Vision-based single-lane and multi-lane estimation under highway and mountain road scenario

Runfa Li [1], Yancong Deng [1], Luyao Xu [2]

[1]Department of Material Science and Engineering, Jiangsu University

212013, Zhenjiang, China

[e-mail: 3150704065@stmail.ujs.edu.cn]

[1]Department of Electrical and Computer Engineering, University of California, San Diego

92093, San Diego, USA

[e-mail: yad002@eng.ucsd.edu]

[2]Department of Electrical and Computer Engineering, University of California, Los Angeles

90095, Los Angeles, USA

[e-mail: luyaoxu.ee@ucla.edu]

*Corresponding author: Luyao Xu

Abstract

Lane detection plays an important role to vision-based autonomous vehicle as well as many Adaptive Driver-Assistance Systems (ADAS). In this paper, we propose two different lane-detecting algorithms, namely, the Equally-Separated Algorithm, the Updated-Window Algorithm, to achieve accurate single-lane detection in highway and mountain road environments. The Equally-Separated Algorithm is stable enough when running in simple road conditions, while the Updated-Window Algorithm is more robust when coping with complex road conditions such as the large steering angle, dense shadow of roadside trees, changing light intensity of sunshine, disturbance of passing-by vehicles in adjacent lanes. Based on the Updated-Window Algorithm, we develop a Multi-Lane Algorithm which manages to cover four lanes simultaneously in highway testing environment including the driving lane, the adjacent lanes that two-lanes away to the driving lane. To further explore the potential and limitation of the algorithm, we test the proposed algorithms in a more complex road condition, the mountain road condition. Extensive results demonstrate that the proposed methods can achieve reliable and accurate land-detection performance under various light conditions both in highway and mountain road scenarios.

Keywords: Lane detection; Autonomous vehicle; Adaptive Driver-Assistance Systems (ADAS); Equally-Separated; Updated-Window; Single-lane; Multi-lane; Highway; Mountain road

1. Introduction

Recent years have witnessed the ever-increasing number of auto-mobiles, which has exerted a series of problems on modern cities such as traffic management, inter-vehicle connection and public safety[1-3]. Annually, there are around 1.2 million people all over the world die as a result of traffic accident[4], around 1% to 3% of the world's gross product is spent on the cost related to accidents caused by vehicles. According to the Road Safety Action Program of European Commission, cost of the traffic accidents has been estimated at 160 million Euros, which accounts for 2% EU's GNP [5]. In this circumstance, the development of safer autonomous vehicle and ADAS has become an urgent issue. Nowadays, the vision-based autonomous vehicle mainly depends on monitoring the exterior of the vehicle[6] [7], which involves edge detection with image processing, lane detection with computer vision.

For edge detection, the gradient-based method is always the first to consider, Sobel detector[8], Canny edge detector[9] are representative detectors using the gradient to detect edges. Many papers also develop algorithms to enhance the gradient between lanes and road. Wang[10] used histogram equalization to enhance the contrast, Kim[11] empirically determined RGB weights to 0.5:0.4:0.1. Wang[12] applies gamma correction on binary images to achieve a dynamic brightness compensation. Yoo[13] proposed a linear discriminant analysis (LDA)-based gradient-enhancing conversion for illumination robust lane detection, this algorithm dynamically adjusts RGB weights to the optimum ratio. For lane detection, there are representative methods such the steerable filter[14][15], and Hough Transformation[16][17]. The steerable filter, constructed by orienting the second derivative of the Gaussian filter, is separable and capable of detecting gradients of pixels in different orientation. Shang[18] builds a steerable filter detecting lanes with the estimated vanishing point position as a high level information, and implement the steerable filter on a Field Programmable Gate Array (FPGA) device without using any external memory. Guo[19] proposes a stripe Hough transformation to enhance extraction of the geometry of the road path to enhance the discrimination of the radar features in Millimeter-wave (MMW) radar image. When time dimension is involved, other filters and models are also applied to optimize and ensure the robustness of lane tracking, such as Kalman filter[20], Particle filter[21], and Hidden Markov Model[22]. On recent years, machine learning and deep learning methods are also involved in the lane detection. In Lim[23]'s research, structural and statistical features of the extracted bright shape are applied to the neural network for finding correct lane marks. Ajaykumar[24] proposed a K-Means Clustering algorithm to cluster data of lanes, while M.Mandalia[25] proposed a SVM-based algorithm by training with behavioral data including acceleration, steering angle, car distance and so on. Studies also attach greater importance on Convolutional Neural Network(CNN) recently, Bojarski[26] trained CNN with minimum data to build an End-to-End algorithm stably running on highway and even some area without visual guidance, while Li[27] simultaneously built two CNN tailored together to perform structured predictions to meet the need for lane detection.

While all researches above develop elaborate and complex algorithms and models, some of them did not attach importance to the computational efficiency and complexity. In this research, we develop two lane detecting algorithms, which benefit in the convenience and easiness to build, and the relatively computational efficiency compared with other complex algorithms and models. We firstly achieve edge detection to both yellow edge and white edge of lanes with the Canny edge detector sweeping over image in RGB color space and HSL color space respectively, then propose two different lane-detecting algorithms that are computationally efficient and robust to detect single lane and fit curve model for the single lane, the two algorithms are first tested respectively on a highway scenario. Then, a multi-lane detecting algorithm is initiated based on the single-lane detecting algorithm. In the highway scenario, the algorithm manages to estimate four lanes simultaneously. To see the

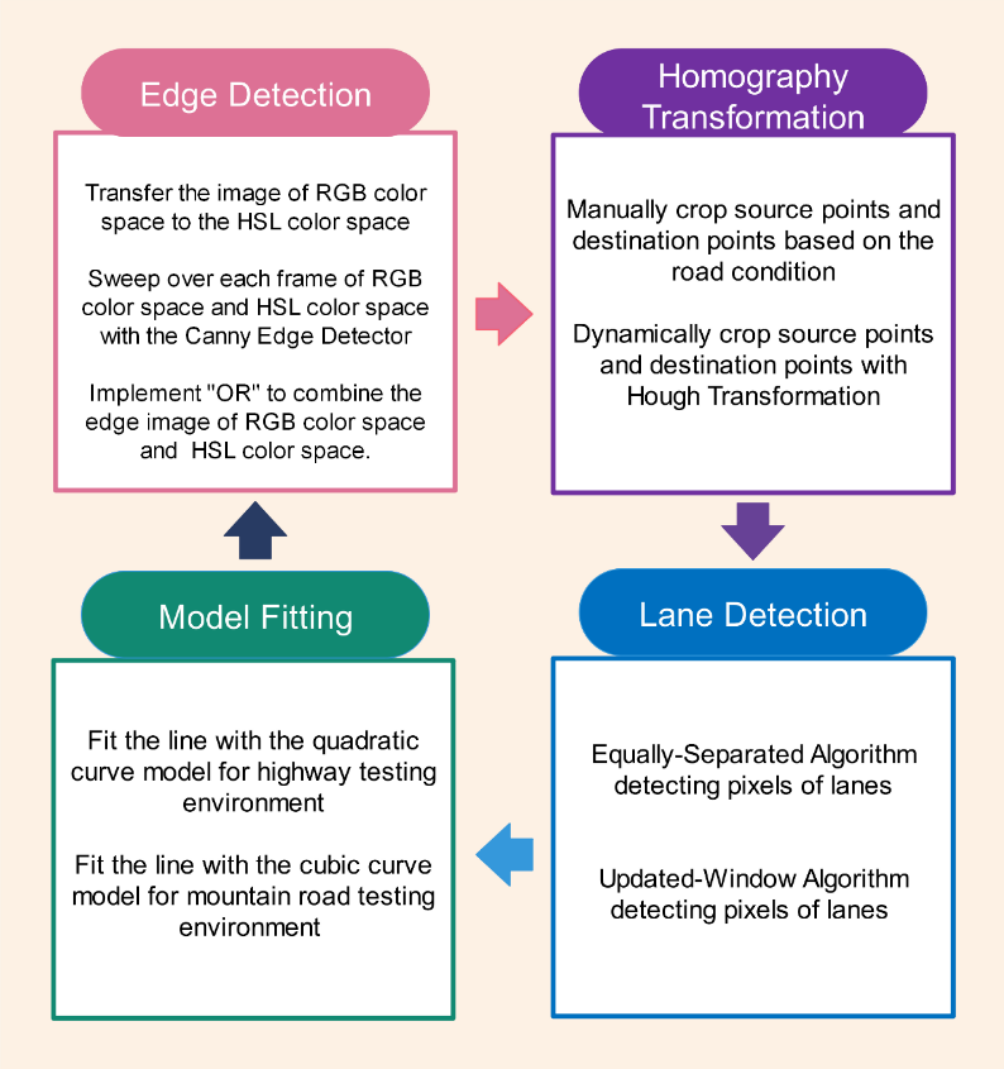


Fig. 1. Framework of the whole process of the vision-based lane-detecting algorithm..

potential and limitation of the algorithm, the single-lane detecting algorithm is finally tested in more complex scenarios on a mountain road testing environment. A simple illustration of the algorithm can be generalized in **Fig. 1**.

The remainder of paper is organized as follows. In section II, we describe the whole

process of edge detection, namely the extraction of the edges from the original frame, in our research. In section III, we discuss the pros and cons of two ways to implement the Homography Transformation, and select the superior one in our research. In section IV, the Equally-Separated Algorithm and the Updated-Window Algorithm are proposed, pseudo-code, explanation, and mathematical formula for each algorithm are introduced to make it clear. In section V, we detail the fitting model. In section VI, we propose the Multi-Lane Detecting Algorithm based on the Updated-Window Algorithm. In section VII, we test the Updated-Window Algorithm in a more complex road condition in the mountain road testing environment to further explore the potential and see the limitation of the algorithm. Finally, conclusions and prospect are drawn in section VIII.

2. Edge Detection

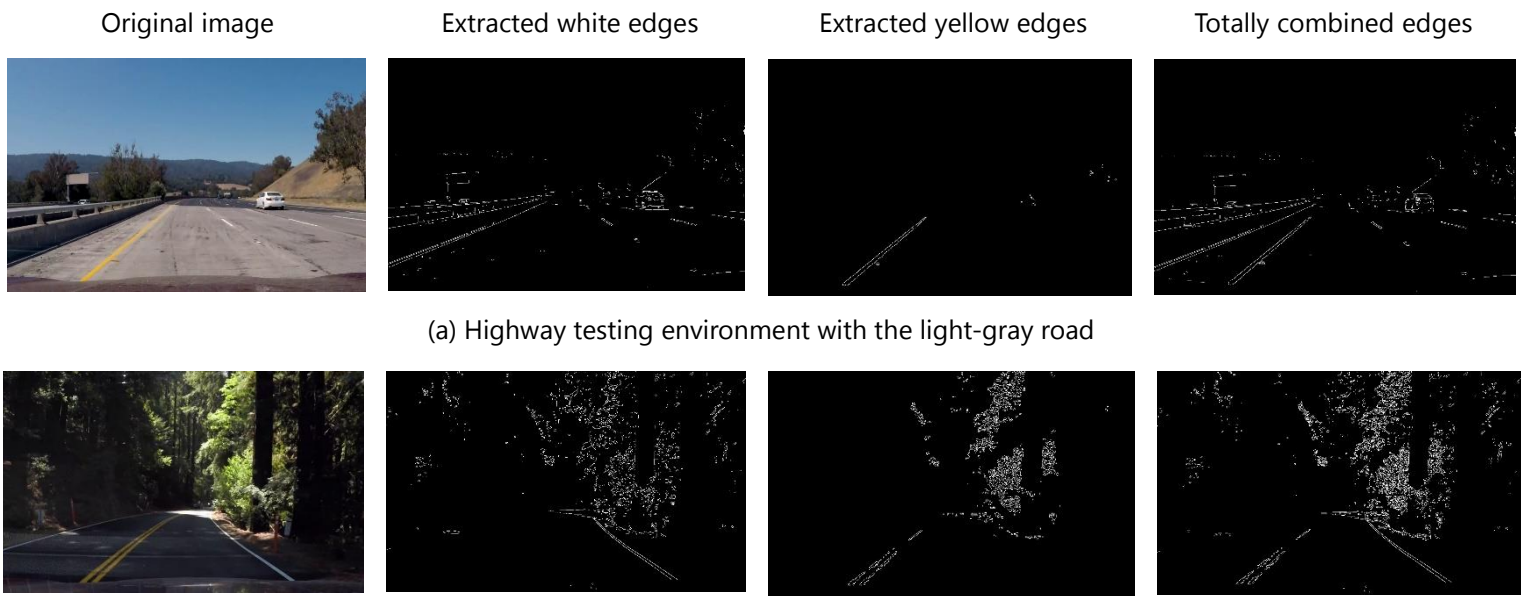
In a typical road condition, the lanes are in bright colors while the road are in dark colors, in this situation, gradient-based algorithms are suitable to detect lane edges. Widely-used gradient-based algorithms including the Canny edge detector, the Sobel edge detector and so on. The adaptive Canny edge detector are with two thresholds. Pixels with gradient value larger than the higher threshold th_h are regarded as edge pixels, pixels with gradient value smaller than the lower threshold th_l are discarded as non-edges, while pixels with gradient value between two thresholds are edge candidates if they have a path to the lane. In this study, th_h and th_l were set as 250 and 200 respectively due to the road situation. A Gaussian filter is then implemented to further eliminate noise pixels as well as discontinuous short lines. The outcome shows that the lane edges are well-preserved while noises on the road are clearly removed.

However, a thorny issue still left if only implementing the Canny edge detector. Generally, when the yellow line and the white line cover on the dark road, both the two lines can be extracted with Canny edge detector with thresholds set, but in complex testing environment, there are moments when the vehicle is moving on light-grey road **Fig. 2(a)** or the road is with a very dark light condition **Fig. 2(b)**. In these situations, pixels of yellow lines are surrounded by pixels with close intensity to them, which lead to the gradient intensity become less obvious, thus cannot detect yellow lines. Intuitively, enhancing the gradient intensity between the yellow lines and the road pixels should be the first to consider, several methods are proposed and tested, including histogram equalization[10], changing the color ratio to be R:G:B=0.5:0.4:0.1[11], using a updated-color background[28]. However, neither of these two approaches managed to extract yellow lines when driving on light-gray road or in the strong sunshine condition for complex testing environments of highway and mountain road.

Transferring RGB color space to HSL color space is a suitable solution. In this study, to extract yellow lines, the BGR value of the original RGB image is restricted from (10,0,100) to (40,255,255), HSL image was transferred to gray scale image, then implementing the Canny edge detector with the th_h as 210 and the th_l as 100. HSL color space amplify the gradient intensity of yellow lines, by contrast, it declines the gradient

intensity of white lines, which results in the outcome edge image only reserving edge of yellow lines or just little bit of white lines.

As a result, to simultaneously show white edges and yellow edges, implement "OR"



(a) Highway testing environment with the light-gray road

(b) Mountain road testing environment with the dark-light condition

Fig. 2. The process of getting yellow edge in HSL color space and white edge in RGB color space, and total edge images

to add edges of yellow and edges of white to form the ultimate edge image.

3. Homography transformation

Homography transformation is a commonly used strategy in the field of computer vision to transform an image in a given direction to an aim direction. Specifically, in the field of lane detection of autonomous car, the world's view image is transformed to a bird's view image which is convenient for subsequent implementation including detecting lane pixels and curve fitting[29].

$$X=Hx \tag{1}$$

To illustrate in the equation, the vector of world view coordinate x is transformed by the homography matrix H to get the vector of bird's view coordinate X. The homography transformation can be implemented using the built-in function in OpenCV. The built-in function requires four pairs of points as input, namely four points in world's view coordinate, four points in bird's view coordinate, and calculate the homography matrix H as output, which could be used as a map of transformation for all pixels in the world's view to the bird's view.

Many researches have involved Hough transformation for homography transformation. This is a very effective algorithm to deal with complex road conditions when going uphill, downhill, the steering angle is very large, or even the width between two lanes is changing. Inspired by [30], a dynamic homography transformation is also

attempted based on Hough transformation in this research. However, this algorithm is finally discarded because homography transformation is too computationally expensive to support the following lane detecting algorithm.

In our research, homography transformation is achieved by manually cropping four source points and four destination points, which is not only computational effective but also easily elaborated. However, setting source points and destination points must follow several principles instead of randomly selecting points, or else it may lead to some problems.

First, if the target is to detect only the self-lane, a bird's view edge image cannot involve too many lanes as shown in **Fig.3. (a)**. If there were too many lanes, each local area would also aggregate too many lane pixels, sometimes even more irrelevant pixels when other vehicles are passing by. So if lanes lie closely, it is very difficult to detect and categorize pixels in the following step. Meanwhile, when another vehicle passing by, there will be more noises mixing with lane pixels.

Second, as the vehicle is moving, the lane pixels in the bird's view edge image must enter from the up side of the frame instead of entering from left or right side of the frame, as shown in **Fig.3. (b)**. Pixels entering from left and right side of the frame do not hurt the following lane detecting step, but they do hurt the quality of the area that shaped by fitted curve lines and projected back to the world's view coordinate, namely the blue polygon area in our outcome. To illustrate, as shown in **Fig.3. (c)**, there is a notch on the top-left corner and the top-right corner respectively, when the vehicle is turning left and turning right. An optimum edge image in bird's view coordinate should be like **Fig.3. (d)**.

The highway testing environment is a video sequence with a resolution of 1280*720, the matching points have been repeatedly adjusted to the optimum value with four source points as (200, 720), (1100, 720), (590, 450), (685, 450), four destination points as (150, 720), (1020, 720), (300, 0), (980, 0), which turns out the optimum outcome corresponding to the principals above. The outcomes are shown in **Fig.3. (a)(b)(c)(d)** as the vehicle turning left and right respectively.

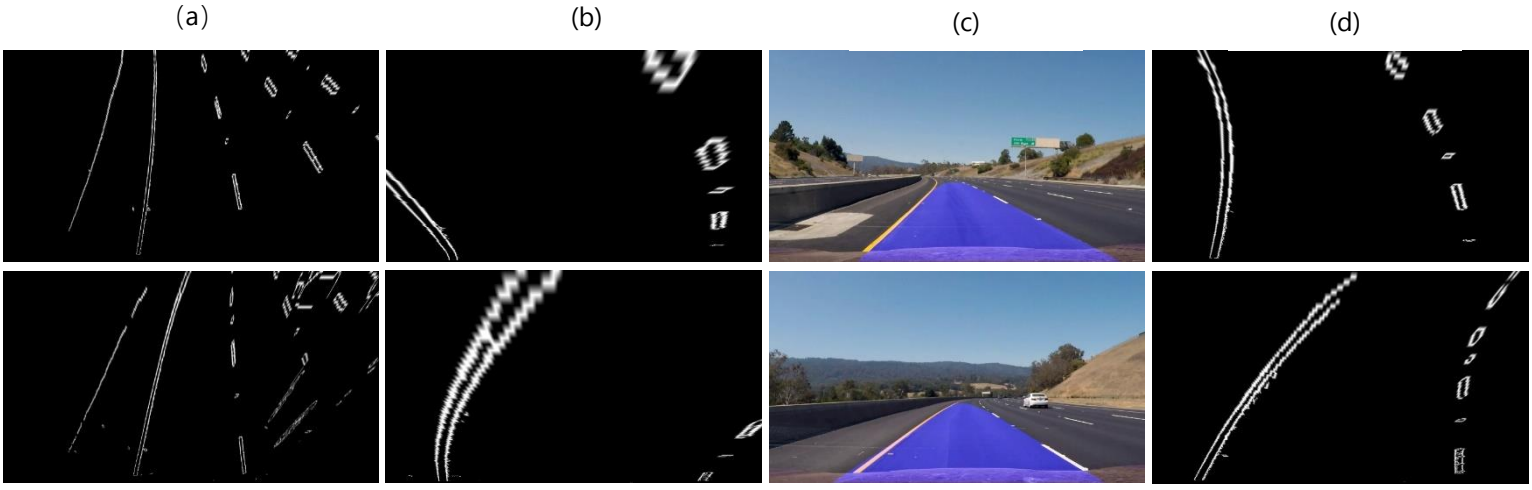


Fig.3. Principles that should be followed by homography transformation. (a), (b), (c) show the problematic outcomes if not obey the principles, (d) shows a standard form of edge image in bird's view coordinate.

4. Lane detection algorithms

After getting the edge image in a bird's view coordinate, the next step is to extract pixels or candidate pixels of the lanes. The algorithm to detect lane pixels must follow several principles, maximumly involving pixels of lanes and eliminate irrelevant pixels and noise. In this study, the Equally-Separated algorithm is developed firstly as the primary algorithm. an Updated-Window algorithm is then developed as the advanced algorithm. The two algorithms are tested in highway environment respectively.

4.1 Equally-Separated algorithm

The Equally-Separated algorithm is quite intuitive. Considering the pretreated bird's view image, there are only pixels of right lane and left lane. As a result, this primary algorithm is to detect and collect pixels of right lane and left lane respectively. The Equally-Separated Algorithm is shown as follows:

(1) Equally separate the bird's view coordinate in terms of the X axis as left rectangular part (LP) and right rectangular part (RP).
(2) Scanning pixel row by row from the upper side of the edge image to the lower side along the Y axis.
(3) In each row, scanning pixels from the left side to the right side along the X axis and extract four pixels, from left to right, the leftmost pixel of the left lane (LL), the rightmost pixel of the left lane (LR), the leftmost pixel of the right lane (RL), the rightmost pixel of the right lane (RR).
(4) Collecting and grouping pixels of each four classes LL, LR, RL, RR, saving LL, LR to L, saving RL, RR to R.

Algorithm1: Equally-Separated Algorithm

Input: **EB** (edge image of each frame in bird's view coordinate)

Begin
LP ← *getHalfPart (EB)*
RP ← *getHalfPart (EB)*
For r := 0 to y **step** 1 pixel **do**
LL ← *getLeftPix* (I_r, *LP);*
LR ← *getRightPix* (I_r, *LP);*
RL ← *getLeftPix* (I_r, *RP);*
RR ← *getRightPix* (I_r, *RP);*
L. append (LL, LR);
R. append (RL, RR);
end **for**
return L, R
end

Output: L (n*2*2 array saving the coordinate of LL and LR of the left line in a frame)
R (n*2*2 array saving the coordinate of RL and RR of the right line in a frame)

Variables
LP: Left part of the image
RP: Right part of the image
I_r: Intensity of pixels in row "r"
LL: The coordinate of the leftmost pixel of the left line in a row of a frame
LR: The coordinate of the rightmost pixel of the left line in a row of a frame
RL: The coordinate of the leftmost pixel of the right line in a row of a frame
RR: The coordinate of the rightmost pixel of the right line in a row of a frame
L: Final array saving all LL and LR in a frame
R: Final array saving all RL and RR in a frame

4.2 Updated-Window algorithm

Comparing with the primary Equally-Separated algorithm, the Updated-Window algorithm has two obvious advantages. First, it is much more robust to irrelevant lines and noises; Second, it is computationally effective with greater enhanced execution speed. The figure illustration of the algorithm is shown in **Fig. 4.** The Updated- Window algorithm is shown below.

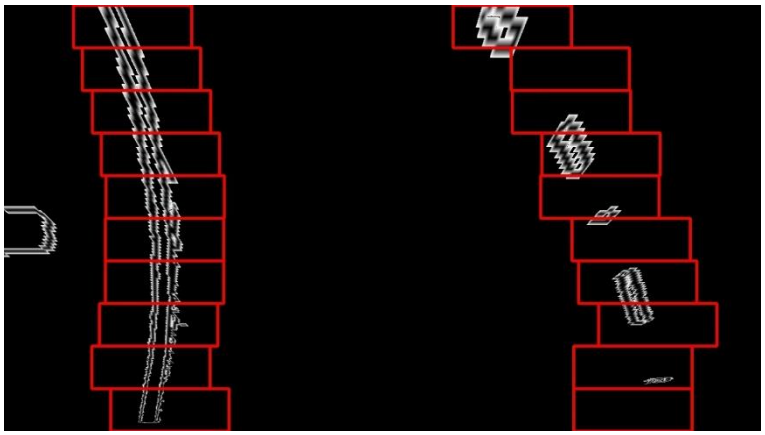


Fig. 4. Illustration of the advanced algorithm

(1) Build a histogram of the nonzero pixel number of lower half part of the bird's view edge image along the Y axis, equally separate the image into left part and right part, find the X position corresponding to the largest numbers of pixel in each part.

$$Histogram_i = \sum_{row=\lfloor \frac{y}{2} \rfloor}^{y} (the\ \ number\ \ of\ \ nonzero\ \ pixels) \tag{2}$$

$$X_l = Argmax(Histogram), where\ \ X_l \leq \lfloor \frac{X}{2} \rfloor \tag{3}$$

$$X_r = Argmax(Histogram), where\ \ X_r > \lfloor \frac{X}{2} \rfloor \tag{4}$$

(2) Regarding the two X positions as the starting center of the left window and the right window to scan over the nonzero pixels in the searching area. Establishing rectangular searching windows and only scan over pixels in the searching window.

$$\text{Left searching area: } \begin{cases} X_l - \frac{WinW}{2} \leq horizontal\ \ range \leq X_l + \frac{WinW}{2} \\ Y - \lfloor \frac{Y}{2WinN} \rfloor \leq vertical\ \ range \leq Y \end{cases} \tag{5}$$

Right searching area: $\begin{cases} X_r - \frac{WinW}{2} \le horizontal\ \ range \le X_r + \frac{WinW}{2} \\ Y - \lfloor \frac{Y}{2WinN} \rfloor \le vertical\ \ range \le Y \end{cases}$ (6)

(3) If the number of nonzero pixels in the searching area exceed the "minUpdate", figure out the average "X" of all pixels in the searching window as the updated center for the next window; otherwise, set the same "X" as the updated center for the next window. The following undated searching windows scan the lines from lower side to the upper side.

$\begin{cases} X_{n+1} = (\sum_{i=0}^{i=m-1} X_n^i)/m \\ Y_{n+1} = Y_n - \ WinH \end{cases}$ (7) or $\begin{cases} X_{n+1} = \ X_n \\ Y_{n+1} = Y_n - \ WinH \end{cases}$ (8)

(4) Collecting and saving the X, Y coordinates of all pixels detected and involved in the searching window.

Algorithm2: Updated-Window Algorithm

Input: **EB**: edge image of each frame in bird's view coordinate
WinN: The number of searching windows for each line
WnW: The width of the searching window
minUpdate: The minimum number of pixels found to update a searching window

Begin
Histogram ← getSumY (EB);
Mid ← whereMidX (EB);
L_go, R_go ← whereMax (Histogram, Mid);
NonZero ← whereNonZero(EB);
L_current, R_current ← L_go, R_go;
WinH ← Int(winN/ getSize(EB(Y)));
For i := 0 to n **step** 1 window **do**
Win_i^L ← getScope(L_current, winW, WinH);
Win_i^R ← getScope(R_current, winW, WinH);
In_L, In_R ← Nonzero(Win_i^L; Win_i^R);
L_indx. append(In-L);
R_indx. append(In-R);
if getNumber(In_L) > minUpdate **then**
L_current_X ← whereAverage(In_L_X);
if getNumber(In-R) > minUpdate **then**
R_current_X ← whereAverage(In_R_X);
end **for**;
L, R ← getCoordinate(L_indx, R_indx);
return L, R
end

Output: **L** (the coordinate of pixels that detected and involved by the left searching windows)
R (the coordinate of pixels that detected and involved by the right searching windows)

Variables
$Histogram$: The histogram of the nonzero pixel number of lower half part of the bird's view edge image along the Y axis
L_go: X, Y coordinate of the center point of the left starting window
R_go: X, Y coordinate of the center point of the right starting window
$L\_current$: X, Y coordinate of the center point of the left updated window
$R\_current$: X, Y coordinate of the center point of the right updated window
$WinH$: The height of the searching window, defined by the height of the frame divided by the number of windows
Win_i^L: The searching area of the i_{th} updated left searching window
Win_i^R: The searching area of the i_{th} updated right searching window
ln_L: The indices of specific nonzero points in the left searching area
ln_R: The indices of specific nonzero points in the right searching area

The most important two parameters for the searching window are the width and the height. The width is by manually setting. By setting a large width, the window can extend to more pixels on the horizontal direction, thus including more details. Moreover, an extended width guarantees the searching window to be able to reach far pixels when the lane is very curve. However, if setting too large width, the searching window may include other irrelevant edges. The height depends on the window numbers, since searching windows are filled and no space left from up to down. The more windows set, the shorter the height would be. With more searching windows, comes more updated times which can track the lane pixels more accurately. In a perspective of dynamic detection, it also makes the detection more stable between each frames; However, if setting too many windows, the updated searching windows would be too much influenced by local pixels instead of representing global trend of the curve lane; Plus, too many updated times would make the algorithm more computational expensive. Based on the factors above, the optimum length is set as 200 and the window numbers as 10 in the highway testing environment.

4.3 Fitting models

Having the detected pixels, the next step is to fit them into specific curve models. Currently, there are a lot of fitting methods, including Ransac[31], Kalman filtering[32], Particle filtering[33]. In this research, the least square method is finally selected considering the computational expense. In terms of the fitting model, the quadratic curve model is used for highway testing environment because the line edges in the bird's view coordinate are in a parabolic shape. To illustrate with equation.

$$x = \frac{1}{2}c_0 y^2 + my + b \tag{9}$$

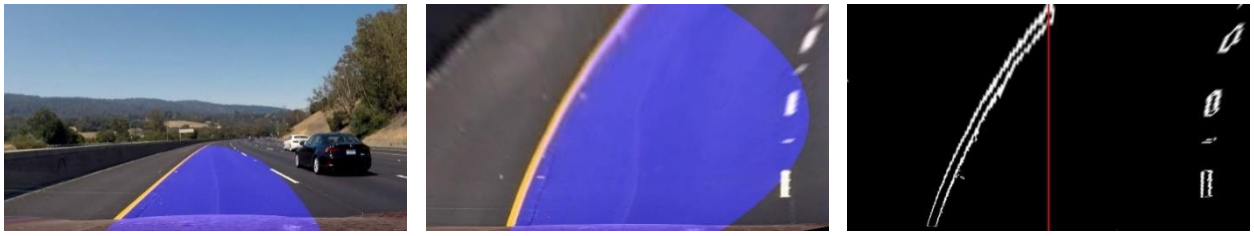
where x, y are horizontal direction and vertical direction respectively, C_0 is the curvature of the lane, m is the slope, b is the offset of the lane. Considering the simplicity of the code, $C_0/2$, m, and b are parameterized as A, B, and C, specifically AL, BL, CL for left line fitting; AR, BR, CR for right line fitting, the area confined by left and right lines is the lane area predicted. Using the same parameters A, B, and two different Cs to fit

globally would further accelerate the execution speed since only one line is fitted and which is then copied to the other line, but this is not a good idea because the curvature and the slope of each line is not necessarily the same, which causes clear errors.

5. Experimental result

The Equally-Separated Algorithm is quite intuitive and easy to elaborate. However,

(a) Detecting errors when other lines involved in the edge image



(b) Detecting errors when pixels of the left lane mistakenly group to the right lane

Fig.5. Illustration of detecting errors of the Equally-Separated algorithm

according to this naïve algorithm, saving pixels in four classes means simulating four lines independently, which is computationally expensive. To optimize the algorithm, it is conceivable to reduce the amount of lines from four to two. To be specific, still equally separating the image into left and right part, but this time, grouping all the pixels in the left part to the class of left lane, and all the pixels in the right part to the class of right lane, thus only simulating two curves later. The outcome shows better stability and robustness than before because more pixels are involved to fit the model. However, the processing speed, about 6 frame per second, hardly changed because the increasing number of pixels used to fit the model offset the speed added by reducing two fitting curves. Although it shows better quality, this algorithm still remains inevitable weakness that it highly depends on the quality of the bird's view image. First, when pixels of other lines appear at the corner of the image, the algorithm cannot eliminate the pixels but mistakenly involve them as candidate pixels of the lanes, which lead to noticeable errors to the final outcome as shown in **Fig.5(a)**. Second, when steering angle is very large, the algorithm may mistakenly categorize pixels of one lane to the group of the other lane, which lead to a very bad quality of the outcome as shown in **Fig.5(b)**.To further enhance the equality, separating each line and simulating each part respectively seems to be a way, however, in this way, not only each end of the line cannot link to each other, but also may lead to the problem of overfitting.

By contrast, when testing with the advanced Updated-Window algorithm, both the quality and the execution time are optimized. The advanced algorithm is quite stable and robust in several complex road conditions with other cars passing by on highway, including conditions such as road with large steering angle **Fig.6(a)**, road with white color **Fig.6(b)**, road with dense shadows of trees **Fig.6(c)**. Plus, the computational expense in the Updated-Window Algorithm has been greatly reduced and the execution speed has been greatly enhanced as shown in **Fig.7**. The execution speed in the original Equally-Separated Algorithm is only 6 frame per second, however, for the advanced Updated-Window Algorithm, the execution speed has increased to 20 frame per second, which is safe enough to catch up with the speed in a real driving scenario.

(a) Large steering angle (b) White road with cars (c) Dense shadows with cars

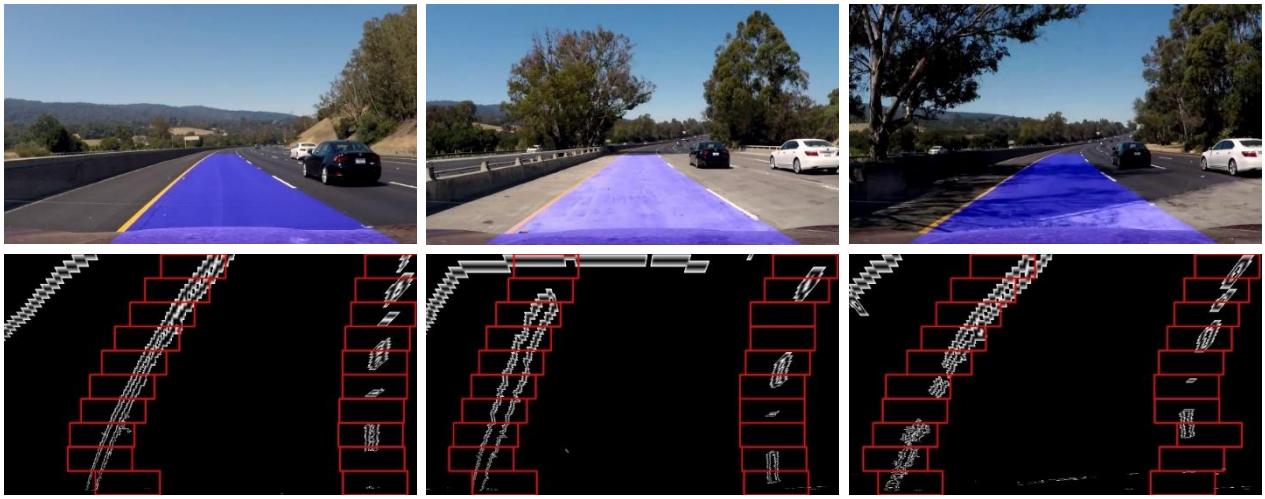


Fig.6. Outcome and illustration of the advance algorithm in complicated road conditions of highway testing environment

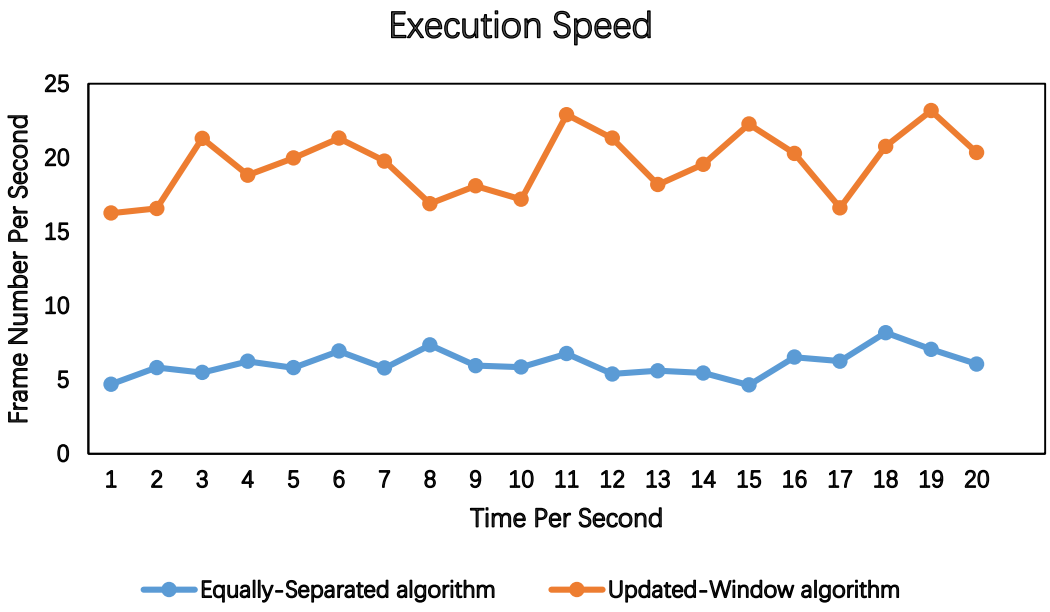
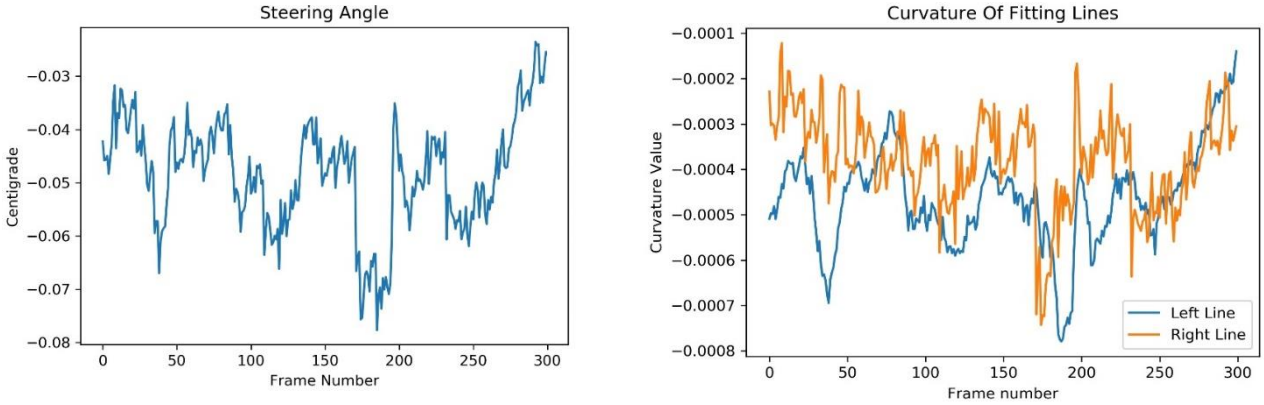
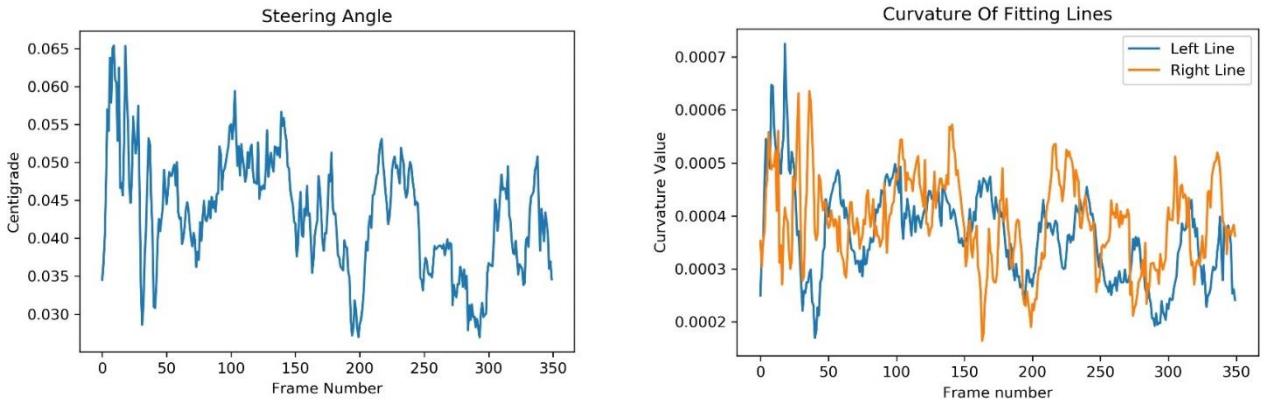


Fig.7. Comparison of the execution speed between the Equally-Separated algorithm and Updated-Window algorithm by selecting first 20 seconds of the highway testing environment.

At the expense of complexity, but in order to obtain optimum fitting outcomes, the right line and the left line are fitted respectively using the quadratic curve model. Six parameters are updated every frame, specifically AL, BL, CL for left line fitting; AR, BR, CR for right line fitting. To authenticate the quality of the fitting lines, the consistency of steering angle and the curvature of the fitting lines, namely AL and AR, are studied. We select 300 frames when turning left and 350 frames when turning right. **Fig. 8** shows the consistency of the steering angle and the curvature of fitting lines. According to the fitting model set in this research, when the vehicle is turning right, both the steering angle and the curvature are minus and their trends are consistent as shown in **Fig.8(a),** both the steering angle and the curvature are positive and their trends are consistent as shown in **Fig.8(b),** the absolute of the curvature is increasing when the steering angle is increasing and vice versa, which indicates that the fitting model perfectly reflect real-time situation of the detected lane.



(a) Consistency of the steering angle and the curvature of fitting lines when turning left



(b) Consistency of the steering angle and the curvature of fitting lines when turning right

Fig.8. Consistency of the steering angle and the curvature of fitting lines

Although the Updated-Window Algorithm runs robustly for both single-lane detection and multi-lane detection on highway environment, when tested in a mountain road environment with harder and complex road conditions, the algorithm still shows many limitations.

In this environment, the fitting model is changed to the cubic curve model because mountain road is always sinuous, sometimes it includes two curves in the opposite

direction, while cubic curves can accurately describe this kind of lane with two curves in the opposite direction. When the mountain road is under a constant light intensity, either in a light condition with much sunlight as shown in **Fig. 9(a)**, or in a dark condition with little sunlight as shown in **Fig. 9(b)**, the algorithm still runs robustly as it performs on highway environment.

(a) Road with much sunlight

(b) Road with little sunlight

Fig. 9. Desirable outcomes on mountain road environment

However, when road conditions become more complicated, the algorithm collapses and begins to mistakenly detect pixels. Main reasons and factors can be generalized as:

(1) Noise caused by dense shadow of trees in light condition, as shown in **Fig. 10(a)**.
(2) Noise caused by dense shadow of the barrier and trees in light condition, as shown in **Fig. 10(b)**.
(3) Noise caused by passing-by vehicle and dispersive sunlight spots in dark condition, as shown in **Fig. 10(c)**.
(4) Lack of lane pixels caused by sudden strong sunlight in dark condition, and noise caused by reflected image of instrument panel on the front windshield, as shown in **Fig. 10(d)**.

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

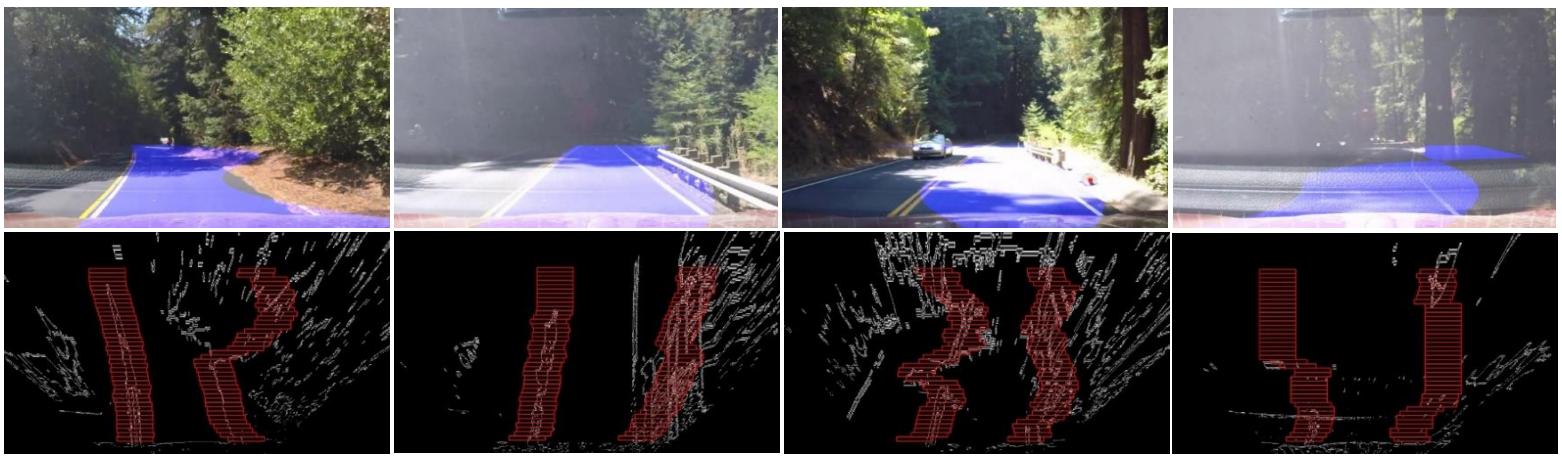


Fig.10. Undesirable outcomes when the algorithm mistakenly detect pixels due to different types of noise and factors

An even more thorny issue comes when the vehicle drives into the road with an extremely large steering angle. **Fig. 11** shows the whole process in this situation. In **Fig. 11(a)**, the searching windows separate from the end of the yellow lane because of the lack of lane pixels and the noise caused by dispersive sunlight spots; In **Fig. 11(b)**, because the searching angle continues to become larger, pixels of the left yellow line are mistakenly detected by searching windows for right white line. In **Fig. 11(c)**, because the steering angle becomes the largest, the right white lane totally disappears from the frame, and the algorithm mistakenly detect the lane beside. In **Fig. 11(d)**, the vehicle is pulling out of the corner, but the condition becomes even complex with dense shadow of trees and the large steering angle, the algorithm is collapsed.

To eliminate noise while preserving the relevant details at the best extent, the parameters of searching windows are adjusted to be optimum for this algorithm by reducing the length as 120, increasing window numbers to 40, and confining their positions in a limited range in terms of the Y axis. However, shortening the length of the searching windows can eliminate noise near the line but may also be too short to catch the curve line, because different pixels along the Y axis have large horizontal

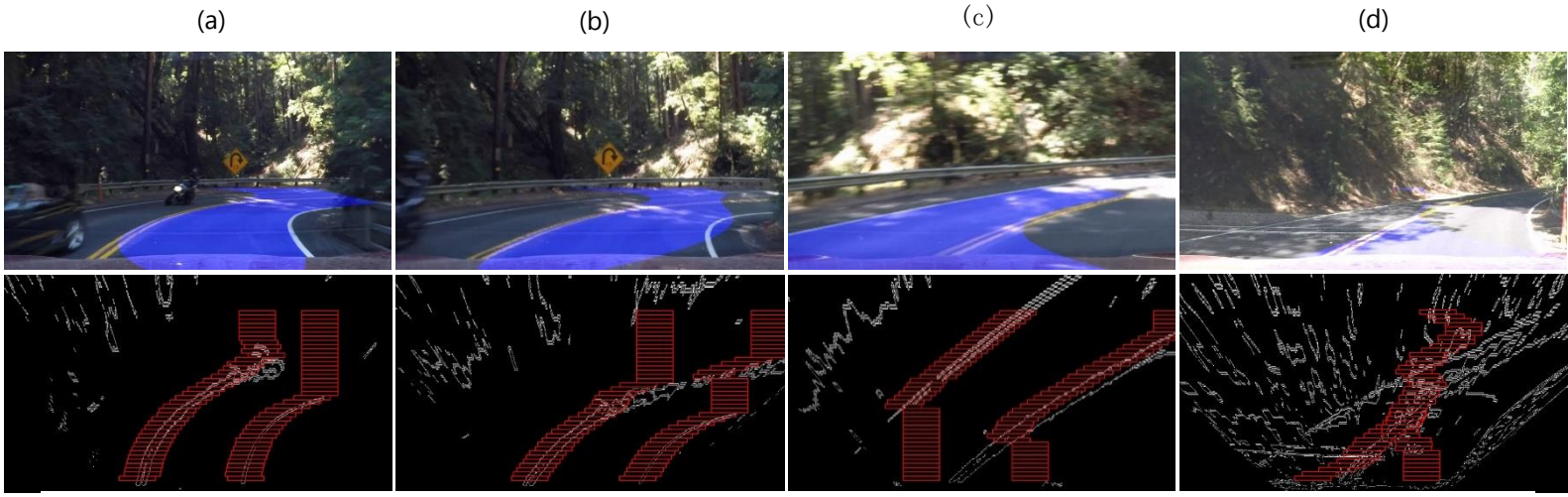


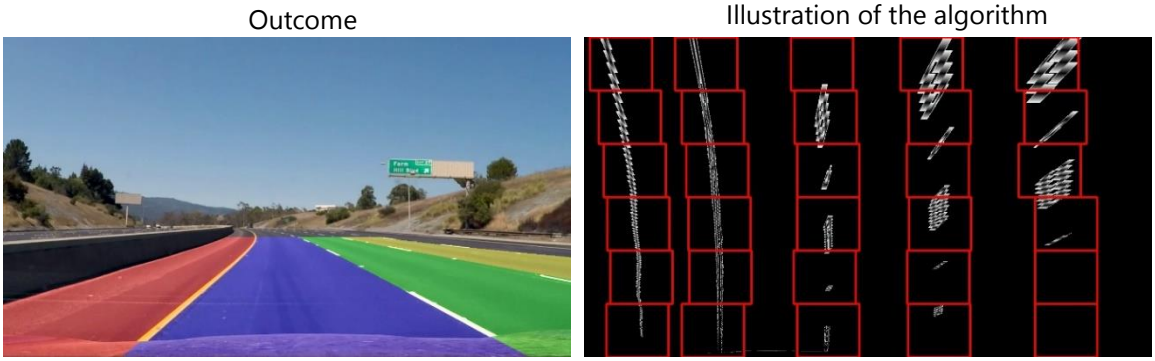
Fig. 11. Consecutive stages of the whole process when the vehicle drives into road with an extremely large steering angle

displacement when the steering angle is large. On the contrary, enlarging the length will catch up with the sudden increase of displacement of pixels but will also involve too much noise. As a result, this algorithm cannot perfectly balance the trade-off between eliminating noise while preserving details in this complex mountain road environment. We must further advance the algorithm, either by promoting the quality of edge image with less noise of shadows and sunlight spots, or by initiating new algorithm of lane detection to catch more detail of lines.

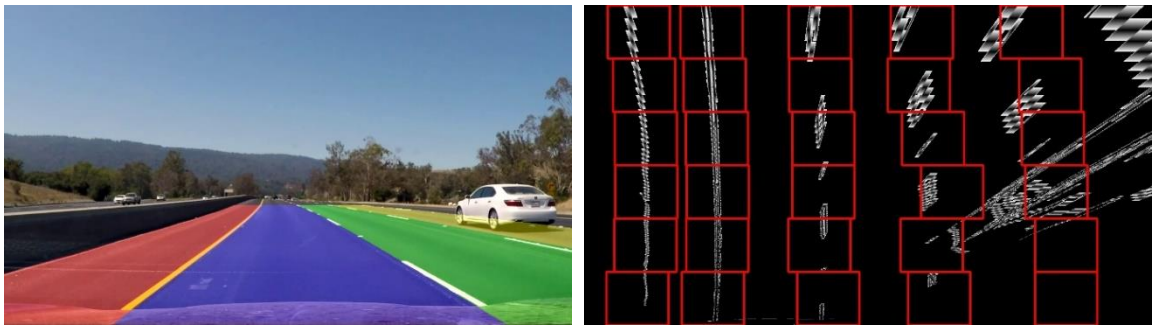
6. Multi-lanes Detection

Using the Updated-Window Algorithm and the quadratic curve fitting model, a multi-lanes detection is successfully achieved, specifically, four-lane detection on the same highway testing environment. Four different colors are used to denote the four lane areas as shown in **Fig. 12**. Because the multi-lanes detection requires to detect

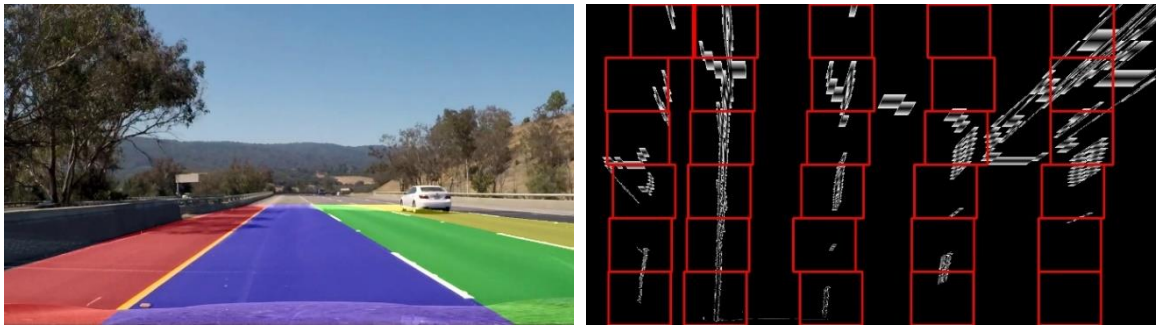
multiple lanes simultaneously in a frame, the bird's view edge image must include pixels of several lanes. As a result, the homography matrix is manually reset to be (237, 720), (1100, 720), (540, 450), (740, 450) as source points and (297, 720), (520, 720), (160, 0), (745, 0) as destination points. Four-lane detection need to detect five lines, in this circumstance, still build a histogram of the nonzero pixel number of lower half part of the bird's view edge image along the Y axis, but this time, there is no need to equally separate the image into left part and right part, instead, find the X coordinate corresponding to the five largest numbers of pixel in the histogram as the center of five starting windows. The width and height of the searching window are also reset. In the bird's view edge image, since each lane become close to each other, a shorter width is necessary to avoid mistakenly involving pixels of lateral lanes, the width is reduced to 140. The width of searching window is enlarged by reducing window numbers to 6, because a wider window on the perpendicular direction can involve more pixels one time and give stable updated position of the next window. The outcome shows that this algorithm is quite robust coping with different complex road environments on highway as shown in **Fig. 12**.



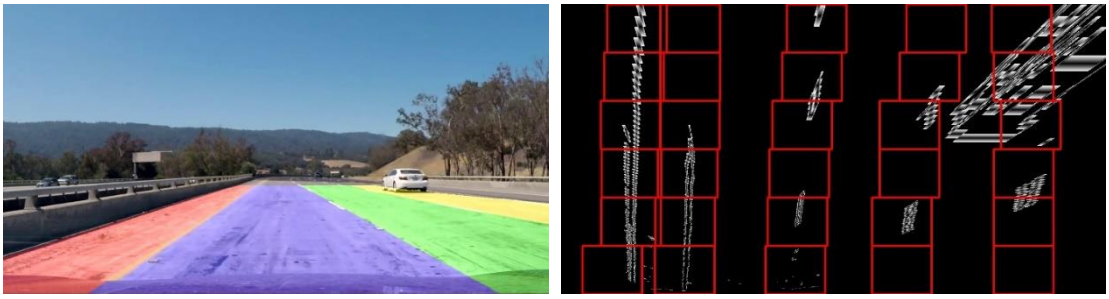
(a) Simple highway condition



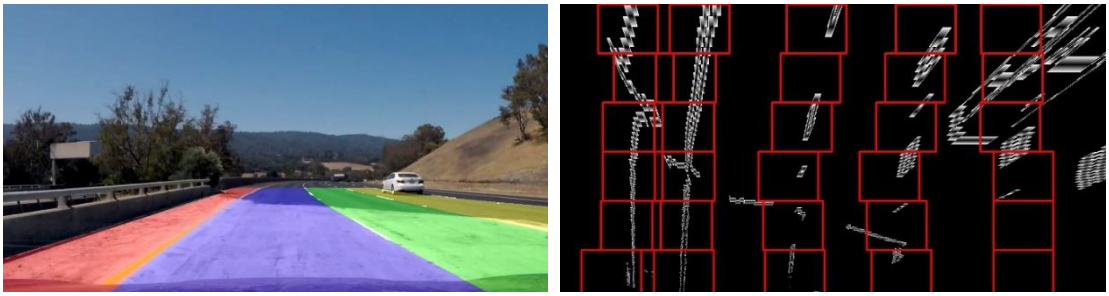
(b) With one car passing by



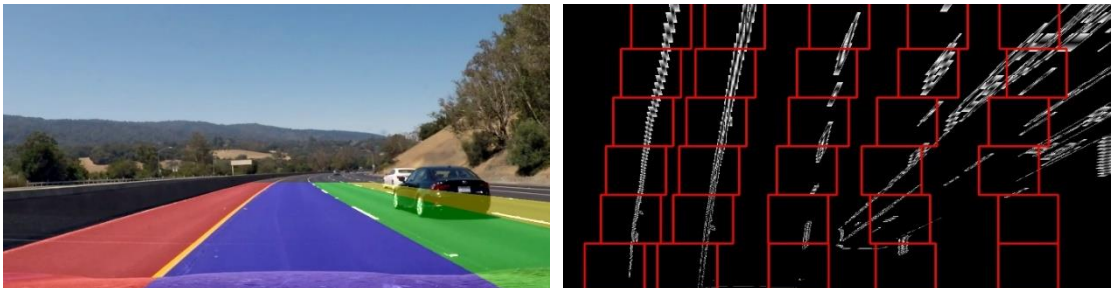
(c) Shadow of the tree with one car passing by



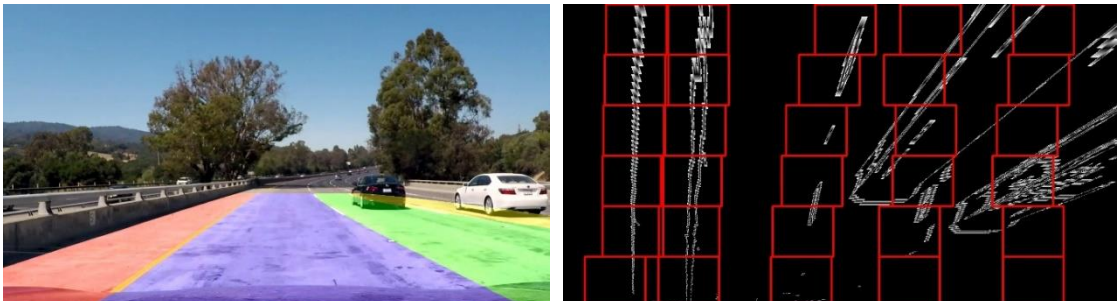
(d) Light road with one car passing by



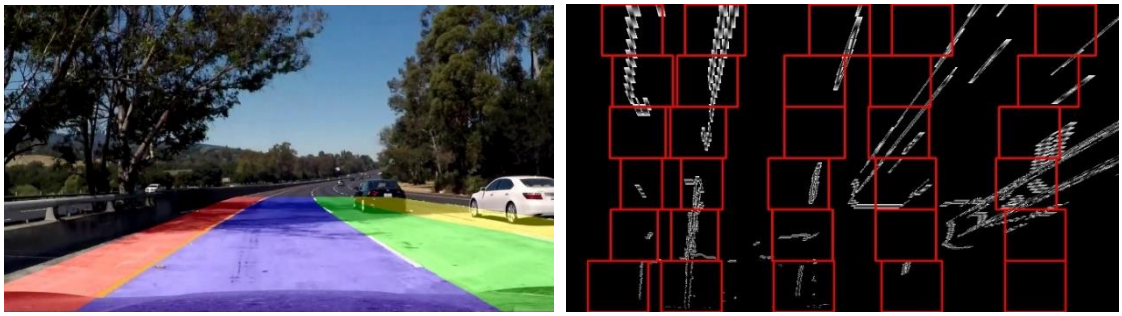
(e) Boundary of light road and dark road with one car passing by



(f) Large steering angle with two cars passing by



(g) Light road with two cars passing by



(h) Dense shadow of trees and the boundary of light road and dark road with two cars passing by

Fig. 12. Outcome and illustration of the algorithm of different complex road conditions of highway testing environment

7.Conclusion

This paper has proposed two lane-detecting algorithms, namely the Equally-Separated Algorithm, the Updated-Window Algorithm. After fitting into specific curve models, the real-time estimation of the lane in highway environment is achieved. Both two algorithms run robustly on highway, while the Updated-Window Algorithm is more computationally efficient and manages to process more details complex conditions, such as the lane with dense shadow of trees, with light color which has less gradient intensity to yellow lines, with sudden change of color from dark to white and white to dark, with large steering angle and so on. To see the potential of the Updated-Window Algorithm, a multi-lane detection algorithm is developed and manage to detect four lanes simultaneously on the highway testing environment. The multi-lane detection algorithm inherits all advantages of the Updated-Window Algorithm. Moreover, it is quite robust to passing-by vehicles on adjacent lanes. Based on the Updated-Window Algorithm, we test the scope of its application with an even more complex road condition, the mountain road environment. The outcome is perfect when the road is under a constant light condition, but it is sometimes undesirable when the road is exposed to inconstant light conditions, such as dispersive sunlight spots and dense shadow of plants. To overcome this limitation, we will further improve the algorithm with lane tracking, by adding the time dimension to establish the relationship between each frame. We will also establish a Convolutional Neural Network (CNN) based model so as to enable the algorithm adapting to more complex environments.

References

[1] Zhu Xiao, Xiangyu Shen, Fanzi Zeng, Vincent Havyarimana, Dong Wang, Weiwei Chen, and Keqin Li. Spectrum Resource Sharing in Heterogeneous Vehicular Networks: A Non-Cooperative Game-Theoretic Approach with Correlated Equilibrium, IEEE Transactions on Vehicular Technology, Oct. 2018, Vol. 67, No. 10, pp: 9449-9458.

[2] Zhu Xiao; Pingting Li; Vincent Havyarimana; Hassana Maigary Georges; Dong Wang; Keqin Li. GOI: A Novel Design for Vehicle Positioning and Trajectory Prediction under Urban Environments. IEEE Sensors Journal, July, 2018, Vol. 18, No. 13, pp: 5586-5594.

[3] Dong Wang; Jiaojiao Fan ; Zhu Xiao; Hongbo Jiang ; Hongyang Chen ; Fanzi Zeng ; Keqin Li. Stop-and-Wait: Discover Aggregation Effect Based on Private Car Trajectory Data. IEEE Transactions on Intelligent Transportation Systems, 2018.

[4] World Report on Road Traffic Injury Prevention, World Health Org.,Geneva, Switzerland, 2009.

[5] M. Caner Kurtul, Road Lane and Traffic Sign Detection & Tracking for Autonomous Urban Driving," Bogazici University, Master Thesis 2010.

[6] H. Gomez-Moreno, S. Maldonado-Bascon, P. Gil-Jimenez, and S. Lafuente-Arroyo, Goal evaluation of segmentation algorithms for traffic sign recognition, IEEE Trans. Intell. Transp. Syst., vol. 11, no. 4, pp. 917930, Dec. 2010.

[7] A. Doshi and M. Trivedi, On the roles of eye gaze and head dynamics in predicting drivers intent to change lanes, IEEE Trans. Intell. Transp. Syst., vol. 10, no. 3, pp. 453462, Sep. 2009.
[8] U. Ozgunalp, R. Fan, X. Ai, and N. Dahnoun, Multiple lane detection algorithm based on novel dense vanishing point estimation, IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 3, pp. 621632, 2017.
[9] Taqi Tahmid, Eklas Hossian, Density Based Smart Traffic Control System Using Canny Edge Detection Algorithm For Congregating Traffic Information, 3rd International Conference On Electrical Information and Communication Technology (EICT), 978-1-5386-2307-7/17/ 2017, IEEE.
[10] J. Wang, Y. Wu, Z. Liang, and Y. Xi, Lane detection and tracking using a layered approach, in Proc. IEEE Int. Conf. Inf. Autom., Harbin, China, Jun. 2010, pp. 17351740.
[11] Z. Kim, Robust lane detection and tracking in challenging scenarios, IEEE Trans. Intell. Transp. Syst., vol. 9, no. 1, pp. 1626, Mar. 2008.
[12] G.Wang, C.Lin, and S.Chen, Applying fuzzy method to visionbased lane detection, Exp. Syst. Appl., vol. 37, no. 1, pp. 113126, Jan. 2010.
[13] H. Yoo, U. Yang, and K. Sohn, Gradient-enhancing conversion for illumination-robust lane detection, IEEE Transactions on Intelligent Transportation Systems, vol. 14, pp. 1083 1094, September 2013.
[14] C. Li, B. Dai, R. Wang, Y. Fang, X. Yuan, and T. Wu, Multi-lane detection based on omnidirectional camera using anisotropic steerable filters, Iet Intelligent Transport Systems, vol. 10, no. 5, pp. 298307, 2016.
[15] M. Nieto and L. Salgado, Real-time vanishing point estimation in road sequences using adaptive steerable filter banks, Advanced Concepts for Intelligent Vision Systems, LNCS, pp. 840848, 2007.
[16] Son, J., Yoo, H., Kim, S., & Sohn, K. (2015). Real-time illumination invariant lane detection for lane departure warning system. Expert Systems with Applications, 42(4), 1816-1824.
[17] Gaikwad, V.; Lokhande, S. Lane Departure Identification for Advanced Driver Assistance. IEEE Trans. Intell. Transp. Syst. 2014, 16, 19.
[18] E. Shang, J. Li, X. An, and H. He. Lane Detection Using Steerable Filters and FPGA-based Implementation. 2011 Sixth International Conference on Image and Graphics, pages 908913, Aug. 2011.
[19] K. Y. Guo, E. G. Hoare, D. Jasteh, X. Q. Sheng, and M. Gashinova, Road edge recognition using the stripe Hough transform from millimeterwave radar images, IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 825833, Apr. 2015.
[20] Yingping Huang, Yangwei Li, Xing Hu and Wenyan Ci, "Lane Detection Based on Inverse Perspective Transformation and Kalman Filter," KSII Transactions on Internet and Information Systems, vol. 12, no. 2, pp. 643-661, 2018.
[21] Mustafa Eren Yıldırım and Yücel Batu Salman, "Directional Particle Filter Using Online Threshold Adaptation for Vehicle Tracking," KSII Transactions on Internet and Information Systems, vol. 12, no. 2, pp. 710-726, 2018.
[22] Ning Ye, Yingya Zhang, Ruchuan Wang and Reza Malekian, "Vehicle trajectory

prediction based on Hidden Markov Model," KSII Transactions on Internet and Information Systems, vol. 10, no. 7, pp. 3150-3170, 2016.
[23] Sungsoo Lim, Daeho Lee and Youngtae Park, "Lane Detection and Tracking Using Classification in Image Sequences," KSII Transactions on Internet and Information Systems, vol. 8, no. 12, pp. 4489-4501, 2014.
[24] Gupta, A. and Merchant, P.S.: Automated Lane Detection by K-means Clustering: A Machine Learning Approach, Electronic Imaging, Vol. 14, (2016) 16.
[25] Hiren M. Mandalia and Dario D. Salvucci. Using support vector machines for lane change detection. In In Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting, 2005.
[26] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016.
[27] J. Li, X. Mei, and D. Prokhorov. Deep Neural Network for Structural prediction and Lane Detection in Traffic Scene. IEEE Transactions on Neural Networks and Learning Systems, PP(99):114, 2016.
[28] Bing-Fei Wu and Jhy-Hong Juang, "Real-Time Vehicle Detector with Dynamic Segmentation and Rule-based Tracking Reasoning for Complex Traffic Conditions," KSII Transactions on Internet and Information Systems, vol. 5, no. 12, pp. 2355-2373, 2011.
[29] S. Cheng and M. Trivedi, Lane tracking with omnidirectional cameras: Algorithms and evaluation, EURASIP J. Embedded Syst., vol. 2007,no. 1, p. 5, Jan. 2007.
[30] Seung-Nam Kang, Soomok Lee, Junhwa Hur, and Seung-Woo Seo, Multi-lane detection based on accurate geometric lane estimation in highway scenarios, in Intelligent Vehicles Symposium Proceedings, 2014 IEEE, June 2014, pp. 221226.
[31] M. Nieto, A. Corts, O. Otaegui, J. Arrspide, and L. Salgado, Real-time lane tracking using Rao-Blackwellized particle filter, Journal of Real-Time Image Processing, 2012
[32] Deok-Kwon Lee and Ju-Seok Shin, ”Real-Time Lane Detection and Tracking System Using Simple Filter and Kalman Filter”, 2017 Ninth International Conference on Ubiquitous and Future Networks, pp. 275-277, Jul. 2017.
[33] M. Nieto, A. Corts, O. Otaegui, J. Arrspide, and L. Salgado, Real-time lane tracking using Rao-Blackwellized particle filter, Journal of Real-Time Image Processing, 2012