sin(\theta) \tag{3}$$

λ is the length between the origin and the pedal of detected line and θ is the angle of its perpendicular line.

In [32], Matas et al. proposed the PPHT, which has been commonly accepted as one of the best line detection methods based on Hought transform theory. The algorithm PPHT proceeds as follows:

1. Randomly, select a new point for voting in the accumulator array, with contributions to all available bins (as referenced in [32], bin stands for a pair of (λ, θ)). Then remove the selected pixel from the input image.
2. Check if the highest peak (the pair of (λ, θ) with the most voting points) in the updated accumulator is greater than a pre-defined threshold $th(N)$. If not then go to Step 1.
3. Find all lines with the parameter (λ, θ) which was specified by the peak in Step 2. Choose the longest segment (which can be denoted by starting point Pt_0 and ending point Pt_1) of all lines.
4. Remove all the points of the longest line from the input image.
5. Remove all the points of the selected line in Step 3 $(Pt_0 - Pt_1)$ from the accumulator, which means those points do not attend any other voting process.
6. If the selected segment is longer than a pre-defined minimum length, then take the segment $(Pt_0 - Pt_1)$ as one of the output results.
7. Go to Step 1.

As we can see from Eq. (3), HT returns (λ, θ) of every detected line and takes all straight lines into account, which makes it time consuming and too sensitive to straight lines (regardless of some undesirable lines with short length) and generates undesired marking candidates. In this paper, PPHT is employed instead of HT in order to cope with the above problem and to minimize computation costs. Hough transform also consumes much more time than PPHT in order to present all lines. From our experiments, we have realized that HT usually returns 10–20 lines while PPHT returns 5–10 as shown in Fig. 6. Approximately, we can say that PPHT can save around half computation cost of HT (Fig. 6). As explained in [32], PPHT improves the process of HT by minimizing the number of voting pixels.

In addition, lane detection has its own requirements; the detector should only respond to lines with specific characteristics (with the same shape as lane markings). Recalling the two drawbacks in Section 4.3 (as shown in Fig. 5), sometimes vehicles or contours of surroundings appearing in the region of interest can be erroneously detected by HT. These lines not only bring a variety of directions but also have shorter lengths compared to real lane markings, hence are not eligible to be chosen as lane marking candidates. In PPHT, constraints can be given by setting a minimum line length, which only takes lines with qualified length as output. Besides, we can remove

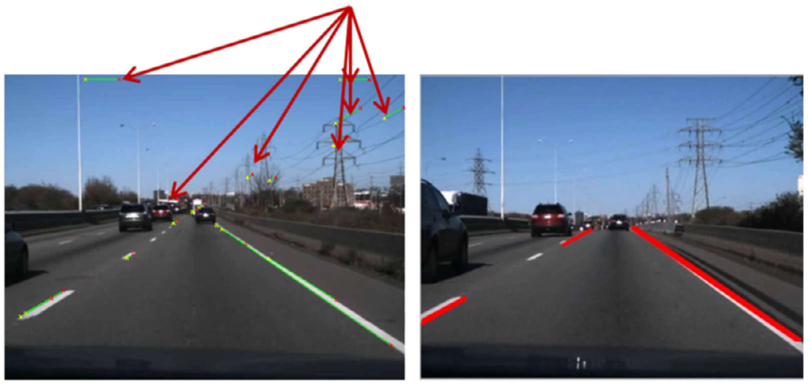

Fig. 6. Left: the detection of lane markings using HT. Note the detection of noisy line information; Right: the detection of lane markings using PPHT.

lines with annoying angles by applying the following constraints:

$$T_1 \le \left| \frac{y_1 - y_0}{x_1 - x_0} \right| \le T_2 \tag{4}$$

where T_1 and T_2 are the pre-defined thresholds set by users to define the angle values, and (x_0, y_0) and (x_1, y_1) are the coordinates of end-points Pt_0 and Pt_1, respectively. Considering the height of our dashboard-camera from the ground and its resolution 640×480, the threshold range is defined experimentally, and it is set to 0.2 for T_1 and 10 T_2. These values mean that the angel between lines and the driving orientation should be set approximately between 10 and 85 degrees.

6. Lane marking recognition

Road surface markings are used to provide appropriate information to drivers and pedestrians, for instance lines and pictograms. A pictogram marking is a well-designed ideogram which conveys a message as an understandable graphic, for example HOV signs. Line markings separate between lanes, and are used to manage traffic by guiding vehicles to avoid collisions. This is achieved by exploiting both the color (white or yellow) and the form (solid or dashed) of the line markings. For example, a yellow color of a line infers that this line is used to separate a road into two lanes of opposite directions. In this system, we propose an algorithm that recognizes and distinguishes between line markings' forms and colors. Based on the results of Section 5, we proposed a line recognition method which recognizes line markings with different colors and forms. A parallelogram must be constructed based on the starting and ending points of PPHT (as shown in Fig. 7), for left and right line markings, respectively. Edge points of lane markings can be included by the parallelogram.

6.1. Yellow and white lines

It is commonly known that yellow and white line markings are sometimes confusing. This is due to the irregular textures of road surfaces, environmental conditions (such as sunlight), daytime and other complex factors. Thus, the boundary between yellow and white is not always easily defined, particularly when using RGB color space. In our context, the problem lies mostly in how to distinguish yellow and white for pavement markings, even they do not clearly contrast with each other in real scenarios. In fact, the selection of the appropriate color space is of high importance, because it affects the detection results. Empirically, we found that line markings are painted in colors which are more distinguishable in HSV than in RGB color space. That is, in our work, each frame is first converted from RGB to HSV color space. Specifically, R, G, and B are converted to the floating-point format and scaled to fit the 0 to 1 range. For each point candidate in the parallelogram range in Fig. 7(b), its RGB value can be obtained from original frames taken by cameras. With the following equations, RGB value can be converted into HSV value:

$$V = max(R, G, B); \tag{5}$$

$$S = \begin{cases} \dfrac{V - min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$H = \begin{cases} \dfrac{60(G - B)}{V - min(R, G, B)} & \text{if } V = R \\ \dfrac{120 + 60(B - R)}{V - min(R, G, B)} & \text{if } V = G \\ \dfrac{240 + 60(R - G)}{V - min(R, G, B)} & \text{if } V = B \end{cases} \tag{7}$$

if $H < 0$, then $H = H + 360$. The final range of (H, S, V) is

$0 \le V \le 1; 0 \le S \le 1; 0 \le H \le 360$

For a pixel point, if

$$\begin{aligned} S &> 0.2 \\ &\text{and} \\ V &> 0.4 \end{aligned} \tag{8}$$

this pixel can be deemed as yellow, otherwise it is white.

6.2. Solid and dashed lines

Recognizing the form (solid or dashed) of a lane marking is a challenging task for driver assistance systems. The recognition of such markings depends on the vehicle's speed, the condition of the pavement, the surrounding environment, and time (night and day). An example of an unwanted scenario is when a solid line marking on an eroded surface is recognized as a dashed marking. Shadows of any object on the road can have a similar effect. Moreover, a lack of marking pixels may result in a missed detection of a line marking. In order to deal with these situations and to avoid erroneous recognitions, we develop a module that recognizes line markings. It has been experimentally proved that a parallelogram area with more than 170 marking pixels is likely to contain a solid line, otherwise it contains a dashed line. As shown in Fig. 7, seeing from the two parallelograms based on starting and ending points detected by PPHT, dashed lines and solid lines can be obviously distinguished with different numbers of marking pixels.

6.3. Pictogram marking recognition

In this paper, we develop an algorithm that recognizes HOV signs. Other pavement signs are left to our future work. HOV lanes, also

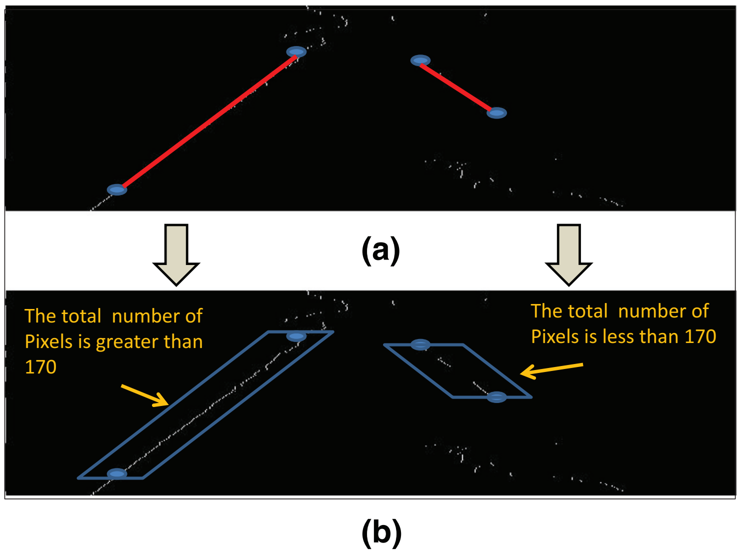
(a)
The total number of Pixels is greater than 170
The total number of Pixels is less than 170
(b)

Fig. 7. (a) Is the lane markings detected by PPHT, with starting and ending points; (b) shows the parallelogram area constructed based on the position of starting and ending points.

known as carpool lanes or diamond lanes, are traffic lanes reserved for vehicles with a specified number of occupants, at specified hours. Their aim is to decrease traffic congestion by increasing person throughput. To detect HOV signs, we proceed as follows. The previously detected and recognized line markings are used to keep MSER regions located between line markings, which contain, among others, HOV signs, and remove outside regions. First, we extend line markings to reach the bottom of the ROI. In practice, each line is a thick line constituted by a set of contiguous geometrical lines. Only those lines (noted l_r and l_f) that are closest to the middle line of the lane are kept; these are to narrow the detection range. Second, for each labeled MSER region, we verify whether its coordinates are located between l_r and l_f. This is performed by comparing the coordinates (x_{R_i}, y_{R_i}) of a randomly selected point from each MSER region R_i located between l_r and l_f to the coordinates' positions of l_r and l_f. If (x_{R_i}, y_{R_i}) is located inside the area between l_r and l_f, R_i is selected as a candidate region which possibly contains HOV sign; otherwise, it is removed. After the completion of this step, we observed some undesirable blobs. To filter them out, we have decided to adopt a heuristic approach based on the dimensions and colors of the blobs. The necessary dimensions were determined manually by observing the dimensions of blobs. Larger blobs which may represent vehicles or trucks are filtered out. As most HOV signs are white, only white blobs are kept.

In some challenging situations, more than one marking is found on the lane. In order to deal with this issue, we traverse all contour points and find out the top, bottom, leftmost, and rightmost points. Since, we have already obtained these points, the line linking the top and bottom points should be vertical to the line connecting the leftmost and rightmost points. Also, the orientation of the line linking top and bottom points should point to the vanishing point. If a given region satisfies these requirements, we regard this sign as an HOV sign.

7. Tracking of lane markings

It is well known that the addition of a lane tracking stage after lane detection increases the probability of detecting lane markings, especially in harsh conditions (e.g., rough and rural roads or rainy weather). The most common lane trackers used in the literature are Kalman and Particle filters. Kalman is used to predict the post state based on the previous state and current measurements by updating their covariance matrix. This process is then looped by feeding the corrected state to the next instance. In fact, Kalman Filter has been used to track (λ, θ) of Hough transform [26]. In this paper, Kalman filter is used to track both ends of each lane markings, because we have used PPHT in detection.

We construct two Kalman trackers for right and left lane markings. For the end-points Pt_0 and Pt_1 of a given segment, the state vector X_k is:

$$X_k = AX_{k-1} + BU_k \tag{9}$$

where A is the state updating matrix and B is the input control matrix. In this case, we define the state

$$X = [X_{Pt_0} Y_{Pt_0} X_{Pt_1} Y_{Pt_1} X'_{Pt_0} Y'_{Pt_0} X'_{Pt_1} Y'_{Pt_1}]^T \tag{10}$$

where X' and Y' are the first derivatives of X and Y, respectively. Experimentally, with the best tracking performance for our testing video, we define the state updating matrix as following:

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$U_k = 0 \tag{11}$$

We take the detected coordinates of Pt_0 and Pt_1 as measurement Z_k for every frame, where Z is given by:

$$Z = [X_{Pt_0} Y_{Pt_0} X_{Pt_1} Y_{Pt_1}]^T \tag{12}$$

A is the transition 8×8 matrix bringing state X from time $k-1$ to k. A models the evolution of the state when no input is applied. If we consider a simplified model (low velocity and high frame rate) A could be equal to I. From Eqs. 10 and 11, the estimated value $\widehat{X}_k$ of X_k

Table 1
Video clips used for experiment. (D: dashed; W: white; S: solid; Y: yellow).

	Clip #1	Clip #2	Clip #3	Clip #4	Clip #5	Clip #6
Lighting	Cloudy	Sunny	Sunny	Cloudy	Night	Night
Location	Highway	Urban	Highway	Urban	Urban	Urban
Traffic	Heavy	Light	Medium	Light to heavy	Heavy	Medium
Road surface	Rough	Smooth	Smooth	Smth-Rough	Smooth	Rough
Frame NO.	912	864	1080	960	1200	1043
Frame speed	24fps	24fps	24fps	24fps	24fps	24fps
Markings/Frame	2	1	2	2	2	2 or 1
Line type	DW	SY	DW-SW	DW-SY	DW	DW-SY

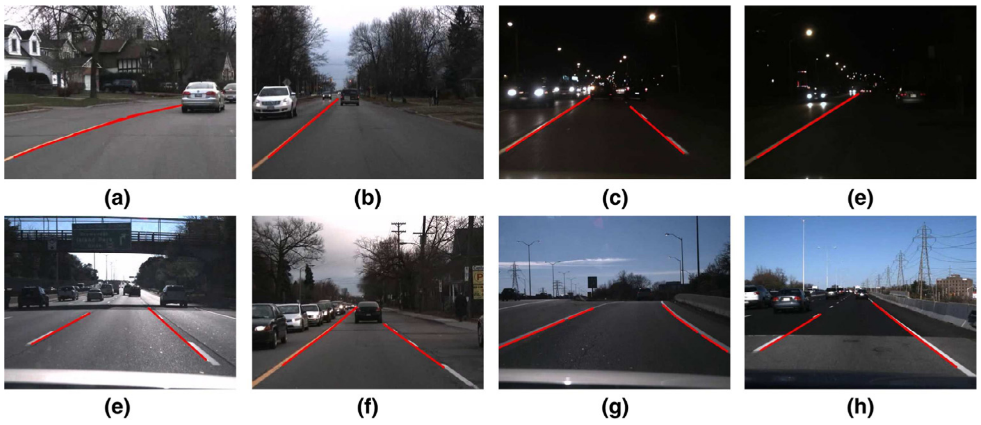

Fig. 8. Correct detection in different scenarios: (a) urban area: curvy line marking occluded with cars; (b) urban area: single line marking; (c) urban area: heavy traffic with strong lighting; (d) urban area: medium traffic; (e) urban area: medium traffic, sunny, rough and shadowy road; (f) urban area: rough road with heavy traffic, cloudy; (g) highway: slope and curvy lane; (h) highway: different road color, heavy traffic.

is given by:

$$\widehat{X}_k = \begin{pmatrix} X_{Pt_0} + 0.5X'_{Pt_0} \\ Y_{Pt_0} + 0.5Y'_{Pt_0} \\ X_{Pt_1} + 0.5X'_{Pt_1} \\ Y_{Pt_1} + 0.5Y'_{Pt_1} \\ X'_{Pt_0} \\ Y'_{Pt_0} \\ X'_{Pt_1} \\ Y'_{Pt_1} \end{pmatrix}$$

which implies the following:

$$\Rightarrow \begin{aligned} \widehat{X}_{Pt_0\,(k|k-1)} &= \widehat{X}_{Pt_0\,(k-1|k-1)} + 0.5\widehat{X}'_{Pt_0\,(k-1|k-1)} \\ \widehat{Y}_{Pt_0\,(k|k-1)} &= \widehat{Y}_{Pt_0\,(k-1|k-1)} + 0.5\widehat{Y}'_{Pt_0\,(k-1|k-1)} \\ \widehat{X}_{Pt_1\,(k|k-1)} &= \widehat{X}_{Pt_1\,(k-1|k-1)} + 0.5\widehat{X}'_{Pt_1\,(k-1|k-1)} \\ \widehat{Y}_{Pt_1\,(k|k-1)} &= \widehat{Y}_{Pt_1\,(k-1|k-1)} + 0.5\widehat{Y}'_{Pt_1\,(k-1|k-1)} \end{aligned}$$

$\widehat{X}$ and $\widehat{Y}$ represent the estimated values of X and Y, respectively. In order to meet the velocity direction between two measured points, the weights of $\widehat{X}$ and $\widehat{Y}$ should be set to the same value (0.5 in our case). Experimentally, we found that 0.5 is the best value that satisfies tracking performance of our system.

8. Experiment

Our experiment focuses on the performance evaluation of the proposed system. To test the real-time performance of our proposed system, several videos were taken from Ottawa roads, using a $FL3-U3-13S2C-CS$ camera mounted in the front of an experimental car and fixed at a height of 1.3 m above the ground. The program was implemented in C++ under Windows using the OpenCV library, with the hardware environment of Intel core i3 CPU having 2.30 GHz and 4G RAM. Of particular note is the resizing of our original images (with resolutions of 640 × 480) into 640 × 240 before the use of MSER, in order to improve time efficiency. This resizing does not influence the final experimental results.

Video clips taken from downtown and highway areas in Ottawa represent most real driving scenarios during the night and day. These videos represent some common situations with different lighting conditions (sunny, cloudy and night), traffic (heavy, medium and light), and road surface (rough and smooth), which people might encounter in real life. Some of the videos were listed in detail; see examples in Table 1.

We perform several experiments to evaluate the efficiency and accuracy of our system. The evaluation of each method is performed using metrics as explained in Sections 8.1 and 8.2. Fig. 8 shows frames with correct detection of line markings. These pictures show that our proposed system performs acceptably well in different real scenarios.

8.1. Line marking detection performance evaluation

For the purpose of detection performance evaluation, two metrics are used: False Positive Rate (FPR) and False Negative Rate (FNR). False Positive (FP) and False Negative (FN) considers the presence or absence of lane markings. FPR refers to the probability of falsely selecting a given object or contours such as vehicles, curbs, trees or electrical poles, as a lane marking. On the other hand, FNR refers to the situation when a lane marking is falsely rejected by the detection method.

Three methods are considered and implemented for comparative evaluation: method (a), method (b) and our method called (c). The method (a) uses an edge-segmentation method based on Sobel filter, followed by PPHT to detect line markings. Whereas the method (b) uses MSER segmentation without refinement, followed by PPHT. The reason behind this choice is to show the accuracy and efficiency of our

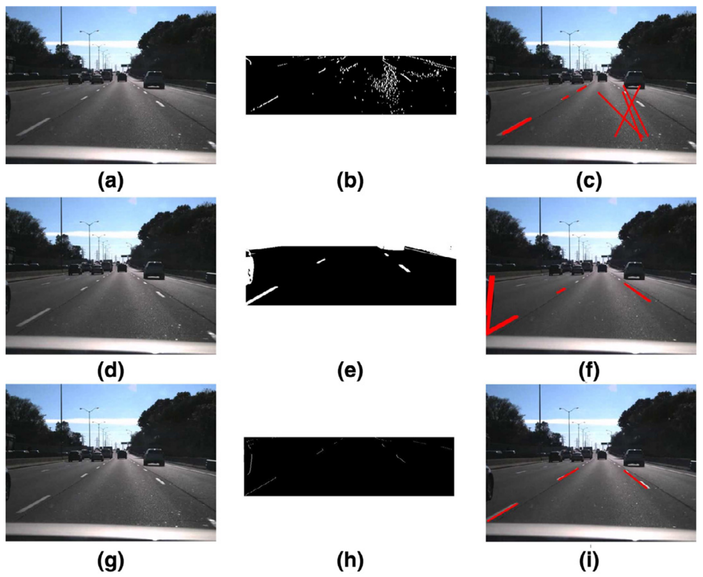

Fig. 9. Detection performance of edge-based, MSER-based and the proposed refined MSER-based segmentation methods. Images (a), (d) and (g) represent the same frame taken from Clip #1. Images (b), (e) and (h) are the segmentation results of edge-based, MSER-based and refined MSER-based methods, respectively. Images (c), (f) and (i) are the results of the application of PPHT after edge-based, MSER-based and refined MSER-based methods, respectively.

Table 2
Edge segmentation for line markings detection.

	Process-time	No. markings	TPR (%)	FPR (%)
Robert	26 ms	582	66	31
Canny	94.5 ms	788	87	44
Prewitt	48.3 ms	812	90	45
Sobel	32.3 ms	609	92	9.5

Table 3
Comparative performance evaluation in (%) for line marking detection at night.

	Clip #5			Clip #6		
	DR	FPR	FNR	DR	FPR	FNR
Met. (a)	57.8	1	42.2	64.1	0	35.9
Met. (b)	74	14.8	26	80.6	9.7	19.4
Met. (c)	77.6	7.1	22.4	83.3	4.5	16.7

method (*c*) (which refines method (*b*)) against edge-based methods, and against MSER-based segmentation without refinement.

Fig. 9 clearly shows that the refined MSER Segmentation (Fig. 9(i)) keeps desired details (line markings) and removes some details (e.g., cars, trees and curbs), when compared with edge segmentation (Fig. 9(c)) and MSER segmentation without refinement (Fig. 9(f)).

8.1.1. Comparison of different edge segmentation methods

As an experiment, we have compared the Sobel operator with other filters such as Canny, Prewitt and Robert, as shown in Table 2. Different edge segmentation methods are applied on 300 frames. After segmentation, we have used PPHT to detect line markings, without tracking and other refinement stages. There are 600 line markings to be detected for these 300 frames. Table 2 proves that Canny is unsuitable for line marking detection, with a False Positive Rate (FPR) of 44%, and consumes much more time than other methods (94.5*ms*). On the other hand, Sobel performs the best comprehensively of all four methods. This makes it reasonable to use Sobel as a representation of edge segmentation in comparison with MSER segmentation and our proposed method (as explained in Section 8.1.2)

8.1.2. Comparative evaluation

Conventionally, for experiments in line marking detection, ground truth is either hand-annotated on each frame or determined by visual inspection, as performed in [3,16,19]. Researchers usually qualitatively judge whether or not the result for each frame fits with ground truth. Different experimental results are presented in Tables 3 and 4 for night and daytime, respectively.

It is shown in Table 3 that, in spite of highest FNR and lowest DR, the edge-based segmentation (method (*a*)) has much lower FPR than the other two methods (only 1 and 0%). This is mainly because poor lighting at night makes the annoying edges less visible. Besides FPR, MSER segmentation performs significantly better than edge segmentation during nighttime, and method (*c*) refines the results of MSER segmentation.

In Table 4 we see that MSER-based segmentation (*b*) generally performs better than edge-based segmentation (*a*), especially for Clip #4, which represents very common situations in urban areas. In Clip #4, method (*b*) increases the detection rate from 77% of method (*a*) to 95.8%, while dramatically reducing the FNR from 23 to 4.2%. However, method (*b*) presents some instability when it comes to other clips (having a FPR of 30.4% and FNR of 5.1%). In response to the instability of MSER segmentation, method (*c*) effectively refined the results of method (*b*). Method (*c*) increases the detection rate by almost 10% for Clip #2 and Clip #3, compared with method (*b*). Moreover, method (*c*) decreases FPR and FNR for all four video clips.

It is noted that because of poor lighting conditions at night and unpredictable light interference of cars, the line marking detection system may perform much worse than in the daytime, which is reflected in our experimental results in Table 3. We have determined that the success of line marking detection during nighttime conditions is mostly affected by traffic density, and sometimes by lighting systems on the road.

8.2. Marking recognition performance evaluation

Similar to the comparative performance evaluation (Section 8.1), we also employed the hand-label method to qualitatively judge the correct recognition of the nature of line markings (color and form), as well as false positive (FP) and false negative (FN) results. The definition of false positive (FP) and false negative (FN) results in the

Table 4
Comparative performance evaluation in (%) for line marking detection in daytime.

	Clip #1			Clip #2			Clip #3			Clip #4		
	DR	FPR	FNR	DR	FPR	FNR	DR	FPR	FNR	DR	FPR	FNR
Met. (a)	72.7	18.7	27.3	84.3	0	15.7	80.1	7.7	19.9	77	10.7	23
Met. (b)	69.6	14.2	30.4	89.2	5.1	10.8	84.3	5.5	15.7	95.8	9.8	4.2
Met. (c)	75.3	7.4	24.7	97.9	4.2	2.1	94.9	0	5.1	100	3.6	0

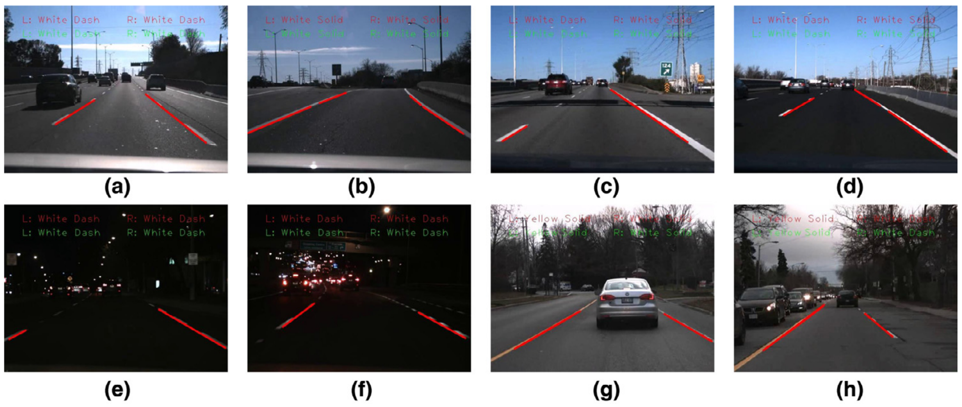

Fig. 10. Correct results in different scenarios: (a), (b) and (d) show the highway scenario with strong sunlight; (c) shows line markings occluded with shadows in highway; (e) and (f) show the nighttime situation; (g) and (h) are in urban area, with rough road surface.

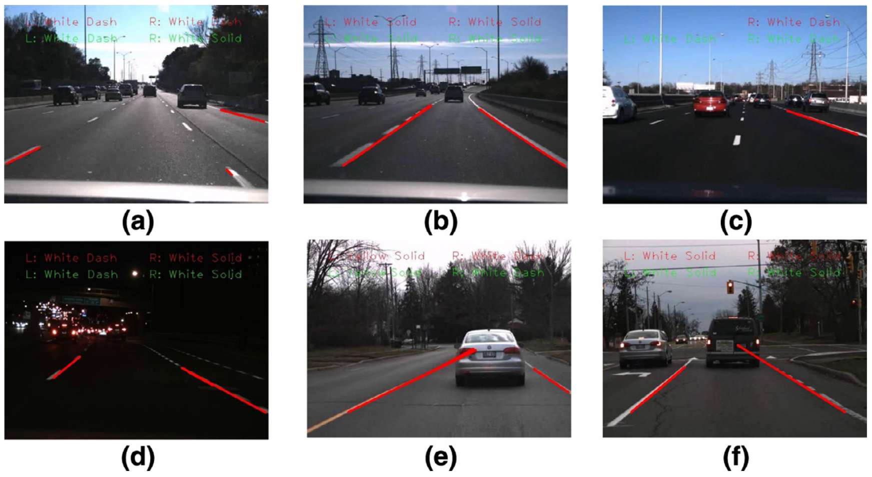

Fig. 11. False positive (FP) and false negative (FN) results in different scenarios: (a), (b, (d) and (f) show that the proposed system recognizes dashed lines as solid lines; (c) and (e) indicate a the situation in which solid lines are recognized as dashed lines.

marking recognition system excludes the line marking detection failure and is based only on correct detection results:

- A solid line being incorrectly recognized as a dashed line gives one FP score to dash recognition and one FN score to solid recognition.
- For a situation in which a dashed line is recognized as a solid line, it gives one FP score to the solid line and one FN score to the dashed line.
- Similar to solid and dashed lines, white and yellow lines being erroneously recognized as each other also give FP and FN scores in the same manner as above.

Fig. 10 shows the sample frames with correct recognition of line markings. Fig. 10(a)–(d) show different situations on a highway; Fig. 10(g) and (h) are taken from an urban area; Fig. 10(e) and (f) show a night scene in an urban area, with lighting interference. These pictures show that our proposed system scheme performs well in most real scenarios. Table 5 shows the marking recognition performance in real-time video, indicating the recognition rate (RR) and false positive rate (FPR) for the same video clips described in Table 1. As indicated in Table 5, RR stands for recognition rate, and FPR stands for false positive rate. Since we only consider the detected results of the detection stage, the denominator is the total number of detected line marking (the output of PPHT stage) when calculating the RR, FPR and FNR of each recognition item.

Fig. 11 shows that the proposed recognition scheme might inevitably fail in some challenging situations. Solid and dashed lines can be erroneously recognized as each other, as can yellow and white lines. This determines the metrics used in this part. Taking color recognition as an example, the false negative results of white should be equal to the false positives of yellow in this case, and vice versa. This is because the line markings are either solid or dashed, either white or yellow in most real scenarios. As we can see in Table 5, recognition is best achieved in a sunny highway scenario (e.g. 95.11 and 95.33% for solid and dashed lines respectively in Clip #3, 95.63% for solid lines in Clip #2); this is because highways are usually better constructed and have relatively less traffic density than urban

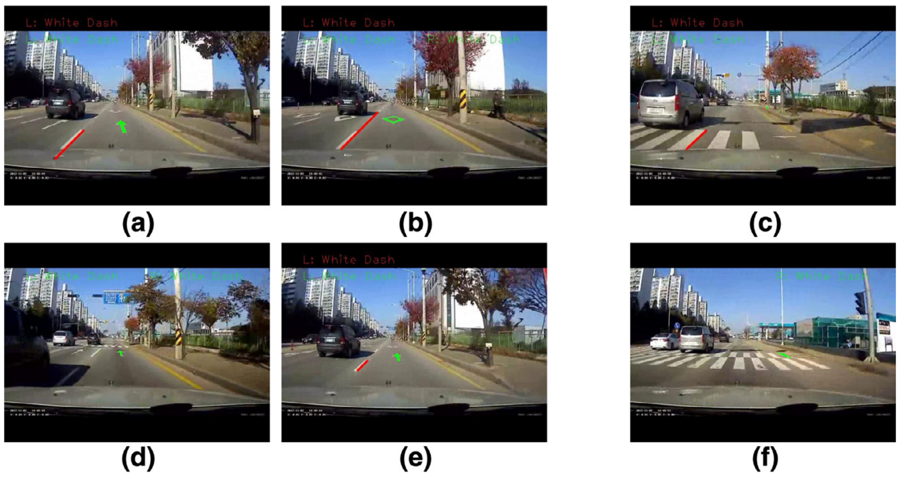

Fig. 12. Line markings are marked as red lines, while road signs are marked as green areas. Diamonds can be successfully extracted from the road surface. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 5
Hand-label comparison for line markings recognition. For video clips without one line-type (solid, dashed, yellow or white), the two evaluation metrics for this line-type should be indicated as N/A (not applicable).

	Clip #1		Clip #2		Clip #3		Clip #4	
	RR	FPR	RR	FPR	RR	FPR	RR	FPR
White	100.00	22.99	100.00	N/A	100.00	N/A	100.00	15.66
Yellow	77.01	0	N/A	0	N/A	0	84.34	0
Dash	94.60	41.85	95.63	9.61	95.11	4.67	96.88	5.24
Solid	88.15	5.40	90.39	4.37	95.33	4.89	94.76	3.12

Table 6
Testing videos for HOV recognition.

	Clip #7	Clip #8
Lightening	Sunny	Sunny and shadowy
Location	Urban	Urban
Traffic condition	Light	Medium
Road Surface	Flatly smooth	Smooth to rough
Frame NO.	704	642
Frame speed	24fps	24fps
Number of HOVs	4	3

Table 7
Comparative evaluation on HOV recognition performance.

	Clip #7			Clip #8		
	RR (%)	FPR (%)	FNR (%)	RR (%)	FPR (%)	FNR (%)
HOV	89.85	2.73	10.15	65.85	4.88	34.15

roads. In particular, when considering line marking forms, dashed lines are more easily detected with the proposed method based on MSER blobs; this leads to a relatively high RR and FPR.

In the differentiation of color, using a segmentation method based on HSV space, white line recognition is surprisingly stable for all scenarios (with 100% RR for all four videos). However, yellow lines can be recognized as white lines because of the weak contrast with the background, especially in poor lighting conditions. This leads to the recognition that white lines overwhelm yellow lines in Clip #4 (with the RR of 100 and 84.4%, respectively, for white and yellow), which is characterized by cloudy weather and rough road surface.

8.3. Road sign recognition performance evaluation

In this paper, only a fraction of road signs (diamonds) are discussed and recognized in the real-time videos we have obtained from the Internet. The description of video clips that have been used are summarized as Table 6.

Similar experimental methods to marking recognition have been used to evaluate the road sign recognition performance. As shown in Fig. 12, diamonds painted on the road surface can be successfully extracted (as in Fig. 12 (a), (b), (d) and (e)), while in challenging situations such as cross-roads and vehicles, other objects could be recognized as road signs (as shown in Fig. 12 (c) and (f)).

Table 7 shows the performance of our proposed HOV recognition scheme. Because of the proposed refinement strategy, the FPR is quite low as we expected, while, however, missing recognition makes the FNR high. Especially for Clip #8, where shadow interference frequently affects the recognition results, 34.15% of true HOV signs are missing out. The FPR is due mainly to other pictogram signs such as arrows cross-walk which can be recognized as HOV signs. Several factors may affect the recognition performance of HOV signs. These factors vary from the environmental conditions such as snow, sun, the speed of the car, to the quality of signs painting.

9. Conclusion

In this paper, we presented a real-time framework able to detect, an track lanes, and recognize the nature of their marking (i.e., lines and pictograms). To localize and detect the ROI, Maximally Stable Extremal Region (MSER) technique was investigated. Then, a three-stage refinement algorithm was then introduced to enhance the results of MSER step. First, we computed the Minimum Bounding Rectangle of all MSER blobs. Next, we used some properties of MBRs to filter out undesirable blobs. Finally, to localize lane marking edges, we used a scanning method. To achieve the real-time requirements of our system, a variant of Hough transform known as the

Progressive Probabilistic Hough Transform (PPHT) was used in the detection stage to detect line markings. After detection, the recognition of left and right line markings (colors and forms) is performed in the HSV color space. To recognize HOV signs, we developed new algorithm based on the results of MSER algorithm. At the end, Kalman filter is used to track both ends of each detected line marking. Several experiments are conducted to show the efficiency of our system.

Acknowledgment

This work was partially supported by Canada Research Chair Programs, DIVA Strategic Research Network, and Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] A. Xiangjing, S. Erke, Song Jinze, L. Jian, H. Hangen, Real-time lane departure warning system based on a single FPGA, EURASIP J. Image Video Process. 2013 (1) (2013) 1–18.
[2] H. David, M. Majid, Detection of lane departure on high-speed roads, Proceedings of International Conference on Pattern Recognition Applications and Methods (2012) 529–536.
[3] D.O. Cualain, C. Hughes, M. Glavin, E. Jones, Automotive standards-grade lane departure warning system, IET Intell. Transp. Syst. 6 (1) (2012) 44–57.
[4] V. Vincent, A. Manuel, E. Bruno, B. Stephane, B. Jean-Christophe, Road markings detection and tracking using hough transform and Kalman filter, Proceedings of International Conference for Advanced Concepts for Intelligent Vision Systems (2005) 76–83.
[5] R.K. Satzoda, M.M. Trivedi, Drive analysis using vehicle dynamics and vision-based lane semantics, IEEE Trans. Intell. Transp. Syst. 16 (1) (2015) 9–18.
[6] Q. Li, L. Chen, M. Li, S.L. Shaw, A. Nuchter, A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios, IEEE Trans. Veh. Technol. 63 (2) (2014) 540–555.
[7] L. Xiangyang, F. Xiangzhong, W. Ci, Z. Wei, Lane detection; parallel-snake; planar homography; vanishing point estimation, J. Intell. Robot. Syst. 77 (2015) 597–609.
[8] N. Marcos, A.L. Jon, S. Luis, Road environment modeling using robust perspective analysis and recursive Bayesian segmentation, Mach. Vision Appl. 22 (6) (2011) 927–945.
[9] W. Yifei, D. Naim, A. Alin, A novel system for robust lane detection and tracking, Signal Process. 92 (2) (2012) 319–334.
[10] D. Pomerleau, RALPH: rapidly adapting lateral position handler, in: Proceedings of the Intelligent Vehicles '95 Symposium, 1995, pp. 506–511.
[11] S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, H. Chen, A novel lane detection based on geometrical model and Gabor filter, in: IEEE Intelligent Vehicles Symposium (IV), 2010, La Jolla, CA, USA, 2010, pp. 59–64.
[12] S. Stephan, K. Sarath, A. Alen, D. Gamini, Efficient lane detection and tracking in urban environments, in: Proceedings of the 3rd European Conference on Mobile Robots, 2007.
[13] A. Shihavuddin, K. Ahmed, A.K. Rashed, Road boundary detection by a remote vehicle using radon transform for path map generation of an unknown area, Int. J. Comput. Sci. Netw. Secur. 8 (8) (2008) 64–69.
[14] D. Jiayong, H. Youngjoon, A real-time system of lane detection and tracking based on optimized RANSAC B-spline fitting, in: Proceedings of the 2013 Research in Adaptive and Convergent Systems, 2013, pp. 157–164.
[15] A. Mohamed, Real time detection of lane markers in urban streets, IEEE Intell. Veh. Symp. 712 (2008).
[16] S. Sayanan, T. Mohan, Integrated lane and vehicle detection, localization, and tracking: a synergistic approach, IEEE Trans. Intell. Transp. Syst. 14 (2) (2013) 906–917.
[17] W. Qinghua, Y. Zehong, S. Yixu, J. Peifa, Road boundary detection in complex urban environment based on low-resolution vision, in: Proceedings of 11th Joint International Conference on Information Sciences, 2008.
[18] H. Sun, C. Wang, N. El-Sheimy, Multi-platform/multi-sensor remote sensing and mapping (M2RSM), Proceedings of International Workshop on Automatic Traffic Lane Detection for Mobile Mapping Systems 15 (2011).
[19] B. Amol, H. Monson, S.T. Mark, Robust lane detection and tracking with RANSAC and Kalman filter, in: Proceedings of 16th IEEE International Conference on Image Processing (ICIP), 2009, p. 32613264.
[20] Y. Sibel, Y. Gökhan, D. Ekrem, Keeping the vehicle on the road: a survey on on-road lane detection systems, ACM Comput. Surv. 46 (1) (2013) 2:1–2:43.
[21] M.C. Joel, T.M. Mohan, Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation, IEEE Trans. Intell. Transp. Syst. 7 (1) (2006) 20–37.
[22] W. Jung-Ming, C. Yun-Chung, C. Shyang-Lih, C. Sei-Wang, Lane marks detection using steerable filters, in: Proceedings of 16th IPPR Conference on Computer Vision, Graphics and Image Processing, 2003, pp. 858–865.
[23] S.M. Angel, R.F. Javier, M. Luis, B.L. Miguel, B. Luciano, A color vision-based lane tracking system for autonomous driving on unmarked roads, Auton. Robots 16 (1) (2004) 95–116.
[24] A. Nicholas, Z. Alexander, Vision in and out of vehicles: integrated driver and road scene monitoring, International J. Robot. Res. 23 (2004) 513538.
[25] T. Min, F. Liu, Z. Hu, Single camera 3D lane detection and tracking based on EKF for urban intelligent vehicle, in: Proceedings of Conference on IEEE International Vehicular Electronics and Safety, 2006, pp. 413–418.
[26] A. Mammeri, A. Boukerche, L. Guangqian, Lane detection and tracking system based on the mser algorithm, hough transform and Kalman filter, in: Proceedings of the 17th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2014, pp. 259–266.
[27] W. Phueakjeen, N. Jindapetch, L. Kuburat, N. Suvanvorn, A study of the edge detection for road lane, in: Proceedings of 8th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2011, pp. 995–998.
[28] J. Matas, C. Ondrej, M. Urban, T. Pajdla, Robust wide-baseline stereo from maximally stable extremal regions, Image Vision Comput. 22 (10) (2004) 761–767.
[29] D. Chaudhuri, A. Samal, A simple method for fitting of bounding rectangle to closed regions, Pattern Recognit. 40 (7) (2007) 19811989.
[30] P.V.C. Hough, A method and means for recognizing complex patterns, US Patent: 3,069,654 (1962).
[31] K. Zu, Realtime lane tracking of curved local road, in: Proceedings of the IEEE Intelligent Transportation Systems, 2006, pp. 1149–1155.
[32] J. Matas, C. Galambos, J. Kittler, Robust detection of lines using the progressive probabilistic hough transform, Comput. Vision Image Underst. 78 (1) (2000) 119–137.