2013 IEEE Intelligent Vehicles Symposium (IV)
June 23-26, 2013, Gold Coast, Australia

An Efficient Lane Detection Algorithm For Lane Departure Detection

Heechul Jung[1], Junggon Min[2] and Junmo Kim[3]

Abstract— **In this paper, we propose an efficient lane detection algorithm for lane departure detection; this algorithm is suitable for low computing power systems like automobile black boxes. First, we extract candidate points, which are support points, to extract a hypotheses as two lines. In this step, Haar-like features are used, and this enables us to use an integral image to remove computational redundancy. Second, our algorithm verifies the hypothesis using defined rules. These rules are based on the assumption that the camera is installed at the center of the vehicle. Finally, if a lane is detected, then a lane departure detection step is performed. As a result, our algorithm has achieved 90.16% detection rate; the processing time is approximately 0.12 milliseconds per frame without any parallel computing.**

I. INTRODUCTION

Recently, many studies of Advanced Driver Assistance Systems (ADAS) have been conducted to prevent car accidents. Especially, the Lane Departure Warning System (LDWS), which is a system to warn drivers when the vehicle is moving out of its lane, is the most basic and necessary part of the ADAS. This LDWS, which is usually based on a lane detection algorithm using a camera, is operated under low power and has to share resources such as the Central Processing Unit (CPU), Random Access Memory (RAM), and Read Only Memory (ROM) with other drivers assistant algorithms. Also, to attract customers, the system should have a low price. Therefore, the lane detection algorithm for LDWS, should be compact, light, and efficient.

Previous lane detection algorithms can be classified into three categories such as line or curve fitting based algorithms, color based algorithms, and learning based algorithms. The representative curve fitting based algorithm is Lane detection and tracking using B-Snake, which was proposed by Wang [1]. This algorithm has achieved good performance even when the lane is curved, but it takes a long time to find the lane. Another curve fitting lane detection algorithm was proposed by Aly [2]; this algorithm is efficient and fast, but it is not suitable for computational systems of lower power than that of a desktop computer. Color based algorithms are usually fast and efficient [3]; however, they can be degraded in environments in which the illumination suffers extreme changes. Kim has proposed the Support Vector Machine (SVM) based lane detection algorithm [4]. However, the weakness of learning based algorithms is that it is hard to collect training data using such algorithms. The most recently proposed algorithm is A Novel Lane Detection System With Efficient Ground Truth Generation, which is presented in [5]. This algorithm also requires training data and it requires a high computational cost (0.8 second per frame). Consequently, it is not suitable for low computational systems.

In this work, we concentrate on a low computing system that has a 240 Mhz CPU and 10 MB available memory. Most algorithms are not applicable to a low computing system because they require high computational power. To develop an efficient algorithm, we adopt two approaches: integral image and local search. Also, a simple verification method is used for filtering correct lane detection result. Finally, our algorithm has achieved considerably fast speed and has shown promising results.

This paper is organized as follows. First, an efficient lane detection algorithm for lane departure detection is proposed in Section 2. Second, evaluation of our algorithm is performed in Section 3, using our constructed database. Finally, based on these results, we conclude this paper, and consider some remaining issues, in Section 4.

II. PROPOSED ALGORHTM

In this section, we propose an efficient lane detection algorithm for LDWS. The algorithm is operated according

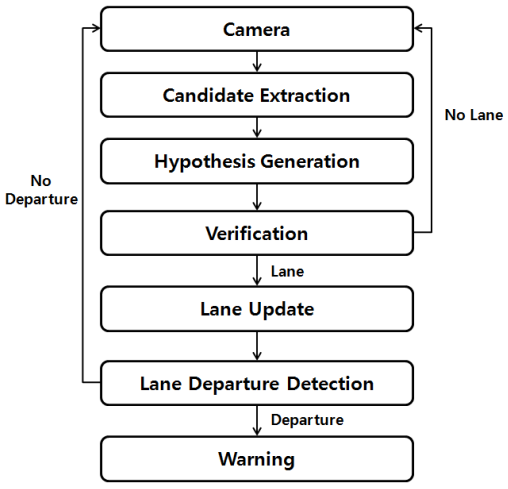

Fig. 1. Block Diagram of Our Algorithm for LDWS

[1]Heechul Jung is with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea, heechul at kaist.ac.kr

[2]Junggon Min is with New Eye Vision company, Ulsan, Korea, push007 at nev.co.kr

[3]Junmo Kim is with Faculty of the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea, junmo at ee.kaist.ac.kr

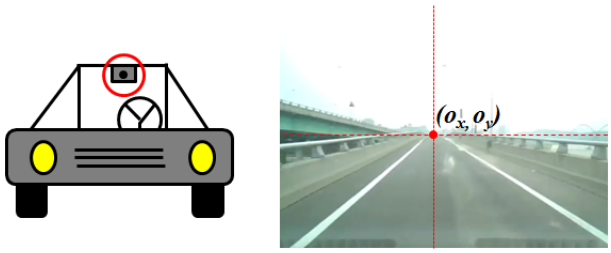

Fig. 2. Camera Position(Red Circle) and Its Output Image

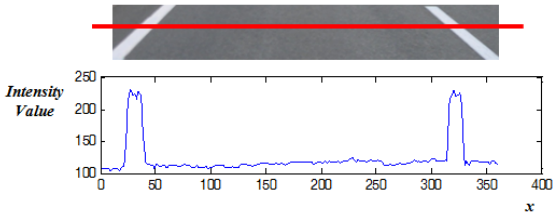

Fig. 3. The Intensity Value along the Red Line on the Image.

to the procedure shown in Fig. 1. First, the candidate points, which are the support points to obtain two lines, are extracted using Haar-like features. Second, each generated line is verified using simple rules that are based on one assumption. The assumption is that the camera is installed at the center of the vehicle. Third, if the verification is passed, then each lane has been detected. Finally, our algorithm detects lane departure, when the driving car is moving out of the lane, and warns the driver to stay in the lane.

A. Camera Installation

In this section, we present the camera installation method that was used to obtain good performance using our algorithm. First, the camera must be located at the center of the vehicle. In other words, the center of the lane is same as the center of image. Second, the focal axis of the camera should be parallel to the road. This means that the vanishing point extracted from a straight lane in the image is near the center of the image. An installed camera satisfying the above constraints is shown in Fig. 2.

However, for more precise setting, camera calibration step using checker board is necessary. This is a process to compute the virtual center (o_x, o_y) in Fig. 2. In other words, the checker board is installed at the midpoint of the front of the vehicle, and then the center (o_x, o_y) is defined as the coordinate of the center of the checker board image. For example, o_x and o_y will be the half of the image width and image height, respectively, provided that the camera is perfectly located at the center of the vehicle.

B. Candidate Extraction

The graylevel of the road surface is usually homogeneous, and the lane has comparatively higher intensity values than the road surface, as shown in Fig. 3. The easiest way to detect the lane is a thresholding method, but this method is not robust when the illumination changes. Another method is to use a filter that is very responsive to differences of intensity value. Typical filters satisfying this property are edge detectors such as Sobel, Laplacian, and Prewitt filters; however, these kinds of filters can be sensitive to impulse noise. Consequently, a smoothing process is necessary, but such a process requires additional computational cost. Therefore, we adopt steerable filters instead of edge detectors; these are less sensitive to noise.

1) Steerable Filter: A steerable filter is an orientation dependent filter, so it is very responsive to the corresponding orientation [6], [7]. The lane in the image usually has a diagonal direction due to perspective view. Hence, to detect the lane in the image, a diagonally directional filter is suitable. However, the lane direction in the image is changeable according to the road direction or camera viewpoint, so to use particular directional filter can be a problem. To solve this problem, we use a vertically directional filters ($0°$, $180°$) as shown in Fig. 4. (b), (d). These two filters are not only responsive to the $330°$ direction, but are also very responsive to the $210°$ drection.

In this step, we are interested in maximal response, which can be one lane, so the position of maximal response should be detected. However, if there are multiple lanes, to detect the maximal response can be a problem. Also, even if there exists only one lane, there can be a problem due to the presence of two lines. To solve this problem, we first divide the image into two non-overlapped rectangle regions such as the left and right of the image.

For example, a 640×480 image is divided into two rectangle regions, each of which has a 320×480 image size. Then, our algorithm searches from right to left for the left image and searches from left to right for the right image. If the response value is higer than the previous value, the algorithm checks if the absolute difference is larger than the threshold T_r. If the value is larger than , the position becomes the candidate position.

2) Approximated Steerable Filter: To convolve the steerable filter with the image can be a bottleneck in our framework due to floating points and redundant computations. To solve these problems, we adopt an approximated steerable filter to the Haar-like features that are presented in [8]. Fig. 5 clearly shows the corresponding Haar-like features of the steerable filters in Fig. 4 and their responses. As a result, the maximal response position is invariant, even if we use Haar-like features. The advantage of Haar-like feature usage is that this enables us to adopt an integral image, which is used for preventing redundant computation[9].

C. Local Search

In this step, we set up the local search region. There are two reasons to use the local search region. First, the computation of the integral image for whole image can be

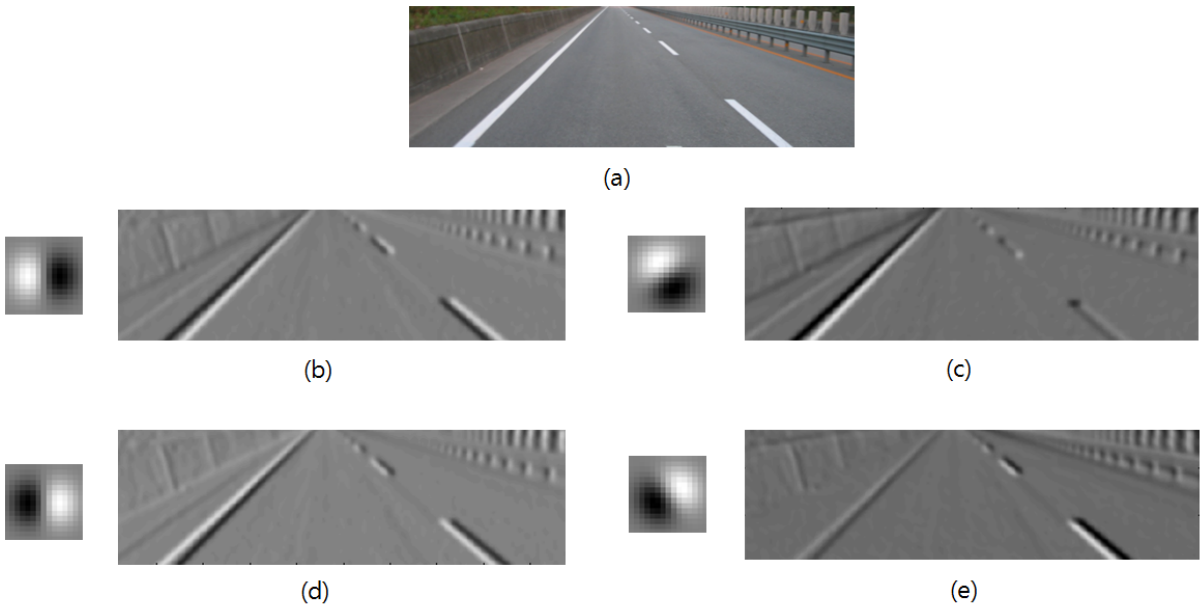

Fig. 4. Steerable Filters and Its Responses (a) Input Image (b) 0°, (c) 45° (d) 180° (e) 135°

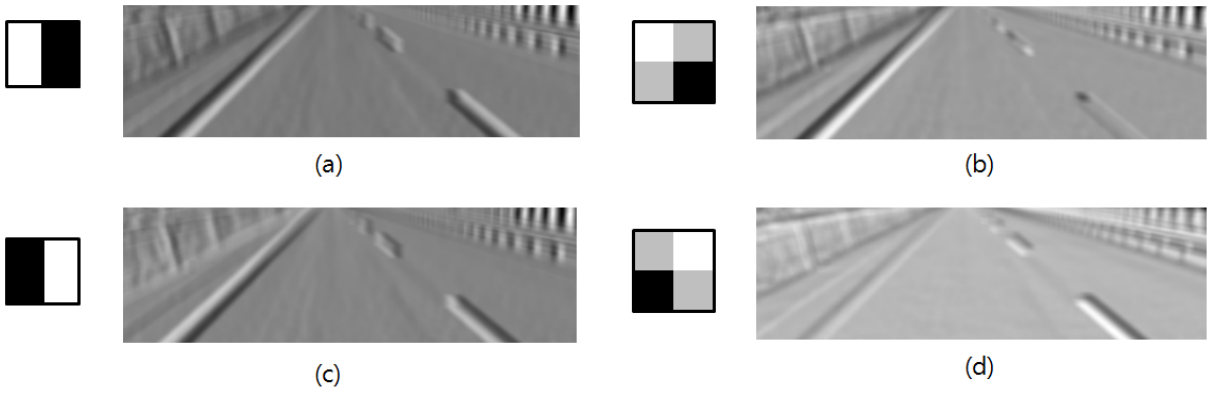

Fig. 5. Corresponding Haar-like Filters to the Steerable Filter in Fig. 4 and Its Response (a) 0°, (b) 45° (c) 180° (d) 135°

a hard task for a low power computing system. Second, the proposed algorithm can cause incorrect detection results when there exist outliers like traffic markers on the road. To resolve the first case, we reduce the search region into 3 local regions for each lane. Consequently, the computational cost is reduced, because the integral image is partially calculated only for those three regions. For the second case, we focus on the position of the traffic marker. In general, a traffic marker is allocated in the center of the lane. To avoid incorrect results caused by a traffic marker, the center part of the local regions should be removed. Finally, we have local search regions as shown in Fig. 6.

D. Hypothesis Generation

In this step, the hypotheses are generated. First, each lane (left, right) is computed from the detected points using the following equation.

$$\mathbf{l}_1 = \mathbf{p}_1 \times \mathbf{p}_3, \quad \mathbf{l}_2 = \mathbf{p}_4 \times \mathbf{p}_6 \tag{1}$$

$\mathbf{p}_i$ is the homogeneous coordinate of point i. Moreover, $\mathbf{l}_1$ and $\mathbf{l}_2$ represent the left and right lanes, respectively. In world coordinates, the two lanes are parallel, so the extracted lines($\mathbf{l}_1$, $\mathbf{l}_2$) shoud be converged at infinity.

The vanishing point, which is the point at infinity, is calculated using the following equation.

$$\mathbf{v} = \mathbf{l}_1 \times \mathbf{l}_2 \tag{2}$$

$\mathbf{l}_1$ and $\mathbf{l}_2$ indicate lines calculated by the candidate points; the cross product of the two lines is a point. The above process is clearly expressed in Fig. 7

E. Hypothesis Verification

Hypothesis verification is performed for two lanes $\mathbf{l}_1$ and, $\mathbf{l}_2$ and for vanishing point $\mathbf{v}$. The lane verification is

performed to check if each extracted lane is correct or not. This is performed by calculating the distance between a point and a line. The distance from point $\mathbf{p}$ to line $\mathbf{l}$ is defined as follows[10]:

$$distance(\mathbf{l}, \mathbf{p}) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} \tag{3}$$

where $\mathbf{p} = [x_0 \quad y_0 \quad 1]^T, \mathbf{l} = [a \quad b \quad c]^T$.

This is an exact solution for the distance from a point to a line, but it can be a heavy task for low power computingsystem if the number of points is increased. Consequently, we simplify the above eqaution using a constraint. In general, the distance between a point and a line is defined as the distance between the point $\mathbf{p}$ and the point $\mathbf{q}$, which is the intersection between line $\mathbf{l}$ and its perpendicular, which contains the point $\mathbf{p}$ where $\mathbf{q} = [m \quad n \quad 1]^T$. However, in our algorithm, we dont use the perpendicular. Instead, the line parallel to the x-axis is used. Consequently, n is equal to y_0, and so the simplified equation is as follows.

$$distance^*(\mathbf{l}, \mathbf{p}) = |m - x_0| \tag{4}$$

To verify each lane, we use threshold T_l as follows

$$distance^*(\mathbf{l}_1, \mathbf{p}_2) < T_l, \quad distance^*(\mathbf{l}_2, \mathbf{p}_5) < T_l \tag{5}$$

In other words, if the distance from the point p_2 to the line l_1 is under the threshold T_l, then the line is correct. In a way similar to that used in the above verification method, verification for the vanishing point is also performed. First, the vanishing line has to be defined. The vanishing point lies on the vanishing line, and the line is the horizon in reality. Actually, to find the vanishing line in the image is not an easy task. To reduce calculation strain on the system, we installed the camera in order to locate the horizon at the center of the image. Hence, the vanishing line is defined as follows:

$$\mathbf{l}_v = [0 \quad -1/o_y \quad 1] \tag{6}$$

where o_y is the y coordinate of the virtual center in image. Hence, the distance from the vanishing point to the vanishing line is easily induced.

Fig. 6. Local Search Region(Black Rectangles)

$$distance^\dagger(\mathbf{l}_v, \mathbf{v}) = |y_1 - o_y| \tag{7}$$

where the vanishing point $\mathbf{v} = [x_1 \quad y_1 \quad 1]^T$.

To verify the vanishing point, the thresholding value is also used.

$$distance^\dagger(\mathbf{l}_v, \mathbf{v}) < T_v \tag{8}$$

F. Lane Departure Detection

Once the lane is detected, the lane departure detection step is performed. This lane departure detection function can be simply implemented. First, the criteria should be defined in order to determine that the vehicle is moving out of its lane. We use the horizontal center line as the criteria.

We assume that the distance between the vanishing point and the horizontal center line increases if the vehicle is going out of its lane. Consequently, we calculate the distance using the following equation.

$$distance^\ddagger(\mathbf{l}_h, \mathbf{v}) = |x_1 - o_x| \tag{9}$$

where the horizontal center line $\mathbf{l}_h = [-1/o_x \quad 0 \quad 1]^T$; o_x is the x coordinate of virtual center in image. Similarly to the hypothesis verification method, the lane departure is detected using the threshold.

$$distance^\ddagger(\mathbf{l}_h, \mathbf{v}) > T_d \tag{10}$$

This value T_d is dependent on the width of the vehicle. In other words, T_d has a large value for compact cars, and a small value for large-sized vehicles. Also, we can determine the lane departure direction using the sign of the equation $(x_1 - w/2)$. Finally, we obtain the lane departure detection with its direction as follows:

$$departure = \begin{cases} right & \text{if } (x_1 - o_x) < -T_d \\ left & \text{if } (x_1 - o_x) > T_d \\ no & \text{otherwise} \end{cases} \tag{11}$$

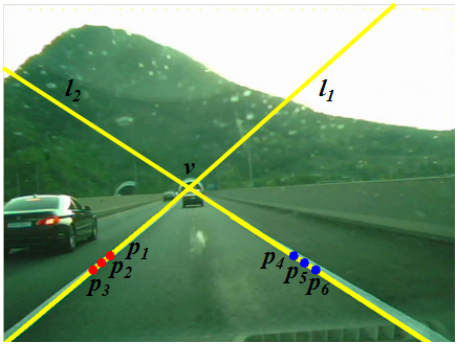

Fig. 7. Candidate Points $p_1, \ldots, p_6$, Produced Two Lines l_1, l_2 using each Candidate Point. Calculated Vanishing Point v using The Lines.

Fig. 8. Experimental Results using Our Algorithm for Various Situations. (a) Dotted lane (b) Solid lane (c) Yellow color lane(right) (d) Road traffic mark (e) Right curve (f) Left curve (g) Specular reflection (h) Night

III. EXPERIMENTAL RESULT

We have performed two kinds of experiments, both quantitative and qualitative evaluations, to verify our lane detection algorithm a desktop computer that had an Intel i5 2.8 GHz CPU and 8 GB RAM. Also, we have ported the algorithm into a low computational system that has a 240 Mhz CPU and 10 MB available RAM. Our algorithm was implemented using C language. First, for qualitative evaluation, we have used our constructed database, which contains various conditions; the results are shown in Fig. 8. The image size is 640×480; image was recorded at 24 frames per second. Our algorithm worked well, even when the lighting conditions changed drastically. The lane departure detection results are shown in Fig. 9. The departure detection clearly worked but the lane detection did not work correctly in this case due to using a local search region. Second, we have manually selected 5000 frames in the daytime for quantitative evaluation. The detection rate is 90.16%; the average processing time is 1.21×10^{-4} seconds per frame. The standard deviation of the processing time is 8.5047×10^{-6}. Finally, our algorithm has been ported onto a car black box, which is a low computational power system. In general, car black boxes do not have enough resources, because they store the input video in real time (20fps). Our algorithm on the system worked at approximately 14 fps and is compact enough so that the binary file size is only 6.3 KB.

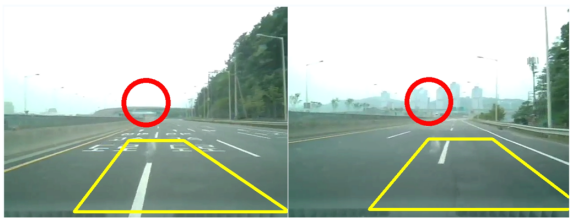

Fig. 9. Lane Departure Detection Results. Red circle means that the lane departure is detected.

IV. CONCLUSION

We have developed an efficient lane detection algorithm and appied it to lane departure detection. Our algorithm utilizes Haar-like features to detect the lane efficiently because such features enable the system to use integral image. Futhermore, we have presented a hypothesis generation and verification method using an assumption based on the camera installation constraints. As a result, our algorithm has achieved a 90.16% detection rate; the processing time was approximately 0.12 milliseconds per frame. In the future, we will extend our algorithm using a robust algorithm like Random Sample Consensus in order to manipulate more various situations such as a stain on the windshield, moving wipers, and shadows on the road.

ACKNOWLEDGMENT

The authors would like to thank New Eye Vision (`http://www.nev.co.kr/`). Also, this research is supported in part by Leading Industry Development for Chung-cheong Economic Region (A0022 00627), and National Core Research Center of the National Research Foundation of Korea (2012-0000987).

REFERENCES

[1] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision Computing*, vol. 22, no. 4, pp. 269 – 280, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0262885603002105

[2] M. Aly, "Real time detection of lane markers in urban streets," in *Intelligent Vehicles Symposium, 2008 IEEE*, june 2008, pp. 7 –12.

[3] K.-Y. Chiu and S.-F. Lin, "Lane detection using color-based segmentation," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, june 2005, pp. 706 – 711.

[4] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 1, pp. 16 –26, march 2008.

[5] A. Borkar, M. Hayes, and M. Smith, "A novel lane detection system with efficient ground truth generation," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 1, pp. 365 –374, march 2012.

[6] W. Freeman and E. Adelson, "The design and use of steerable filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 13, no. 9, pp. 891 –906, sep 1991.

[7] T. Gao and H. Aghajan, "Self lane assignment using egocentric smart mobile camera for intelligent gps navigation," *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 0, pp. 57–62, 2009.

[8] M. Villamizar, A. Sanfeliu, and J. Andrade-Cetto, "Computation of rotation local invariant features using the integral image for real time object detection," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 4, 0-0 2006, pp. 81 –85.

[9] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511 – I–518 vol.1.

[10] "Wikipedia, the free encyclopedia: Distance from a point to a line," http://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line.