A Robust, Real-Time and Calibration-Free Lane Departure Warning System

Islam Gamal[1], Abdulrahman Badawy[2], Awab M.W Al-Habal[2], Mohammed E.K Adawy[2], Keroles K. Khalil[1], Magdy A. El-Moursy[1], Ahmed Khattab[2]
[1]*Mentor, A Siemens Business, Cairo, Egypt*
[2]*Electronics and Electrical Communications Department, Faculty of Engineering, Cairo University, Giza, Egypt*
E-mails: islam_gamal@mentor.com, Abdulrahman_Badawy@outlook.com, awab.habal.95@gmail.com, mohammedelborolossy@gmail.com, Keroles_Khalil@mentor.com, magdy_el-moursy@mentor.com, akhattab@ieee.org

***Abstract*—A real-time and calibration-free lane departure warning system algorithm is proposed. The pre-processing stage of the lane departure algorithm is carried out using Gaussian pyramid to smooth the image and reduce its dimensions, which decreases the unnecessary details in the image. A lane detection stage is then developed based on the Edge Drawing Lines (EDLines) algorithm that is a real-time line segment detector, which has false detection control. The reference-counting technique is used to track the lane boundaries and predict the missing ones. Experimental results show that the proposed algorithm has accuracy of 99.25% and average processing time of 80 fps (frame per second). The proposed algorithm is efficient to be used in the self-driving systems in the Original Equipment Manufacturers (OEMs) cars.**

Keywords— Driver Assistance System (DAS); Lane Departure Warning System (LDWS); Edge Drawing (ED); Region of Interest (ROI); Line Segment Detection (LSD)

I. INTRODUCTION

Population growth and the rapid increase in the number of vehicles have led to more traffic accidents every year, which become a serious problem. There are many reasons causing these traffic accidents, but unintended lane departures, which are caused by drivers, occupy the fourth rank among these reasons. According to the National Highway Traffic Safety Administration (NHTSA), 37% of deaths in traffic accidents in USA are caused by lane departures [1]. That prompts the development of many Driver Assistance Systems (DASs) which provide the driver with essential information about the surroundings and prevent driver from making unintended mistakes [2].

Lane Departure Warning System (LDWS) is a mechanism designed to warn the driver when the vehicle begins to move out of its lane unless a turn signal is on in that direction. Most of the proposed techniques in the literature uses machine vision (MV) to implement the LDWS [2]. These techniques use the existing infrastructure and it can be adapted easily with road design changes. They are also based mainly on the inverse perspective mapping (IPM) to ease lane detection by getting the bird's eye view of the road in front of the car. IPM has a high computational time, which affects the real-time performance of the system. Also, it is parameter-based and requires calibration of the camera for every different type of car which makes the system unportable [3]. A new algorithm based on MV to implement a non-IPM-based LDWS with high efficiency and real-time performance is provided in this paper, as shown in Fig 1. A real-time and calibration-free LDWS (RTCF-LDWS).

The remaining of this paper is organized as follows. In section II, a brief overview of the related previous work is presented. In section III, the proposed algorithm is introduced and described in details. In section IV, experimental results for the proposed algorithm are provided. Some conclusions are portrayed in section V.

II. RELATED WORK

In this section, the related work in LDWS is reviewed. In the data acquisition stage of the LDWS which is based on MV technology, there are two main approaches of using the camera: the single-camera approach [4],[5], and the multi-camera approach [6]. The single camera approach is the most widely used in the industry because of its low cost. Multi-camera approach [6] provide data redundancy resulting in high accuracy. However, using more than one camera increases the cost and the required computational time.

LDWS consists of different stages, in the pre-processing stage unnecessary details in the image are reduced by applying image filters such as the Gaussian Filter [5] and Median Filter [7]. Region of Interest (ROI) extraction is used also to determine the sufficient portion of the image required to detect the lanes correctly [5].

Line detection stage is done in two steps: edge detection then line segmentation. For edge detection Canny edge detector is used in [8] to find the intensity gradients of the image. In [3] Laplacian edge detection is used. For line segmentation many

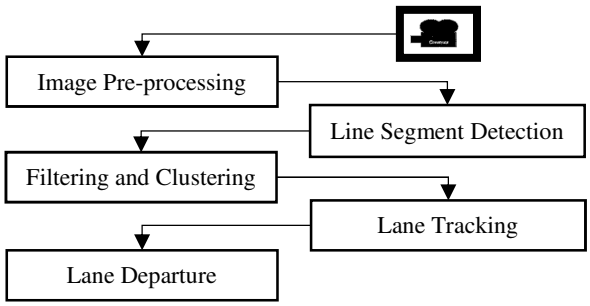

Fig 1 Block diagram of the proposed algorithm

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

methods are used as in [9]. Another methods combines the two stages in one stage as Hough transform [10] and Line Segment Detector (LSD) [11].

Following line detection, tracking methods have been used to track the lane boundaries. In [3], a scoring mechanism has been proposed to keep track of the lane boundaries using the concept of reference-counting and predict their locations if they are missed in one frame or more.

In the last stage, the position of the car with respect to the lane is calculated in each frame using successive information about it then the departure can be detected as in [14]. The stages of the proposed algorithm are described in the next section.

III. THE PROPOSED ALGORITHM

The stages of the algorithm are shown in Fig 1 and explained in detail in the following subsections.

A. *Image Pre-processing*

In real-time LDWS (like the presented one in this paper), it is very important to shrink the image as much as possible to minimize the processed items and decrease the processing time. This is achieved in two sub-stages: ROI extraction, and image smoothing.

1) *ROI extraction*

In the proposed algorithm, it is required to detect the lane boundaries on the road; therefore, the part of the image that contains the ground "the bottom part" is the most important part. An adaptive ROI is extracted as shown in Fig 2. The most important points in the ROI are the forth and the fifth which separate the top and the bottom parts of the image by detecting what is called the vanishing point.

The vanishing point is the point at which all lines in the image plane intersect. To ensure the stability of vanishing points calculations, it is calculated every frame but it is updated only every 10-20 frames using a feedback loop. Therefore, the calculations are stabilized and not affected by dropping frames. The vanishing point and the other ROI points are initialized such that the system converges to an appropriate region.

2) *Image smoothing*

In this stage, Gaussian pyramid approach (which is the most commonly used) is adopted as it gives the best results with the minimum processing time compared with other smoothing techniques. The main idea of Gaussian pyramid is to smooth the image then down sample it and repeat this process several times.

Smoothing the image diminishes the details and shrinks it reducing the processing time. Therefore, using this stage reduces the number of detected lines at the line segment detection stage and speeds up the system. Fig 3 shows the results of applying the used line segment detection algorithm on an image with and without using Gaussian pyramid.

B. *Line Detection*

One of the most important stages in any LDWS is the line detection stage. This stage is responsible for detecting edges in a given image (which form lines, curves, or other shapes) and distinguishing the lines from the other shapes. This stage is divided into two sub-stages: the edge detection and the line segment detection.

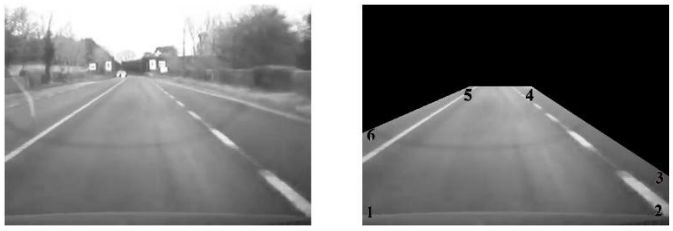

Fig 2 ROI points

1) *Edge detection*

The ED algorithm [13] is used as it runs at real-time and produces each edge as a pixel chain. This algorithm works on a grayscale image and performed as follows: the gradient magnitude is computed at each pixel such that it is possible to extract pixels with maximum gradient. These pixels (which are called the anchors) are connected with smart routing procedure to form the edges.

2) *Line Segment Detection*

EDLines algorithm [14], based on ED algorithm, is used to perform the line segment detection based on Least Squares (LS) line fitting method. It fits points in a certain coordinate system to be expressed linearly in terms of axes of this system. The basic idea is to walk over the edge pixels chain and fit them to lines using the LS method. The algorithm generates an initial line segment and then extends this line segment by adding more pixels to it. The value of the minimum line length depends on the lane boundaries length in pixel unit. Chain pixels are add to the current line as long as the pixels are within a certain distance from the line, e.g., 1 pixel error. The algorithm continues adding pixels to the current line segment until the direction of the line changes.

At this point, a complete line segment is found and the remaining pixels of the chain are then processed recursively to extract more line segments.

C. *Line Filtering and Clustering*

Line segments found by EDLines are defined mainly by a start point (x_1, y_1) and an end point (x_2, y_2). Using these two points the equation of the line in the form $y = mx + c$ can be acquired and the length of the line segment could be acquired, where m is the line slope and c is y-intercept.

Fig 3 The effect of using the Gaussian Pyramid on the number of lines detected by line detection algorithm. The original image to the left and the image after applying 3-level Gaussian Pyramid to the right

In this stage, the line segment is defined using five features (slope (m), intercept Point (c), start point (Sx, Sy), end point (Ex, Ey) and length(l)). The lane is defined by only two lines: left and right ones, but EDLines algorithm finds more than two line segments in the image so there is a need to extract the lane boundaries from these line segments, this is done by the proposed techniques in this stage.

1) Line filtering

In this part, the line segments are filtered so that only the ones that related to lane boundaries are selected and passed to the next stage. The selection process is done using thresholds for line segment features, and it is found that separating lines in the left side of the frame from the lines in the right side is useful for the selection process. This separation and selection process is illustrated in Fig 5.

2) Line clustering

The lane boundaries between two lanes in the same direction are dashed lines with certain thickness which causes the EDLines to detect it as two parallel lines as shown in Fig 4. The problem is similar to classification problem in machine learning where it is required to find lines close to each other then consider them as one line. The distance between two lines is defined using the following equation:

$$Distance = (m_1 - m_2)^2 + (c_1 - c_2)^2 \qquad (1)$$

where m_1, m_2, c_1 and c_2 are slopes and y-intercepts of the two lines. A threshold is used to find similar lines that could be merged into one line whose slope and y-intercept are the mean of the clustered lines' ones.

D. Lane Tracking

In this stage, temporal information about lane boundaries is used to achieve two goals: make a final decision for which clustered lines are lane boundaries and predict lane boundaries if they are not detected. It is very intuitive that things in real life do not change suddenly; therefore, the lane boundaries in the current frame are relatively close to the ones in the previous one. The mentioned concept is the main driver of the tracking algorithm.

Every clustered line is defined by its slope and intercept, so first of all two parameters are added for every line: count and verified flag. These parameters (beside slope and intercept) are

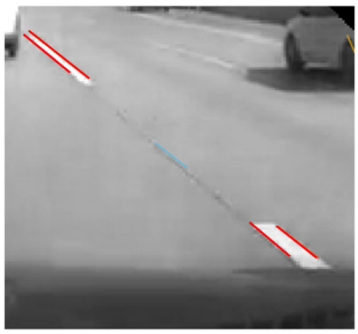

Fig 4 Illustration of problem solved by line clustering

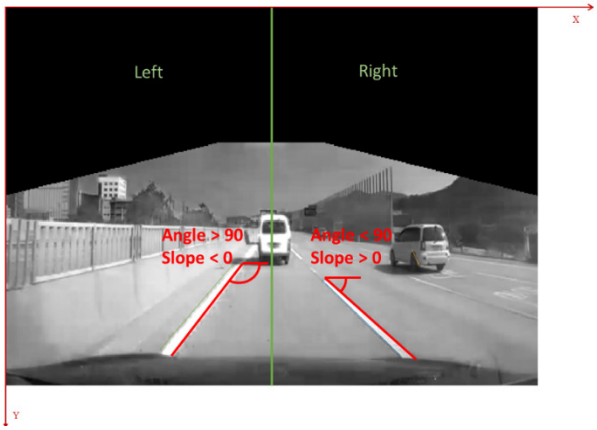

Fig 5 Separating left lines from right lines

updated in every frame and passed from one frame to another and they are collected in a list which is called *TrackedList*.

Tracking is done in two steps: correlating clustered lines with lines in *TrackedList* and updating lines in *TrackedList.* The clustered lines are compared with lines in *TrackedList* to find best match between them and previous lane boundaries, if a line in *TrackedList* has best match then it's features are updated with the new values from matched clustered line, if a line in *TrackedList* has no best match then it's count is decreased only. After that if there are clustered lines with no best match in *TrackedList*, then they are added to *TrackedList* with zero count and zero verified flag.

Finally, lines in *TrackedList* are updated, if the count is larger than a certain threshold the verified flag is raised. This means that this line was detected in many frames and the probability that it is related to the lane boundary is high. If the count of the line is less than a certain threshold then it is deleted from *TrackedList*.

In the second case, where there is no enough information about the previous frames, the tracked list is initialized empty and all detected lines are added to the tracked list. If there is no detected lines, the tracking stage is bypassed until there is a line detected in the image.

E. Lane Departure

In this stage, it is required to find the position of the car with respect to the lane. First of all, the camera is assumed to be mounted behind the windshield mirror so that the mid-point of the car is in the middle of the image. This assumption ensures simpler algorithm for departure calculation.

The lane is defined by its boundaries and to simplify the calculations, the mid-point of the lane is determined by intersecting the boundaries with the bottom of the image (which is a horizontal line with equation $y = image\ height$) then taking half the distance between intersection points. It worth mentioning that only lines with raised verified flag in *TrackedList* are used in the calculations, as they are the most probable to be related to lane boundaries.

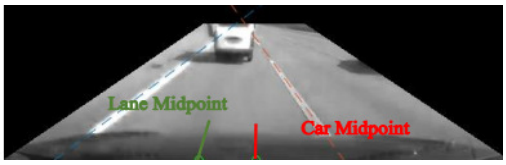

Fig 6 Position of the car with respect to the lane boundaries

Using the mid-point of the lane and the mid-point of the car as shown in Fig 6, the car lane departure is calculated using the following equation:

$$Departure(d) = \frac{L - C}{W/2} \qquad (2)$$

where L is the lane midpoint, C is the car midpoint and W is the lane width.

The values of the departure (d) starts from -100% (the car is moving on the left lane boundary) to 100% (the car is moving on the right lane boundary). In order to ensure good warning behavior, the departure range is divided into three regions: the safe region, the warning region and the danger region as shown in Table I.

Table I Regions Based on Deviation Value

Region	Departure values
Safe region	$-40\%\ to\ 40\%$
Warning region	$-60\%\ to\ -40\%$ $40\%\ to\ 60\%$
Danger region	<-60% and >60%

IV. EXPERIMENTAL RESULTS

To guarantee that the driver safety is accomplished by our system, the system is tested by a various challenging weather and illumination conditions: clear, cloudy, rainy, day, sunset and night. The datasets used are presented in [15],[16],[17] and [18]. The average frame processing time is 12.5 milliseconds on Intel(R) Core(TM) i7-5500U CPU @ 2.4GHz. The detection departure rates are calculated using equation 4 and 5, respectively.

$$Detection\ Rate = \frac{TP}{TP + FN} \qquad (3)$$

$$Departure\ Rate = \frac{Hit}{Hit + Miss} \qquad (4)$$

where *TP* and *FN* are true positive and false negative results, respectively and *Hit* is successfully detected departures and *Miss* is not detected departures. The average detection and departure rates are 99.25% and 100% respectively.

V. CONCLUSIONS

In this paper, a new reliable and robust algorithm to implement a LDWS is introduced. The RTCFLDWS algorithm is real-time and scalable. It reduces the input image using region of interest extraction. Edge detection and line segmentation method called EDLines is applied. It is fast and accurate with false detection control. The filtering and clustering block uses basic machine learning to select only the lines related to lane boundaries from the detected lines. The lane boundaries are tracked while they change as the car moves. The lines are drawn on a GUI display and a warning signal appears when the departure happens. From the results, the proposed system achieved detection rate of 99.25%, departure rate of 100% and average processing time 12.5 milliseconds.

REFERENCES

[1] LeBlanc, D., *"Road departure crash warning system field operational test: methodology and results. volume 1: technical report"*. 2006.

[2] Wu, C.-B., L.-H. Wang, and K.-C. Wang, *"Ultra-low Complexity Block-based Lane Detection and Departure Warning System"*. IEEE Transactions on Circuits and Systems for Video Technology, 2018.

[3] Lotfy, O.G., et al. *"Lane departure warning tracking system based on score mechanism"*. in *Circuits and Systems (MWSCAS), 2016 IEEE 59th International Midwest Symposium on*. 2016. IEEE.

[4] Irshad, A., et al. *"Real-Time Lane Departure Warning System on a Lower Resource Platform"*. in *Digital Image Computing: Techniques and Applications (DICTA), 2017 International Conference on*. 2017. IEEE.

[5] Kortli, Y., et al. *"A novel illumination-invariant lane detection system"*. in *Anti-Cyber Crimes (ICACC), 2017 2nd International Conference on*. 2017. IEEE.

[6] Borkar, A., M. Hayes, and M.T. Smith. *"A new multi-camera approach for lane departure warning"*. in *International Conference on Advanced Concepts for Intelligent Vision Systems*. 2011. Springer.

[7] Baili, J., et al. *"Lane departure detection using image processing techniques"*. in *Anti-Cyber Crimes (ICACC), 2017 2nd International Conference on*. 2017. IEEE.

[8] Kodeeswari, M. and P. Daniel. *"Lane line detection in real time based on morphological operations for driver assistance system"*. in *Signal Processing, Computing and Control (ISPCC), 2017 4th International Conference on*. 2017. IEEE.

[9] Cho, N.-G., A. Yuille, and S.-W. Lee, *"A Novel Linelet-Based Representation for Line Segment Detection"*. IEEE transactions on pattern analysis and machine intelligence, 2018. 40(5): p. 1195-1208.

[10] Lee, D.-K., et al. *"Real-time lane detection and tracking system using simple filter and Kalman filter"*. in *Ubiquitous and Future Networks (ICUFN), 2017 Ninth International Conference on*. 2017. IEEE.

[11] Zhou, F., Y. Cao, and X. Wang, *"Fast and resource-efficient hardware implementation of modified line segment detector"*. IEEE Transactions on Circuits and Systems for Video Technology, 2017.

[12] Berriel, R.F., et al., *"Ego-Lane Analysis System (ELAS): Dataset and Algorithms"*. Image and Vision Computing, 2017. 68: p. 64-75.

[13] Topal, C., O. Ozsen, and C. Akinlar. *Real-time edge segment detection with edge drawing algorithm*. in *Image and Signal Processing and Analysis (ISPA), 2011 7th International Symposium on*. 2011. IEEE.

[14] Akinlar, C. and C. Topal. *"Edlines: Real-time line segment detection by edge drawing (ed)"*. in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. 2011. IEEE.

[15] Lichtenberg, D., "GitHub," Microsoft, [Online]. Available: https://github.com/udacity/CarND-LaneLines-P1/tree/master/test_videos.

[16] Aly, M., "*Real time Detection of Lane Markers in Urban Streets*," IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, June 2008

[17] Yoo, H., U. Yang, and K. Sohn, *"Gradient-enhancing conversion for illumination-robust lane detection"*. IEEE Transactions on Intelligent Transportation Systems, 2013. 14(3): p. 1083-1094.

[18] Yoo, J. H., et al. "A robust lane detection method based on vanishing point estimation using the relevance of line segments." *IEEE Transactions on Intelligent Transportation Systems* 18.12 (2017): 3254-3266.