國立交通大學

資訊科學與工程研究所

碩 士 論 文

車 輛 影 片 中 車 道 線 之 偵 測 及 追 蹤

A Novel Lane Line Detection and Tracking System of Car Videos

研 究 生：陳姿延

指導教授：李素瑛　教授

陳華總　教授

中 華 民 國 一 百 零 一 年 七 月

車輛影片中車道線之偵測及追蹤

A Novel Lane Line Detection and Tracking System of Car Videos

研 究 生：陳姿延　　Student：Tsu-Yen Chen

指導教授：李素瑛、陳華總　Advisor：Suh-Yin Lee, Hua-Tsung Chen

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

July 2012

Hsinchu, Taiwan, Republic of China

中華民國一百零一年七月

車輛影片中車道線之偵測及追蹤

研究生：陳姿延　　指導老師：李素瑛 教授

陳華總 教授

國立交通大學資訊科學與工程研究所

摘　要

近年來為了減少交通事故而發展的車輛輔助安全駕駛議題越來越受重視。其中，車道線偵測在車輛輔助安全系統中是一項必需的元件。在這篇論文中，我們提出一個利用 time-slice images 來進行偵測及追蹤車道線的系統。另一方面，由於虛線較為稀疏，不容易在畫面上偵測到，我們提出梯度值調整方法來提高虛線在畫面上的辨識度。每一張影像經過前置處理、邊緣偵測、峰點找尋及連結找出候選的車道線，接下來利用 time-slice images 進行車道線驗證，找出真正的車道線位置，最後在 time-slice images 上預測這些車道線在新畫面的可能位置，以進行進一步的追蹤。由於我們只針對影像中某幾行進行處理，且我們利用追蹤的方法提供了連續的車道線偵測，這會減少一張影像所需要的處理時間。

實驗中採用行車紀錄器所拍下的影片當作測試資料，對於三種情況下做車道線偵測：一是當從直線開到曲線時，二是當進行車道切換時，三是開在直線上時。我們實作相關研究之車道線偵測演算法並對所執行出來的結果及產生的問題進行討論。實驗結果顯示，我們提出的方法確實能協助某些情況下車道線的偵測。

關鍵字：影像處理、機器視覺、車道線偵測、車道線追蹤

A Novel Lane Line Detection and Tracking System of Car Videos

Student: Tsu-Yen Chen

Advisor: Prof. Suh-Yin Lee

Prof. Hua-Tsung Chen

Institute of Computer Science and Engineering

National Chiao Tung University

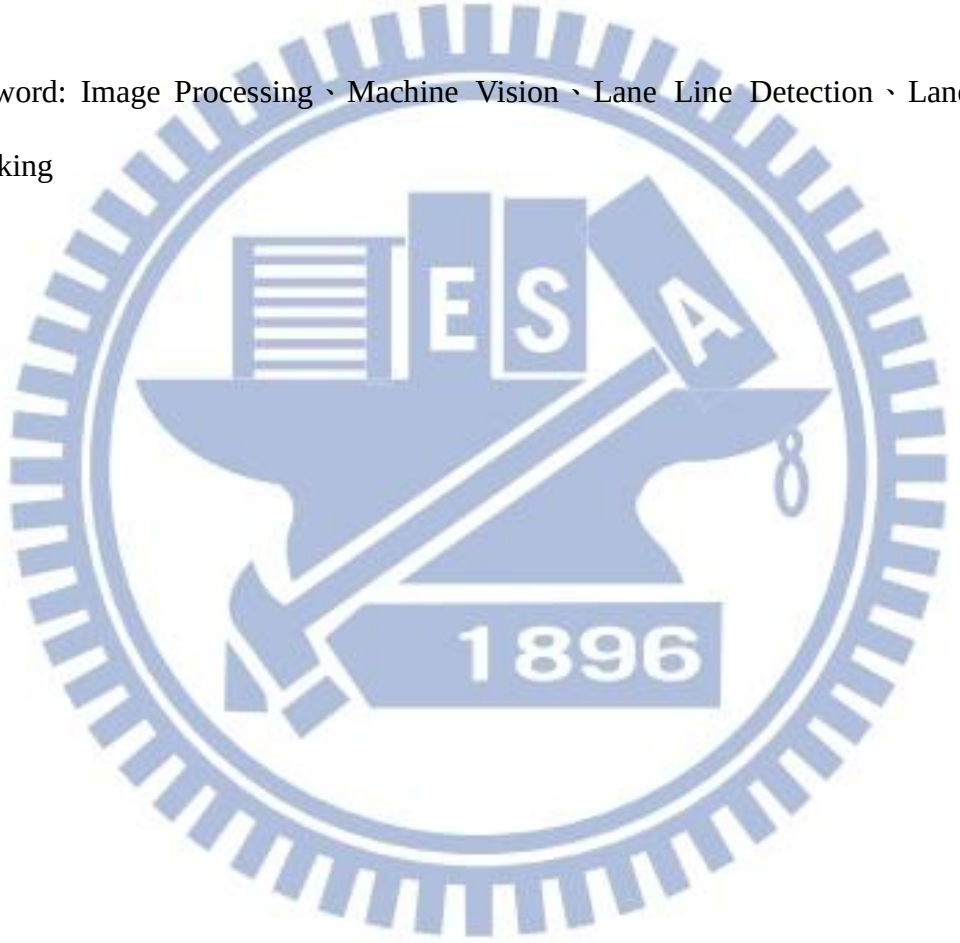
Abstract

In recent years, in order to reduce traffic accidents, developing Driver Assistance Systems for safety has attracted much attention. Lane line detection is an essential component of Driver Assistance System. In this thesis, we propose a lane line detection and tracking system utilizing the time slice images. On the other hand, due to the discontinuousness, dashed lane lines are difficult to detect in a single image. We propose the gradient value adjustment method to enhance the recognition of the dashed lane lines. For each frame, we find the candidate lane lines in the image through pre-processing, edge detection, and peak finding and connecting. Then we detect the true lane line positions by the time slice images in lane line verification. Once the lane line positions are located, we predict their new positions by the time slice images and track them in the new frame. Since we only process several rows of an image and we provide the continuous detection of the lane lines by tracking, the processing time of an image is reduced.

The experiments use the video sequences of real road scenes containing three

conditions: (1) the intermediate case of driving from the straight lane lines to curve lane lines, (2) the lane changing case, and (3) the straight lane lines case. We implement other lane line detection algorithms on the above cases, and then analyze the experimental results and discuss the problems arising in the experiment. However, the experimental results show that our proposed methods can improve the lane line detection in some cases.

Keyword: Image Processing、Machine Vision、Lane Line Detection、Lane Line Tracking

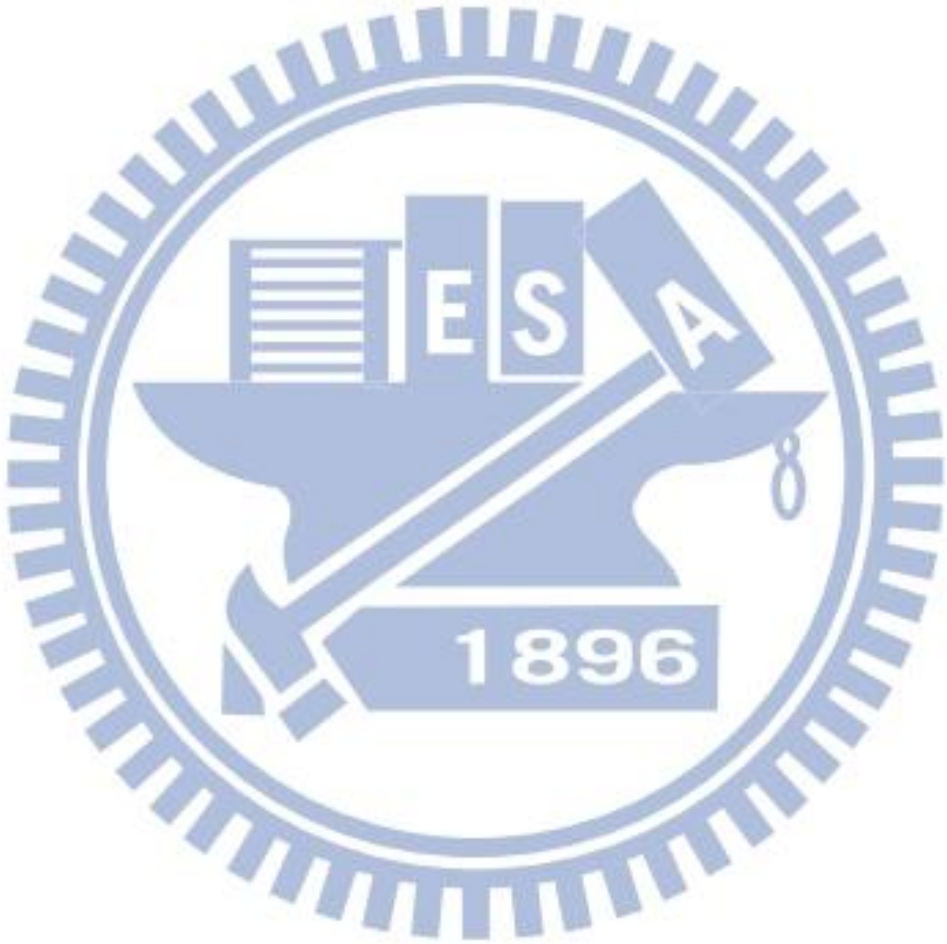


Acknowledgement

I greatly appreciate the patient guidance, the encouragement, and valuable comments of my advisors, Porf. Suh-Yin Lee and Porf. Hua-Tsung Chen. Without their graceful advices and help, I cannot complete this thesis. Besides, I want to extend my thanks to my friends and all members in the Information System laboratory, Hui-Zhen Gu, Yi-Cheng Chen, Chien-Peng Ho, Chien-Li Chou, Chun-Chieh Hsu, Yee-Choy Chean, Li-Wu Tsai, Tuan-Hsien Lee, Chao-Ying Wu, Wei-Zen Wang, Fan-Chung Lin, Li-Yen Kuo, Ming-Chu Chu, Kuan-Wei Chen, Yu-Chen Ho and Tzu-Hsuan Chiang especially. They gave me a lot of suggestions and shared their experiences.

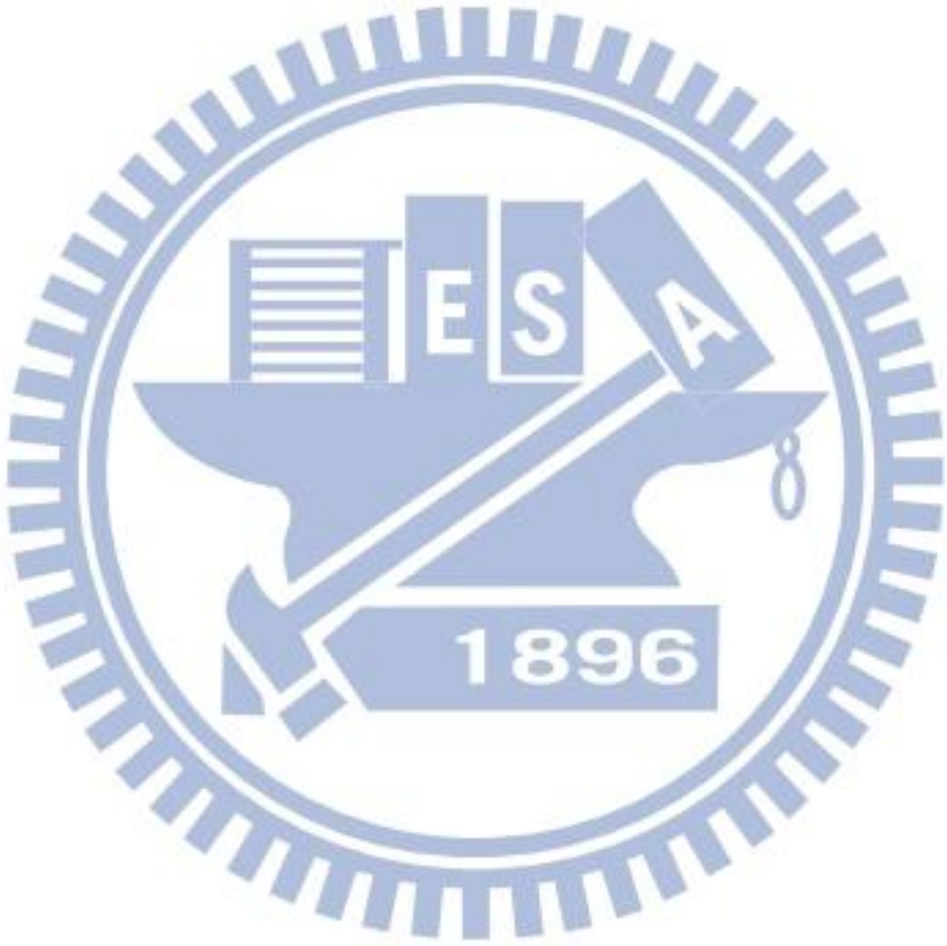
Finally, I would like to express my deepest appreciation to my dear family for their supports. This thesis is dedicated to them.

Table of Contents



List of Figures

List of Tables



Chapter 1. Introduction

1.1 Motivation and Overview

Recently, traffic accidents have become one of the most serious problems in today's world. However, the major factor which leads to road accidents is the carelessness of drivers or improper driving. Therefore, Advance Driver Assistance System (ADAS) [1] and Intelligent Transportation System (ITS) [2] are developed to improve the driving safety and reduce road accidents.

For the vision-based driving assistant systems, lane line detection plays an essential role in providing useful and effective information with respect to the relative position of the vehicle on the road. By means of this information, the driver can better understand the road circumstances and his/her driving situations for safety. So for decades, lane line detection has become a critical research field. However, in most conditions, vision-based lane line detection is simplified into a problem of finding the locations of lane lines in the input road images with or without strong prior knowledge about the lane line positions and drawing the results in the output images. Figure 1-1 shows an example of the lane line detection.

In order to analyze the lane line information from car videos, most lane line detection algorithms are based on image processing techniques to search for the lane lines. In general, the video analysis procedure comprises three major processing steps: (1) selection of the region of interest, (2) lane line detection, and (3) lane line tracking. Nevertheless, in most of the existing works, a fundamental problem is that the performance of video analysis may not be stable with varied environments and different weather conditions, as shown in Figure 1-2, resulting in the difficulty in lane

line detection. Besides, as shown in Figure 1-3, the presence of shadows, the lane line occluded by the vehicles and various markings on the road also affect the detection result. Moreover, most existing works processed the case of straight lane lines and curve lane lines, individually. Few researches discuss about the intermediate case of driving from the straight lane lines to curve lane lines, as shown in Figure 1-4, or lane changing, as shown in Figure 1-5. On the other hand, another critical issue is that image processing is always time-consuming. To cater for the requirement of real-time response, developing efficient system frameworks and video analysis algorithms is an important and inevitable task. Consequently, how to quickly and correctly locate the position of the lane lines from car videos is the core problem and issue in our work.

Figure 1-1 : An example of the lane line detection. (a) Input frame captured from the camera. (b) Output frame where the position of the lane lines is located.

Figure 1-2 : Different weather conditions. (a) Image captured under the sunny day. (b) Image captured under the cloudy day. (c) Image captured under the rainy day.

Figure 1-3 : Different driving environments. (a) The presence of shadows. (b) The lane line occluded by the vehicles. (c) Various markings on the road.

Figure 1-4 : An example of the intermediate case of driving from the (a) straight lane lines to (b) curve lane lines and then back to (c) straight lane lines.

Figure 1-5 : An example of lane changing from the left to right.

In this thesis, we describe our lane line detection and tracking system, then implement and compare several methods on the various cases, as shown in Figure 1-4 and Figure 1-5, and we discuss some problems arising in the course of our experiments. We mount the camera on the upper center of windshield of the vehicle for video capturing when driving. When inputting a car video, our proposed system

locates the vanishing point at first, and then instead of using the whole image, we only process some rows of the image as our rows of interest (ROIs). Slicing through the stack of some frames in time at a ROI generates a time slice image. With the time slice images and the vanishing point's position, we can detect the lane lines and track them on the time slice images. In the experiment, we implement several lane line detection algorithms and test them on real car videos captured at special environment conditions, then we discuss the problems arising in the experiment.

In conclusion, the contributions of this thesis are listed as follows:

- We adopt the peak finding algorithm to find out the points of the candidate lane lines, instead of giving a fixed threshold to classify the pixels in the image into non-lane-line and lane-line classes.
- We propose a gradient value adjustment algorithm to overcome the sparseness problem in detecting dashed lane lines.
- We propose a lane line detection and tracking algorithm using the "time slice images" which involve the lane lines' relationship between the time and space. Also, the time slice images can help us to detect the shape of curve lane lines without calculating the parameters of curve fitting.

1.2 Organization

The rest of this thesis is organized as follows. In Chapter 2, we survey some related works in lane line detection and tracking of car videos. Chapter 3 introduces our system to detect and track the position of lane lines. In Chapter 4, the experimental results of lane line detection and tracking are shown and discussed. At last, we conclude this thesis and describe the future work in Chapter 5.

Chapter 2. Related Works

In this chapter, some related works in lane line detection and tracking of car video are described. A distinction can be made between the problems of lane line detection and tracking. Lane line detection involves determining the location of the lane lines in a single image. Lane line tracking involves determining the location of the lane lines in a sequence of consecutive images, using information about the lane line location from previous images in the sequence to constrain the probable lane line detection in the current image. In each video, the first several image frames will be processed by the lane line detection algorithm, and this will provide a good estimation of lane line tracking for the subsequent frames.

2.1 Related Works in Lane Line Detection

As a basic yet important component in driver assistance systems [1][3], the aim of lane line detection is to detect the relative position of the vehicle on the road and to obtain the lane line information, such as the offset, the orientation, the curvature, and the types. With this lane line information, we can provide a better understanding of road environment to drivers and thus improve the driving safety. Generally, a road image can be classified into a structured or unstructured one, as shown in Figure 2-1(a) and (b), respectively. The most distinguishable characteristic between structured roads and unstructured roads is the existence of lane lines. The structured road has its boundary showing specific features or certain regularity in appearance. The structured road boundary has characteristics such as the painted white or yellow line(s), and regularly

shaped road edges. On the other hand, the boundary of the unstructured road does not have apparent features or regularity in its appearance. Most researches of lane line detection focus on the analysis of structured roads where the lane lines are painted on the road surface.

Figure 2-1 : Two types of road image. (a) Structured road image. (b) Unstructured road image.

The vision-based lane line detection is a complex and challenging task. It is well known that vision cues greatly vary depending upon lighting changes and weather conditions. Moreover, due to the appearance of various markings on the road and the different scenarios, for example, driving on an urban road [4][20] or on a highway [5], it is difficult to recognize the position of lane lines.

During the recent years, various weather conditions for vision-based lane line detection techniques for safety vehicles are taken into consideration and are being developed. There have been many approaches proposed for lane line detection. Those works usually use different road models, 2D [7][12] or 3D [4][6][15][17][19]. Different lane lines, straight [12][14][15][28] or curve [10][30][33], and different lane line patterns, solid or dash painted line [3]-[30], are considered. Different techniques are employed, Hough transformation [11][28], template matching [10][25], or neural networks [8]. The recent survey paper proposed by McCall and Trivedi [3] provides a

comprehensive summary of existing approaches. Most of the methods propose a three-step process. (1) Initializing lane lines by extracting features such as edges [7], texture [8], color [9], and frequency domain features [10]. (2) Post-processing the extracted features to remove outliers using techniques like Hough transform [11][28] and dynamic programming [12], along with computational models explaining the structure of the road using deformable contours [13], and regions with piecewise constant curvatures [6]. (3) Tracking the detected lane lines using the Kalman filter [14] or particle filter [15][16] by assuming motion models such as constant velocity or acceleration for the vehicle. However, these approaches can be classified into two main categories namely feature-based and model-based techniques [3]. Detailed surveys can be found in [3]-[30].

2.1.1 Feature-based Lane Detection

The feature-based methods detect the lane lines in the road images by using some low-level features, such as painted lines [17]-[19], lane line edges [7][22][23], texture [8] and colors [9][20][21]. The advantage of the feature-based methods is that the features are extracted easily. Nevertheless, they highly demand well-painted lane lines or strong lane line edges in road images; otherwise this method may fail. Furthermore, the performance of this method may easily suffer from occlusion or noise. In the following, we review some representative works of feature-based lane line detection.

Broggi and Bertè [17][18] develop an approach applying the *IPM* (Inverse Perspective Mapping) [42] algorithm on the road image. The *IPM* algorithm is a mathematical technique whereby a coordinate system may be transformed from one perspective to another, and it can remove the perspective effect from the acquired image. The perspective effect means that the lane lines width change according to

their distance from the camera. In this case, in order to remove the perspective effect, the *a-priori* knowledge exploited by the *IPM* transform is the assumption of a flat road in front of the vehicle. The *IPM* algorithm maps the acquired image, as shown in Figure 2-2(a), into a new 2-D domain array in which the information content is homogeneously distributed among all pixels and the resulting image represents a top view of the road region in front of the vehicle, as if it were observed from the top, as shown in Figure 2-2(b).

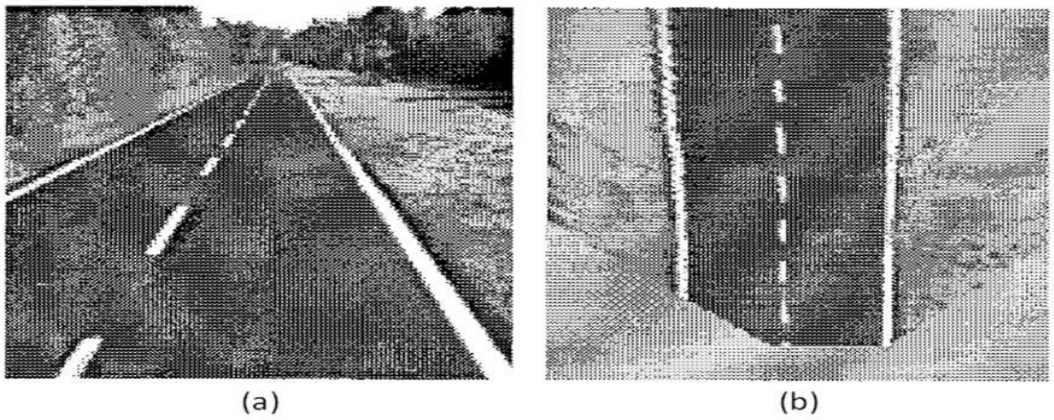


Figure 2-2 : An example of Inverse Perspective Mapping. (a) Original image. (b) Inverse perspective mapped image which seems to be observed from the top.

Here, the assumption used in the definition of a “lane line” is that a lane line is represented by a quasi-vertical bright line of constant width surrounded by a dark region (the road). Hence, the pixels belonging to a lane line have higher intensity values than their left and right neighbors. Thus, the lane line detection is reduced to the determination of horizontal black-white-black transitions. Based on a geometrical transform and on a fast morphological processing, the system is capable of detecting the lane lines. Figure 2-3 shows the results of lane line detection through the *IPM* algorithm.

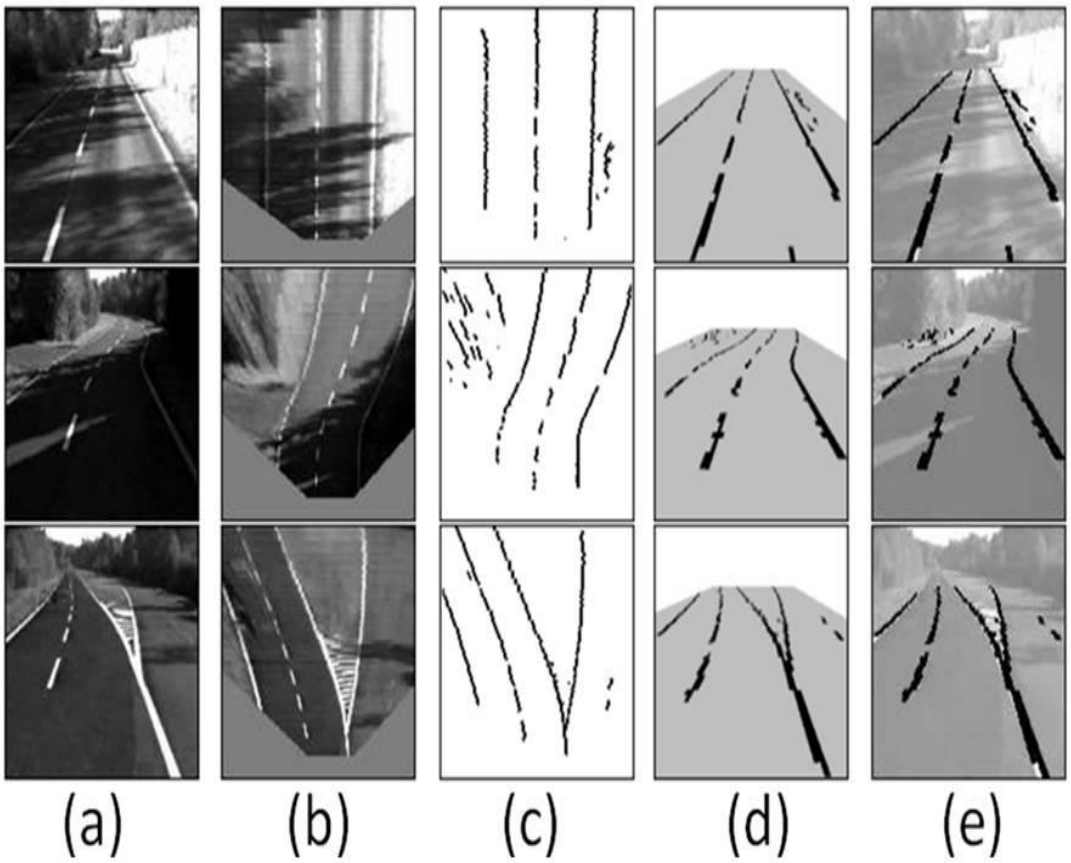


Figure 2-3 : Lane line detection through the removal of the perspective effect in three different conditions: straight road with shadows, curved road with shadows, junction. (a) Input image. (b) Mapped image (3D to 2D) of (a). (c) Result of the line-wise detection of black-white-black transitions in the horizontal direction. (d) Remapped image (2D to 3D) where the grey areas represent the portion of the image shown in (c). (e) Superimposition of (d) onto a brighter version of the original image (a). [18]

Bertozzi and Broggi [19] propose the *GOLD* (Generic Obstacle and Lane Detection) system utilizing a stereo vision-based hardware and software architecture, which aims at improving road safety of moving vehicles. The *GOLD* system removes the perspective effect also by *IPM* algorithm which maps the region ahead of the vehicle into the top view. In *GOLD*, lane lines after the *IPM* transform are modeled as quasi-vertical constant width lines, brighter than their surrounding region. Based on a line-wise determination of horizontal black-white-black transitions, the pixels that have higher intensity value than their horizontal neighbors at a given distance are

detected. However, in this work, not only the lane line detection is implemented but also the obstacle detection.

He *et al.* [20] propose a color-based vision system to determine the road parameters and detect lane lines from urban traffic scenes. Based on the projective transformation, edge detection, binarization and their pre-defined curvature models, as shown in Figure 2-4, this system estimates three candidate boundaries to extract the road region. The result of boundary estimation module is combined with the color information of the capture image to get the road area image. Finally, they utilize this road area image and three candidate boundaries to determine the real road boundaries and to acquire the parameters of road. Cheng *et al.* [21] apply the color of road and lane lines for the image segmentation, and then utilize the size, shape and motion characteristics to determine whether a region belongs to a vehicle or a lane line for false lane line region elimination. Huang and Pan [22] develop a method to detect the lane lines and the road edges of structured and unstructured roads, respectively. In structured road detection, utilizing the vertical Sobel mask and color characteristic at first to detect the points of lane lines. If the number of points is greater than a predefined threshold, the slope filtering is applied to refine the detection results. And the least square approximation is employed to represent the lane lines. The unstructured road detection is performed while the number of detected points is not enough. They extract the road surface with the predefined sets of sampled blocks, and obtain the edge points from the vertical intervals. For the accuracy of detection, the new sampled blocks are updated by the random sampling points from segmented regions.

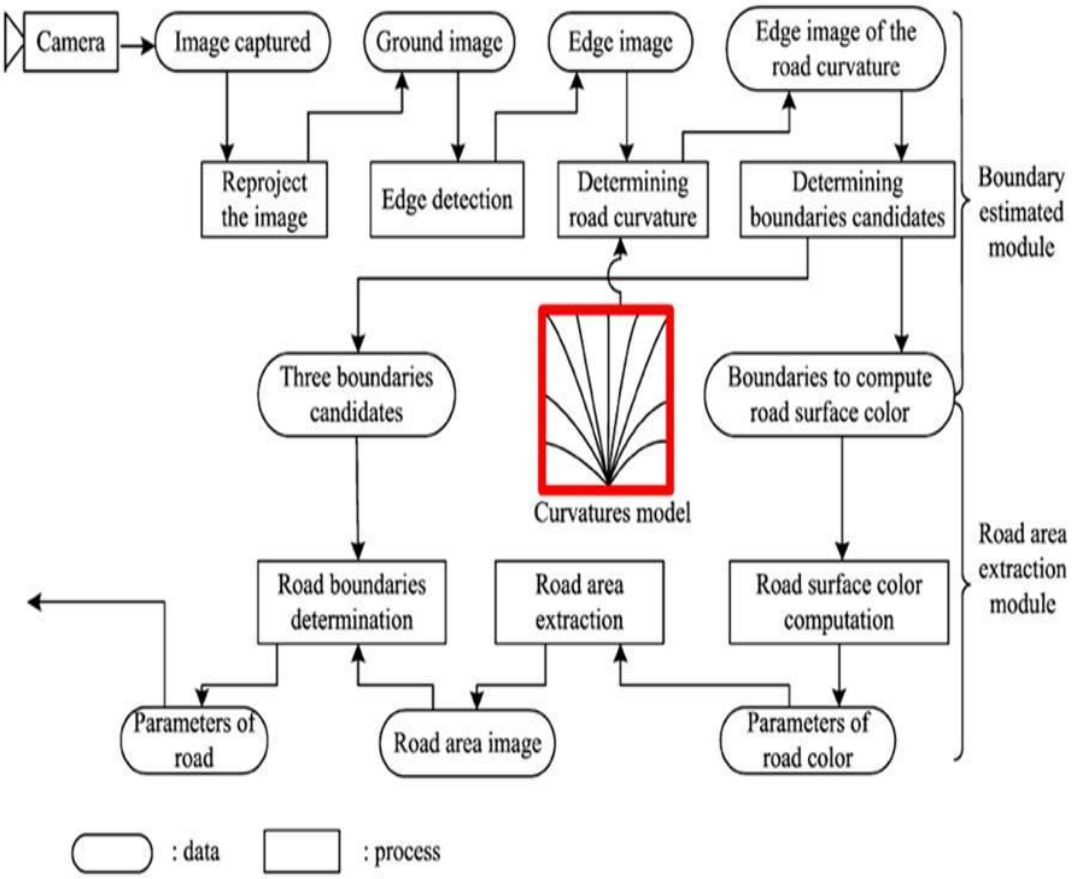


Figure 2-4 : The flowchart of He *et al.*[20]. (Red rectangle is represented as the curvature model they defined.)

Tsai *et al.* [23] propose a lane line detection algorithm using the concept of directional random walks based on Markov process. Two major components are included in this method to decide the correct locations of all lane lines: (1) lane segmentation and (2) edge linking. They first define proper structure elements to extract different lane line features from input frames using a novel morphology-based approach. Then, they utilize a novel linking technique to link all "desired" lane line features for lane lines detection. The technique considers the linking process as a directional random walk which constructs a Markov probability matrix for measuring the direction relationships between lane segments. Then, from the matrix of transition probability, the correct locations of all lane lines can be decided and found in videos. Yim and Oh [24] develop a three-feature based automatic lane line detection algorithm

(TFALDA). It is intended for automatic extraction of the lane lines without the priori information or manual initialization under different road environments. The lane lines are recognized based on similarity match in a three dimensional (3D) space consisting of the starting position, direction, and gray-level value of a lane line as features, as shown in Figure 2-5.

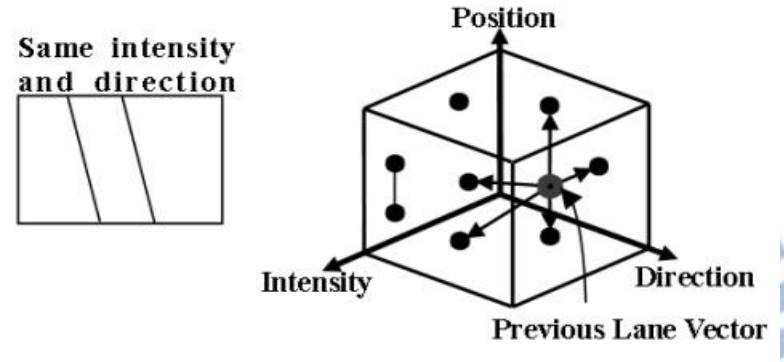


Figure 2-5 : The lane line candidate vectors mapped into the 3D feature space.

2.1.2 Model-based Lane Detection

Model-based methods represent the lane lines through a few geometric parameters. According to the shapes of lane lines, the lane line models can be defined as a straight line model [12][14][15][28] or a parabolic model (that is, curve) [10][30], even a spline model [4][13][29]. Moreover, how to find the best parameters for the model is the core problem to be solved. Compared with the feature-based methods, the model-based methods are less sensitive to weak lane line appearance features and noise.

To acquire the best parameters of lane line model, the likelihood function [10][25], the Hough transform [11][28], and curve fitting [30], had been applied into the lane line detection. However, the model-based methods require a complex modeling process involving much prior knowledge. Constructing a simple model for

one scene can get better efficiency, but this model may not work well in another scene because it cannot describe arbitrary shape of lane lines. So, the simple models are less adaptive. But for the complex models, although they can adapt to multiple scenes and describe arbitrary shape of lane lines, an iterative error minimization algorithm should be applied for the estimation of best model parameters, which is comparatively time-consuming. The process would take much time and would not satisfy the real time requirement of the driving applications. Next, we discuss some representative works of model-based lane line detection.

In [25], Kluge and Lakshmanan present a deformable template model of lane line structure, called Likelihood Of Image Shape (LOIS), to locate the lane lines by optimizing a likelihood function. It is assumed that the left and right lane lines are modeled as two parallel parabolas on the flat ground plane. For each pixel, this algorithm uses a Canny edge detector [26] to obtain the gradient magnitude and orientation. The parameters of perspective projection model are then estimated by applying the Maximum A Posteriori (MAP) estimation [27] and the Metropolis algorithm based on the image gradient. Figure 2-6 shows some results of LOIS' lane detection under the various road environmental conditions.

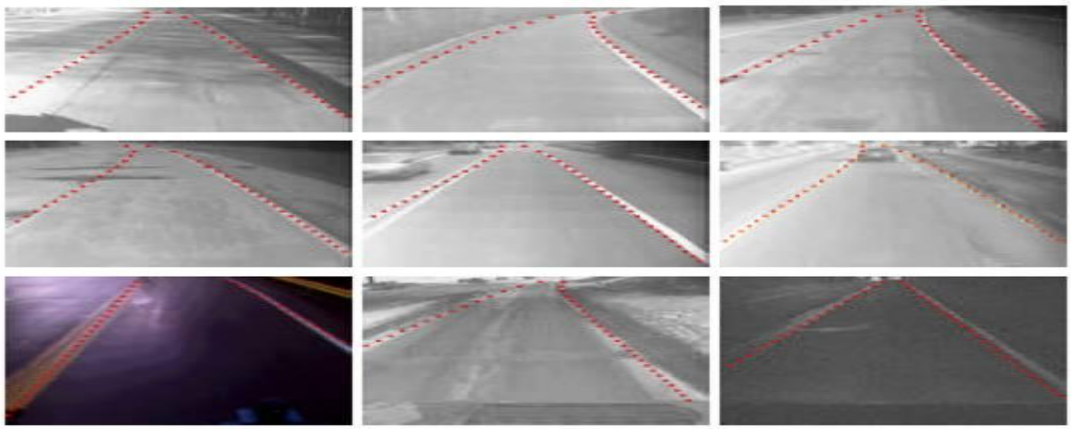


Figure 2-6 : Examples of LOIS' lane line detection [25].

The LANA system [10], proposed by Kreucher and Lakshmanan, is similar to the LOIS system [25] at the detection stage. But LANA combines the frequency domain features of the lane lines with a deformable template for finding the lane line edges. The feature vectors are used to compute the likelihood probability through fitting the detected features to a lane line model. Li *et al.* [11] develop the Springrobot system by using the color and edge gradient as the lane line features and the adaptive randomized Hough transform (ARHT) to locate the curve lane lines on the feature map. A multi-resolution strategy is applied to achieve an accurate solution rapidly and to decrease the running time to meet the real-time requirement. As illustrated in Figure 2-7, they first reduce the size of the original image to $1/2^z$, where z = 1, 2, by bicubic interpolation. The reduced images are called "half image" and "quarter image", respectively. In these images with lower resolution, they apply the ARHT with fixed quantized accuracy to roughly and efficiently locate the global optima of lane lines without regarding the accuracy. The parameters resulting from the previous step can be used as starting values of another ARHT for more accurate location of lane lines. Therefore, the parameter search can be restricted to a small area around the previous solution, saving time and storage complexities. This coarse-to-fine location speeds up the process of lane line detection, thus it offers an acceptable solution at an affordable computational cost. Figure 2-7 shows the results of multi-resolution algorithm in Springrobot system.

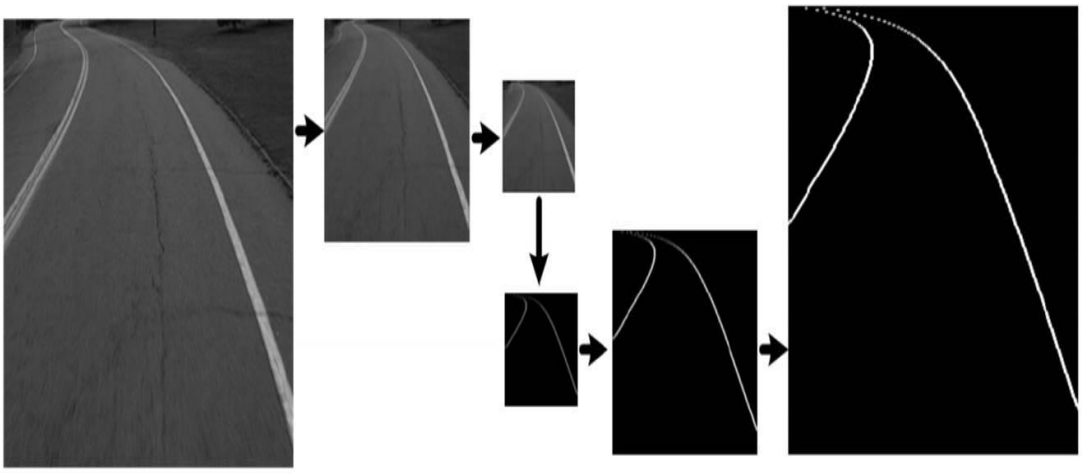


Figure 2-7 : Multiresolution algorithm for detecting the lane line rapidly and accurately [11], which uses the size reducing at first, then applies the ARHT to detect the parameters of lane lines roughly, and finally uses coarse-to-fine location method to offer the better position of lane lines.

Park *et al.* [30] use the lane-curve function (LCF) for lane line detection. The whole process of algorithm is shown in Figure 2-8. The LCF is obtained by transforming the defined parabolic function from the world coordinates into the image coordinates. Moreover, this algorithm needs no transformation of the image pixels into the world coordinates. The main idea of this algorithm is to search for the best-described LCF of the lane-curve on an image. In advance, several LCFs are assumed by changing the curvature and for each LCF, it defines its lane line region of interest. Then, the comparison is carried out between the slope of an assumed LCF and the phase angle of the edge pixels in the lane line region of interest. The LCF with the minimal difference in the comparison becomes the true LCF corresponding to the lane-curve.

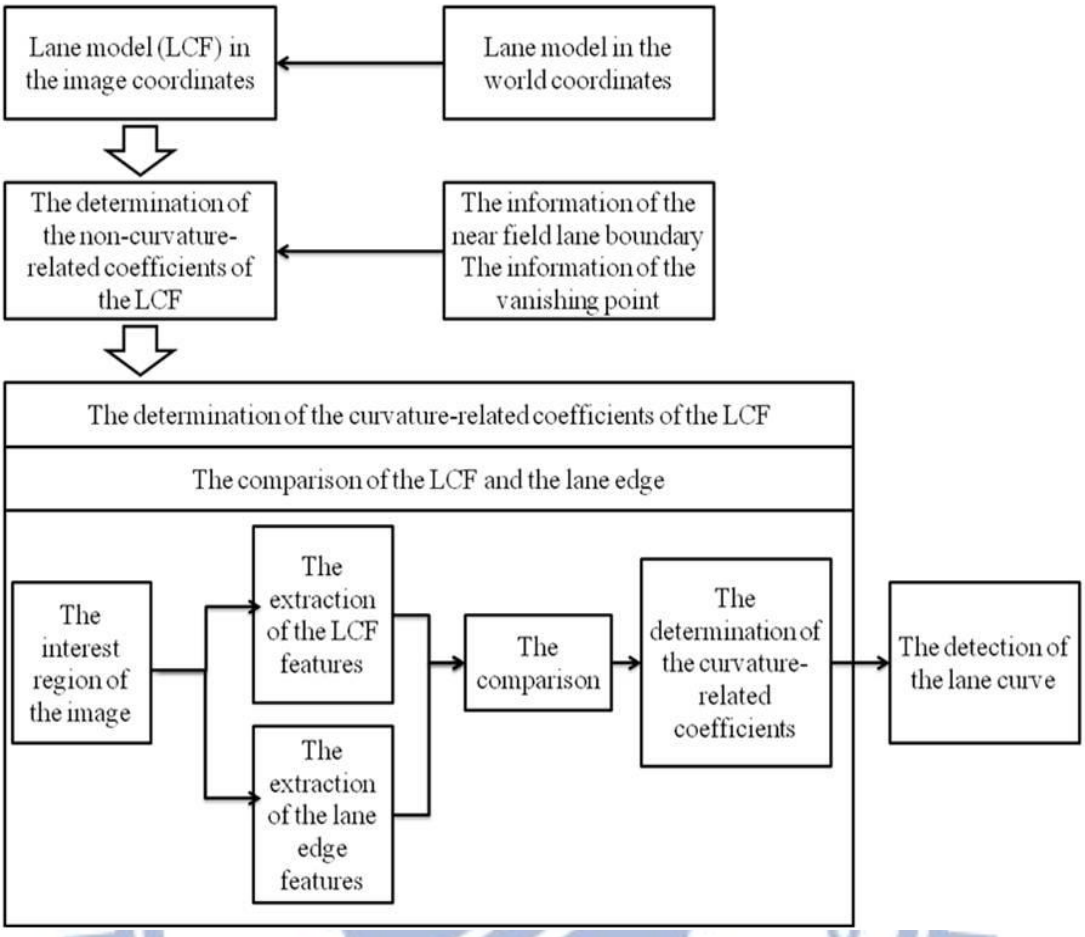


Figure 2-8 : The process overview of LCF [30].

Wang *et al.* [29] propose a Catmull-Rom spline [32] based lane line model. Their algorithm uses a maximum likelihood approach in detecting the lane lines. As Catmull-Rom spline model can form arbitrary shapes by control points, it can describe a wider range of lane line structures than the straight or parabolic model. Figure 2-9 shows the estimation of lane lines to real road image by implementing the Catmull-Rom spline algorithm. In [13], Wang *et al.* propose a B-Snake based lane line detection and tracking algorithm without any camera parameters. The main characteristics of this method are as follows. (1) The Canny/Hough Estimation of Vanishing Point (CHEVP) is presented for providing a good initial position for the B-Snake. (2) The Minimum Mean-Square Error (MMSE) is proposed to determine the control points of the B-Snake model by the overall image forces on two sides of the

lane. (3) The Gradient Vector Flow (GVF) [31] field is used to let the B-Snake move to its optimal solution. The estimation of lane lines by B-Snake is shown in Figure 2-10.

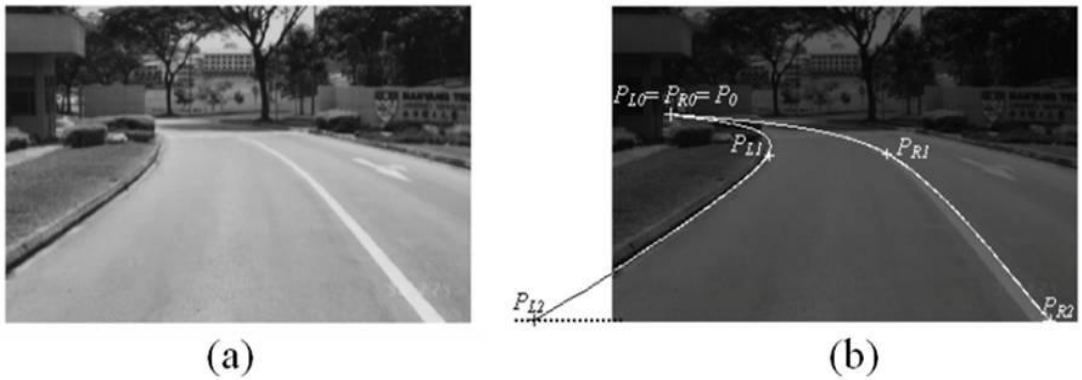


Figure 2-9 : An example of lane line detection by Catmull-Rom spline. (a) Original road image. (b) The result of lane line detection by Catmull-Rom splines. (P_{L0}, P_{L1}, P_{L2}) and (P_{R0}, P_{R1}, P_{R2}) are the control points for left and right side of lane line. P_{L0} and P_{R0} are the same control point, which supposes to be vanishing point. [29]

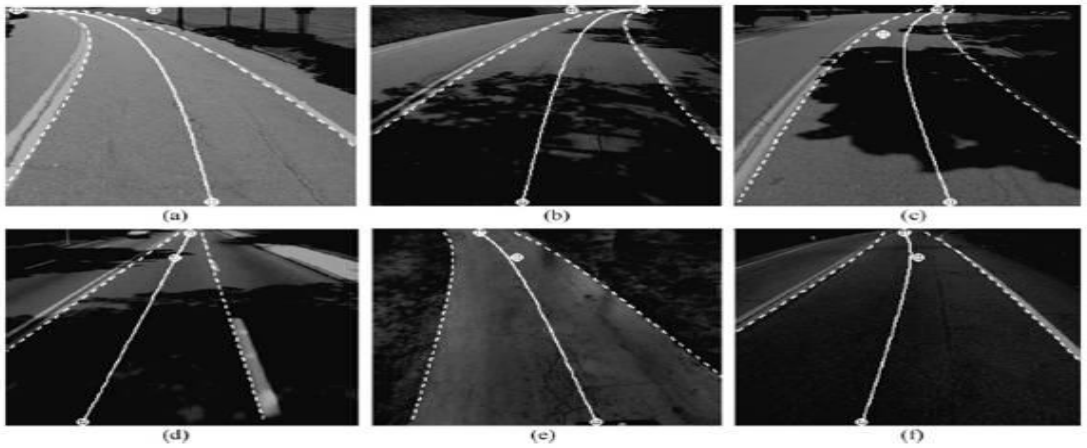


Figure 2-10 : Examples of lane line detection using the B-Snake. [13]

2.2 Related Works in Lane Tracking

However, some researches [7][8][10][12][20][23][30] do not mention about the idea of lane line tracking. They only propose the lane line detection algorithm on each

single frame and do not take into consideration about the relationship between two consecutive frames. However, most researches [6][9][14][15][16][21][28] usually include the lane line tracking into their systems for the purpose of the real-time requirement. Hence, considering there is only small change between two consecutive frames, those systems use information from previous results to facilitate the current detection. The Kalman filter [14] or the Particle filter [15][16] is the common method used to track the lane lines in videos since it can provide the continuous detection on all images in a sequence. Lane line tracking step can drastically reduce the search area in every frame and consequently detect lane lines in an efficient way.

In [14], when the lane lines are detected, the Kalman filters are used to track and smooth the estimates of parameters of lane lines based on the measurements. While tracking, if lane lines are intermittently not detected, then the Kalman filter relies on its prediction to produce estimates. However, if the lane lines go undetected for more than a few seconds, then tracking is disabled until the next detection. This is to avoid producing incorrect estimates when the lane lines do not appear on the road.

Kim [16] choose a particle-filtering algorithm over the Kalman filter to prevent the result from being biased too much on the predicted vehicle motion but to give more weight to the image evidence. Due to the vehicle's vibration and pitch change, the motion of the lane lines in world coordinates is not smooth enough to be properly modeled by a Kalman filter.

Although a lot of lane line detection and tracking algorithms are proposed, few researches mention about the intermediate case of driving from the straight lane lines to the curve lane lines, or the lane changing case. Therefore, we implement several algorithms in the intermediate case and the lane changing case. Then we discuss the problems arising in the experiments.

Chapter 3. Proposed System Architecture

This chapter describes the details of our proposed system. At first, an overview is given in Section 3.1, and the pre-processing is described in Section 3.2. Section 3.3 introduces the method to compute the vanishing point and set the row of interest (ROI) for subsequent processing steps. Then our proposed approach of lane line detection and verification is presented in Section 3.4, and finally, we explain the lane line tracking algorithm in Section 3.5.

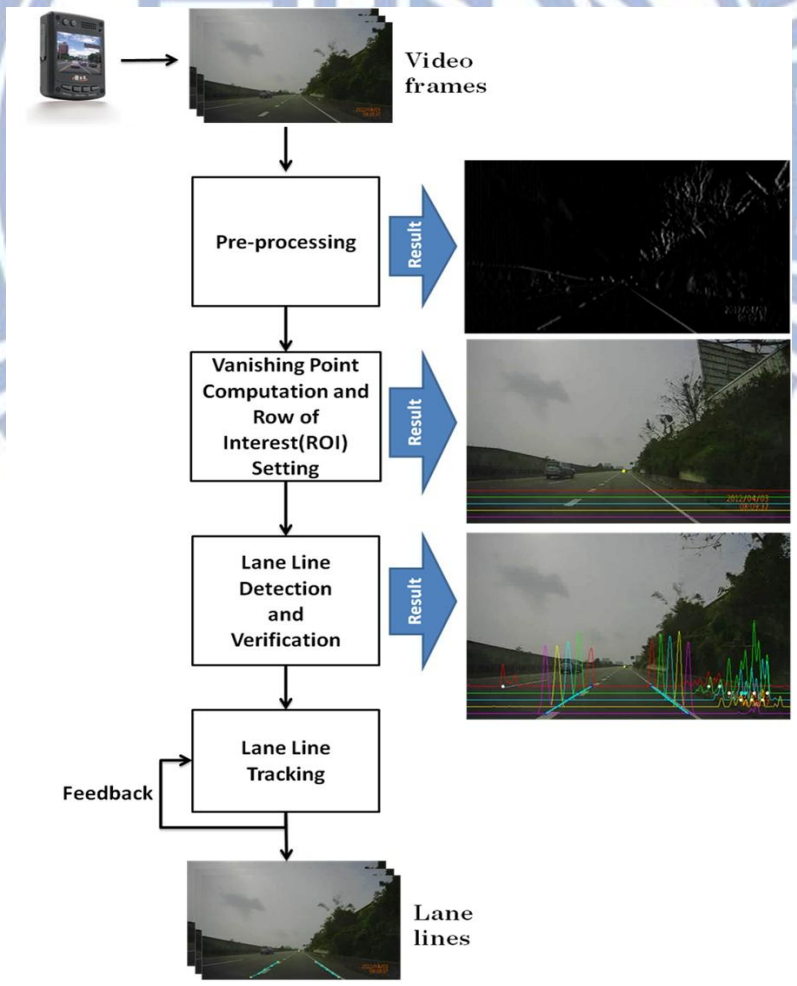


Figure 3-1 : The proposed system architecture.

3.1 Overview of the Proposed System

The overview of proposed system is depicted in Figure 3-1. The system architecture consists of four modules, including (1) *Pre-processing*, (2) *Vanishing Point Computation and Row of Interest (ROI) Setting*, (3) *Lane Line Detection and Verification*, and (4) *Lane Line Tracking*. For each step, we show the sample results on the right side.

For each image acquired from the camera, *Pre-processing* step for noise removal is firstly performed by image smoothing, image normalization and edge detection. In *Vanishing Point Computation and Row of Interest (ROI) Setting* step, the Hough transform and linear least square are applied in order to decide the possible position of the vanishing point and then utilize the vanishing point to delimit the rows of interest (ROIs) which we want to analyze in the following steps. Next, *Lane Line Detection and Verification* uses the gradient histogram of edge image generated from edge detection and some limitations to obtain the lane lines. Lastly, *Lane Line tracking* tracks the lane lines on the time-slice images generated from the ROIs.

3.2 Pre-processing

In this section, pre-processing is performed to reduce the noise and improve the contrast of the original image, then generate the corresponding edge image for subsequent processing. As illustrated in Figure 3-2, the original color image is first converted to the grayscale image. For image smoothing, we use the Gussian filter to eliminate the noise. Then in order to facilitate the extraction of lane lines, we use image normalization to increase the contrast of the image. At last, we extract the edge

features of the lane lines by edge detection.

Figure 3-2 : Flowchart of *Pre-processing* module.

3.2.1 RGB to Gray

At the beginning, the original images are composed of three independent channels for red, green and blue primary color components. Thus, for RGB to grayscale

conversion, we take three channel values of each pixel in the color image and use the conversion formula defined by Eq. (1) [37] to get the value for the corresponding pixel in the grayscale image. Pixels throughout the RGB image are scanned and this procedure is applied to convert a RGB image into grayscale one.

$$Gray = 0.299 * Red + 0.587 * Green + 0.114 * Blue \quad (1)$$

3.2.2 Image Smoothing

To realize the lane line detection, noise disturbance can greatly affect lane line distinction. However, the noise reduction techniques usually involve averaging the value of pixels residing in a local area and generating a blurred or smoothed image. Here, we apply the Gaussian filter [37] to eliminate the noise signal. As one of the specialized weighted averaging filters, the Gaussian filter has been widely adopted in the field of image processing and computer vision for years, and is known for its image smoothing and noise reduction capability.

3.2.3 Image Normalization

Since there are different environment conditions such as the presence of strong shadows, object reflection, illumination variation, and obscurity, the contrast enhancement of image intensity is essential. Image normalization [37] is a spatial domain based image enhancement technique. After image normalization, the distribution of pixels becomes more evenly spread out over the available pixel range. This step normalizes the brightness values of image in the range from 0 to 255, ensuring that the lane lines have high intensity value in every frame, even when the

overall brightness is changing. Figure 3-3 shows an example of image normalization.

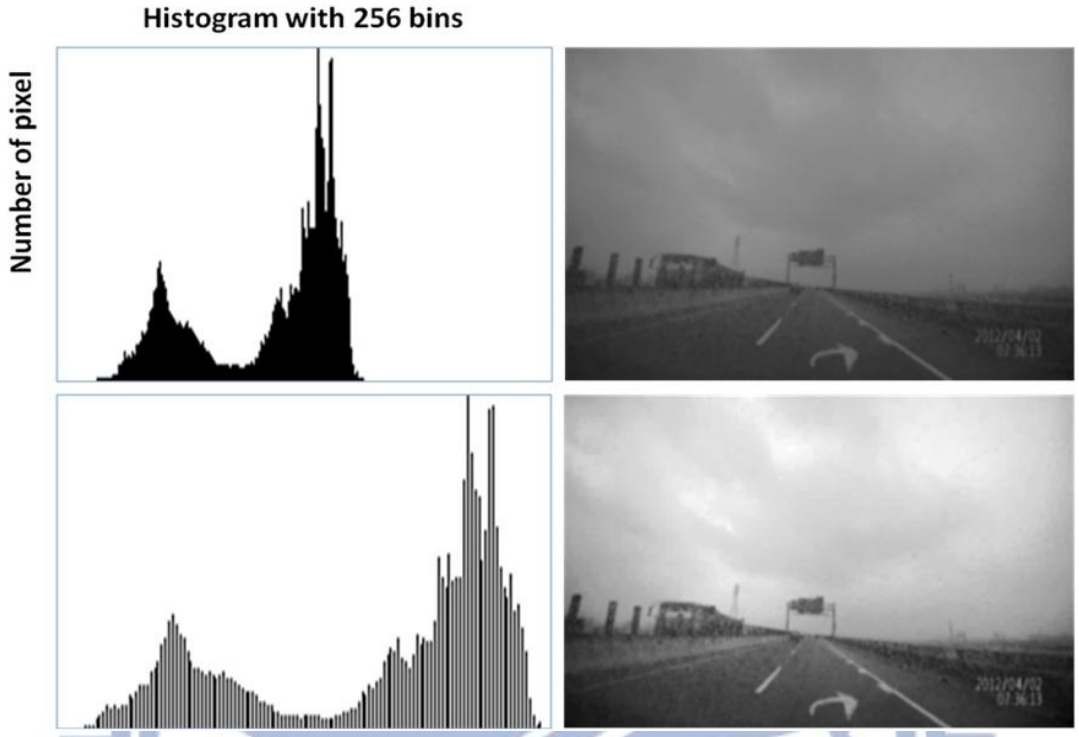


Figure 3-3 : An example of image normalization. Comparing the original histogram of image after smoothing (top) with the normalized histogram of image after normalization (bottom), one can observe that the range of pixel intensity values becomes broader.

3.2.4 Edge Detection

After smoothing and normalizing the image, we want to utilize some features to recognize the lane lines. In order to attract the drivers' attentions, a lane line is usually painted in a special color and owns high contrast (or high edge responses) to the neighboring road surface. Since color features are easily affected by light changes and become unclear at night, we tend to detect the lane lines based on the edge feature and acquire the edge information by Sobel edge detector. Since the horizontal gradient of the lane line is visible, the 3x3 operator for horizontal changes (G_x) is used, as shown in Figure 3-4(a). By applying G_x to each pixel of the image, the horizontal gradient

values of edge pixels are obtained. However, as shown in Figure 3-4(b) and (c), Sobel edge detector usually generates the positive and negative edges at the rim of the object. For computation efficiency, here we only retain the positive edge in the image. Figure 3-5 shows an example of the edge detection result.

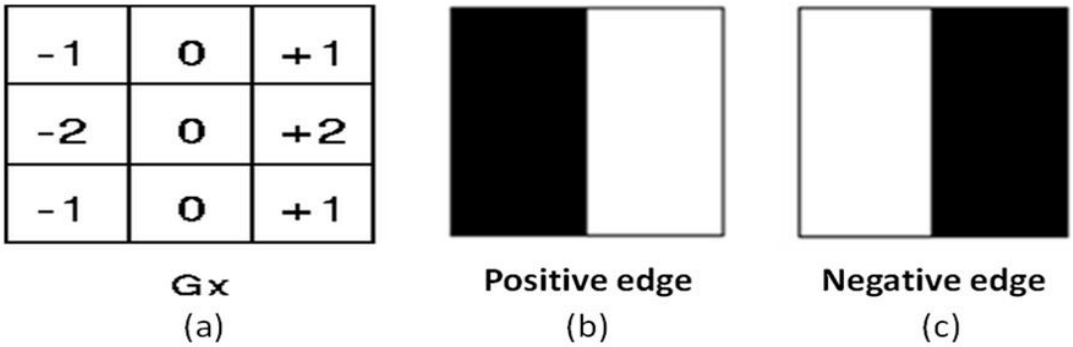


Figure 3-4 : Sobel edge detector. (a) Operator for horizontal changes (G_x). (b) Positive edge whose intensity change along x-direction is from dark to bright. (c) Negative edge whose intensity change along x-direction is from bright to dark.

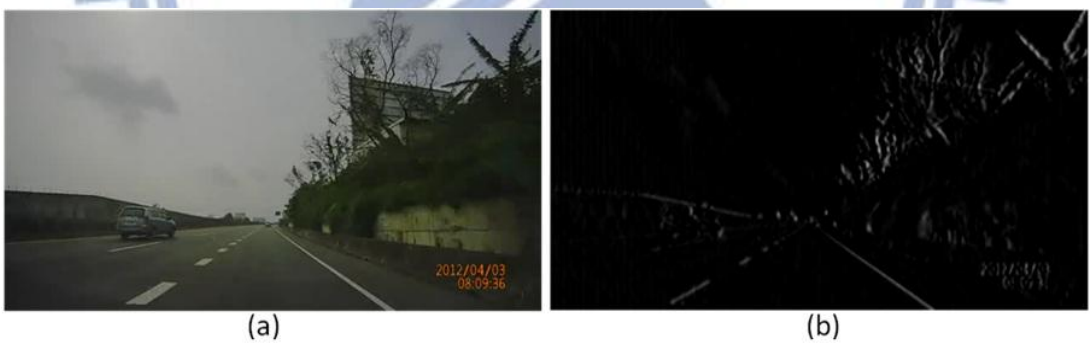


Figure 3-5 : An example of edge detection result. (a) Original image. (b) Result of edge pixels which have positive responses after using G_x.

3.3 Vanishing Point Computation and ROI Setting

In this section, we intend to locate the vanishing point and set the row of interest (ROI) according to vanishing point. As illustrated in Figure 3-6, we first use the Otsu

algorithm to binarize the edge image and then the Hough transformation is applied to extract the representative lines. For each frame, we use the linear least square to estimate the position of the vanishing point from those representative lines. After processing serveral frames, we can locate the position of vanishing point with highest probability. As soon as we get the vanishing point, the ROIs are also defined. The details of the module are described as follows.

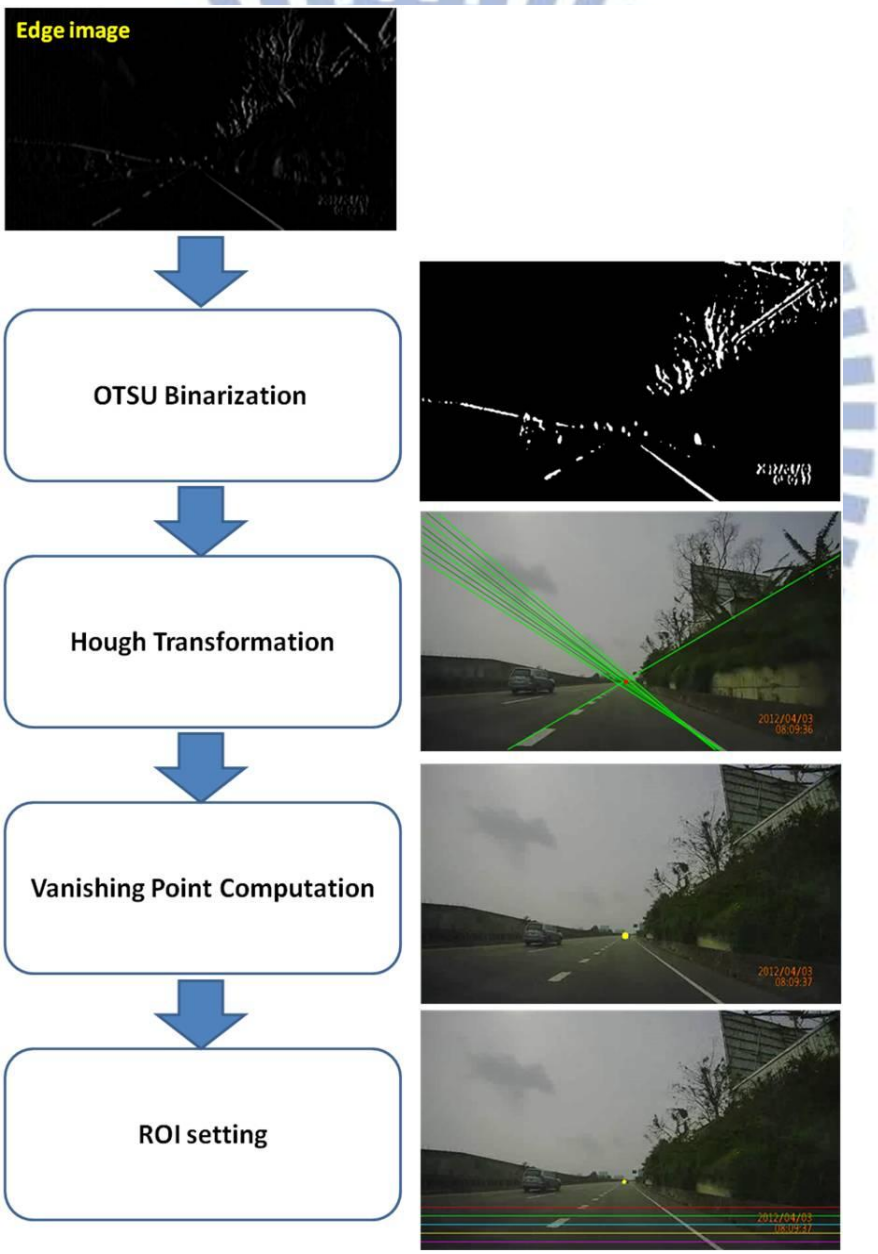


Figure 3-6 : Flowchart of *Vanishing Point Computation and ROI Setting* module.

3.3.1 Otsu Binarization

For the reason that the Hough transformation only accepts a binary image as input, thresholding is utilized here to segment the edge image. Nevertheless, as the lighting conditions are different, an adaptive threshold should be used in this stage. Otsu algorithm [33] is used to search for an ideal threshold adaptively.

The Otsu method exhaustively searches for the threshold which minimizes the intra-class variance, defined as a weighted sum of variances of the two classes. The thresholding process can be simplified into a process about how to partition the image pixels into two classes: C_1 = {0, 1, …, T} and C_2 = {T+1, T+2, …, N_{gl} -1}, where C_i indicates class, T is the chosen threshold and N_{gl} is the number of gradient levels of the image. The intra-class variance $\sigma^2_{\text{intra-class}}$ is defined as

$$\sigma^2_{\text{intra-class}}(T) = q_1(T)\sigma_1^2(T) + q_2(T)\sigma_2^2(T) \tag{2}$$

where $q_i(T)$ and $\sigma_i^2(T)$ indicates the proportion and gradient variance of the C_i pixels, respectively. The class probabilities are estimated as Eq. (3), the means of class are given by Eq. (4), and the individual class variances are defined as Eqs. (5) and (6).

$$q_1(T) = \sum_{i=1}^{T} H(i) \quad and \quad q_2(T) = \sum_{i=T+1}^{N_{gl}} H(i) \tag{3}$$

$$\mu_1(T) = \sum_{i=1}^{T} \frac{iH(i)}{q_1(T)} \quad and \quad \mu_2(T) = \sum_{i=T+1}^{N_{gl}} \frac{iH(i)}{q_2(T)} \tag{4}$$

$$\sigma_1^2(T) = \sum_{i=1}^{T} [i - \mu_1(T)]^2 \frac{H(i)}{q_1(T)} \tag{5}$$

$$\sigma_2^2(T) = \sum_{i=T+1}^{N_{gl}} [i - \mu_2(T)]^2 \frac{H(i)}{q_2(T)} \tag{6}$$

When the threshold T is chosen, the effect of edge image segmentation is obtained through reserving the edge pixels whose gradient levels exceed T. Figure 3-7 shows the result of Otsu algorithm.

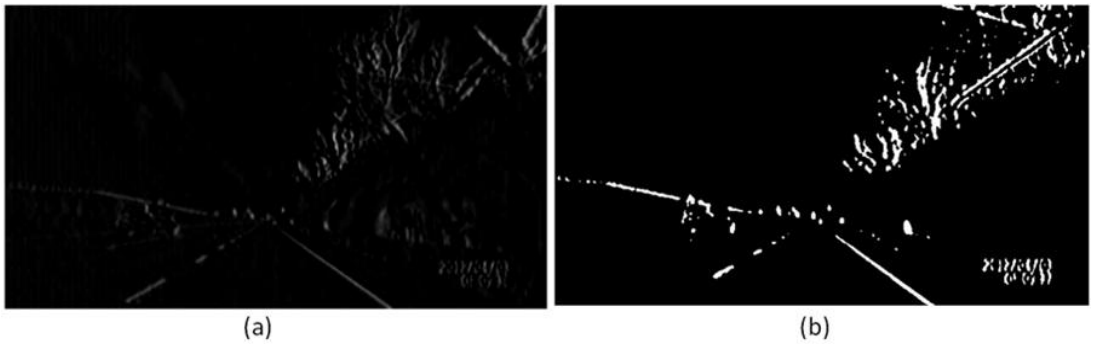


Figure 3-7 : Result of Otsu binarization. (a) Original edge image. (b) The corresponding binary image after using Otsu algorithm.

3.3.2 Hough transformation

Vanishing point is a point in the image plane, to which a set of parallel lines in the 3D space will converge [43]. In order to detect the most prominent straight lines in the image, we apply the Hough transform [34] to the binary image. Hough transform is the voting algorithm deciding whether there are enough pixels to form a particular shape in the image. In our case, we consider the straight lines. Each line has to be represented in the polar coordinates (ρ, θ), where ρ represents the distance from the origin to the line along a vector perpendicular to the line and θ is the angle between the x-axis and the vector perpendicular to the line, as shown in Figure 3-8(a), so that a generic point (x, y) belonging to a line will satisfy the following equation:

$$x\cos(\theta) + y\sin(\theta) = \rho \quad (7)$$

Therefore, by means of Hough transform, a line can be represented as a single point in the polar-coordinate parameter space. Similarly, since infinite lines pass

through any given pixel in the original image, the representation of a pixel in the parameter space is a unique sinusoidal curve (representing all the lines that can pass through that pixel). However, the point of intersection between multiple sinusoidal curves in the parameter space means the line passing through multiple pixels. In the other words, the more cumulative number of the intersection points, the more pixels a line passes through. As illustrated in Figure 3-8(b), the red point represents the line which passes through P_1 and P_2. The parameter space is divided into bins in the ρ and θ space. The total number of intersections in each bin is saved into the accumulator, and then the highest voted lines are returned. Here, in order to save the computation time, we only consider the top 5 lines in the accumulator.

In fact, most related works apply the Hough transformation to detect the lane lines [28][36], and show good performance of the results on the straight lane lines in the road images. However, the Hough transform-based methods can only detect the straight lane lines in the image and the case of driving on the curve lane lines cannot be handled well, as shown in Figure 3-9.

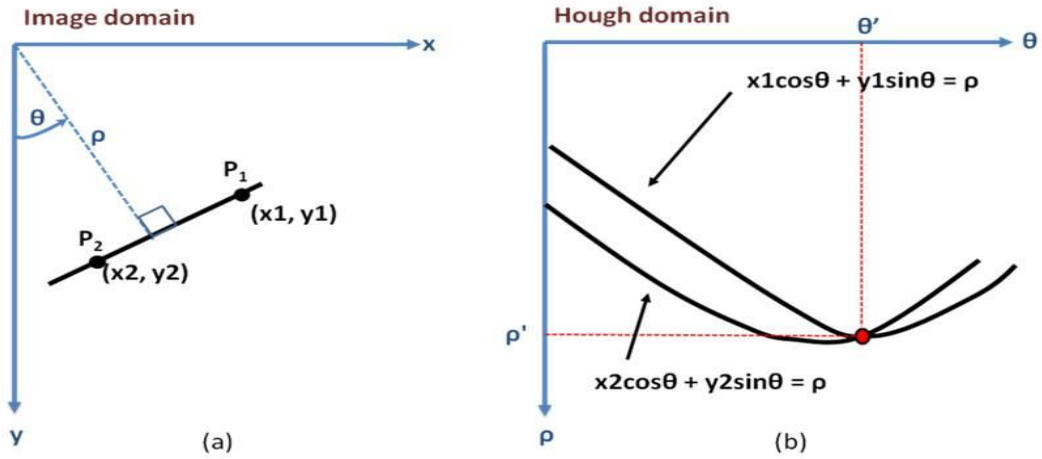


Figure 3-8 : Hough transformation. (a) Polar-coordinate (ρ, θ) representation of a straight line. Each line has a unique representation (ρ, θ). (b) The Hough domain of an image. The red point is the point with the highest number of intersections [38].

Figure 3-9 : The unreasonable representation of the curve lane lines. (a)(b) The curve lane lines in the red circle cannot be detected, and only the straight lane lines in the near field are detected.

There are other types of Hough transformation to recognize the shapes like circles and ellipses that are mathematically expressed in a binary digital image. When the parameters of the circles or ellipses are known in advance, the Hough transformation works well. Otherwise, it is difficult to recognize the curves without prior-knowledge. In addition, some works [6][10][14][16][25][30] utilize the curve fitting to detect the curves. Suppose each point (x, y) belonging to a curve will satisfy the following equation:

$$y = s_1 x^2 + s_2 x + s_3 \tag{8}$$

where s_j (j = 1, 2, 3) are the parameters of the curve. Then the idea of the curve fitting is to find the best parameters s_j (j = 1, 2, 3) which minimize E and E is defined as:

$$E = \sum_{i=1}^{k} [y_i - (s_1 x_i^2 + s_2 x_i + s_3)]^2 \tag{9}$$

where k is the total number of pixels used to fit a curve. This method is intuitive, but we have to know which pixels belong to the curve before curve fitting method. It is the limitation of this method. In addition, the more the number of detected pixels we use in curve fitting, the more time used in calculating the parameters we need.

For the reason mentioned above, we propose a novel algorithm to detect the

curve lane lines efficiently. We exploit the time-slice image inspired by [14] to detect and track the lane lines. Hence, we need to compute the position of the vanishing point at first.

3.3.3 Vanishing Point Computation

After acquiring several prominent lines in the image by Hough transformation, we use the "Linear Least Square" [40] to estimate the position of vanishing point in each frame. Our method is similar to the Vanishing Point Detection method in [44]. Now we have a linear system which involves several linear equations and several variables. A general system of m linear equations with n unknowns can be written as:

$$\begin{aligned} a_{11}u_1 + a_{12}u_2 + \ldots + a_{1n}u_n &= b_1 \\ a_{21}u_1 + a_{22}u_2 + \ldots + a_{2n}u_n &= b_2 \\ \vdots \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad \vdots & \\ a_{m1}u_1 + a_{m2}u_2 + \ldots + a_{mn}u_n &= b_m \end{aligned} \tag{10}$$

where u_i (i = 1, 2, …, n) are the unknowns, a_i (i = 11, 12, …, mn) are the coefficients of the system, and b_i (i = 1, 2, …, m) are the constant terms. A solution of the linear system is an assignment of values to u_i (i = 1, 2, …, n) that satisfies all m equations simultaneously. In matrix-vector notation, the linear system is represented as

$$AU = B \tag{11}$$

where $$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \text{ and } \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{12}$$

In our case, each linear equation determines a line on the xy-plane, so the n is equal to two. Next, we solve this linear system $AU=B$ with Singular Value Decomposition (SVD) [39] to obtain the closest possible solution U. This is

equivalent to minimize the squared norm $\| AU\text{-}B \|^2$, which is a linear least square optimization problem. Figure 3-10 shows the estimated vanishing point result (the red point) in current frame.

Figure 3-10 : The estimated vanishing point result. (a) Original image. (b) The green lines are represented as the prominent lines acquired from the Hough transformation and the red point is represented as the estimated vanishing point in current frame.

However, the vanishing point cannot be detected well in some frames. Hence, how to determine the best and correct vanishing point becomes the issue for us to conquer currently. We consider that the position of vanishing point should not move drastically during a car video, so we add the time conception to choose the correct vanishing point.

Observing N_{van} frames, we record all detected vanishing points and apply the voting method by an accumulator, whose concept is similar to the Hough transformation, to all vanishing points. When the Euclidean distance between two vanishing points is less than a pre-defined threshold, those two vanishing points are treated as the same point, and then their vote is incremented by 1. The highest voted point represents the correct vanishing point we want. The procedure of vanishing point computation is shown in Figure 3-11.

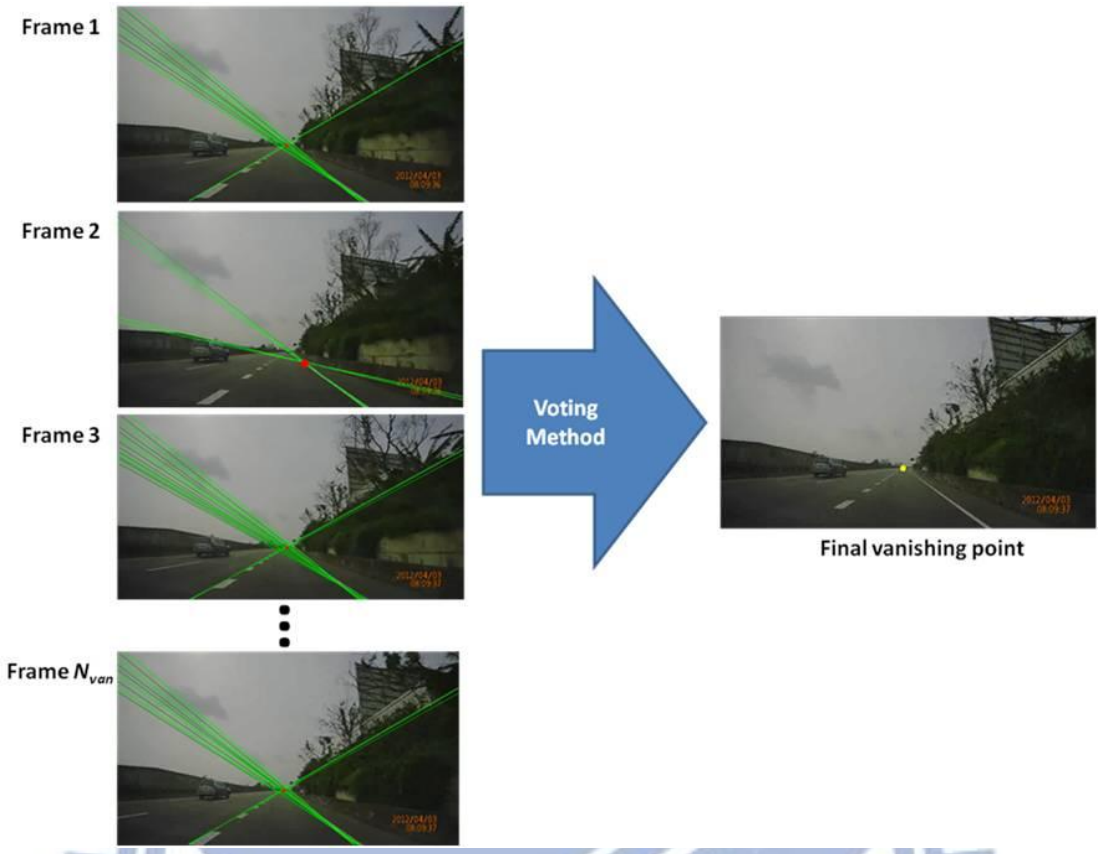


Figure 3-11 : Procedure of vanishing point computation. From N_{van} frames, choosing the highest voted point as the final result of the vanishing point which is drawn in yellow.

3.3.4 ROI Setting

Once the position of vanishing point is obtained, we delimit the rows of interest (ROIs) which are the main parts we want to process within the whole image. Generally, the road region appears under the vanishing point. Under this condition, our ROIs are selected from the region R_{region} under the vanishing point in the image. Nevertheless, instead of processing whole region R_{region}, we only take evenly N_{row} rows within the bottom three-quarter part of R_{region} as our ROIs. The illustration of ROI setting is presented in Figure 3-12.

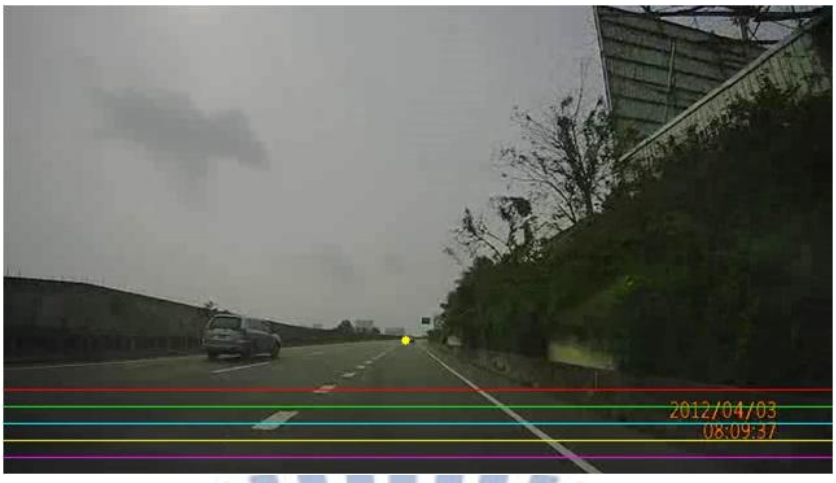


Figure 3-12 : Illustration of ROI setting. The yellow point indicates the final vanishing point, and the five rows (N_{row} = 5) in red, green, cyan, yellow, and purple are the selected ROIs.

3.4 Lane Detection and Verification

In this section, we detect the candidate lane lines and then a verification procedure is performed to remove the false ones. As illustrated in Figure 3-13, we first generate the time-slice image for each ROI to record the moving of lane lines. Then we adjust the edge gradient histogram of each ROI when a new frame comes for enhancing the detection of dashed lane lines. Next, using the smoothing method and peak finding algorithm on the gradient histogram to obtain the peak points. With these peak points, we utilize some constraints to connect the similar peak points together, and further detect the candidate lane lines. Generally, the lateral shift of the vahicle is small between two consecutive frames, that is, the difference of the lane line positions is small between two consecutive frames. Therefore, we can detect the lane line positions in current frame from the surrounding area of the lane line position in the last frame. At last, we verify the candidate lane lines by the concept similar to tracking. The details of this module are described in the following sections.

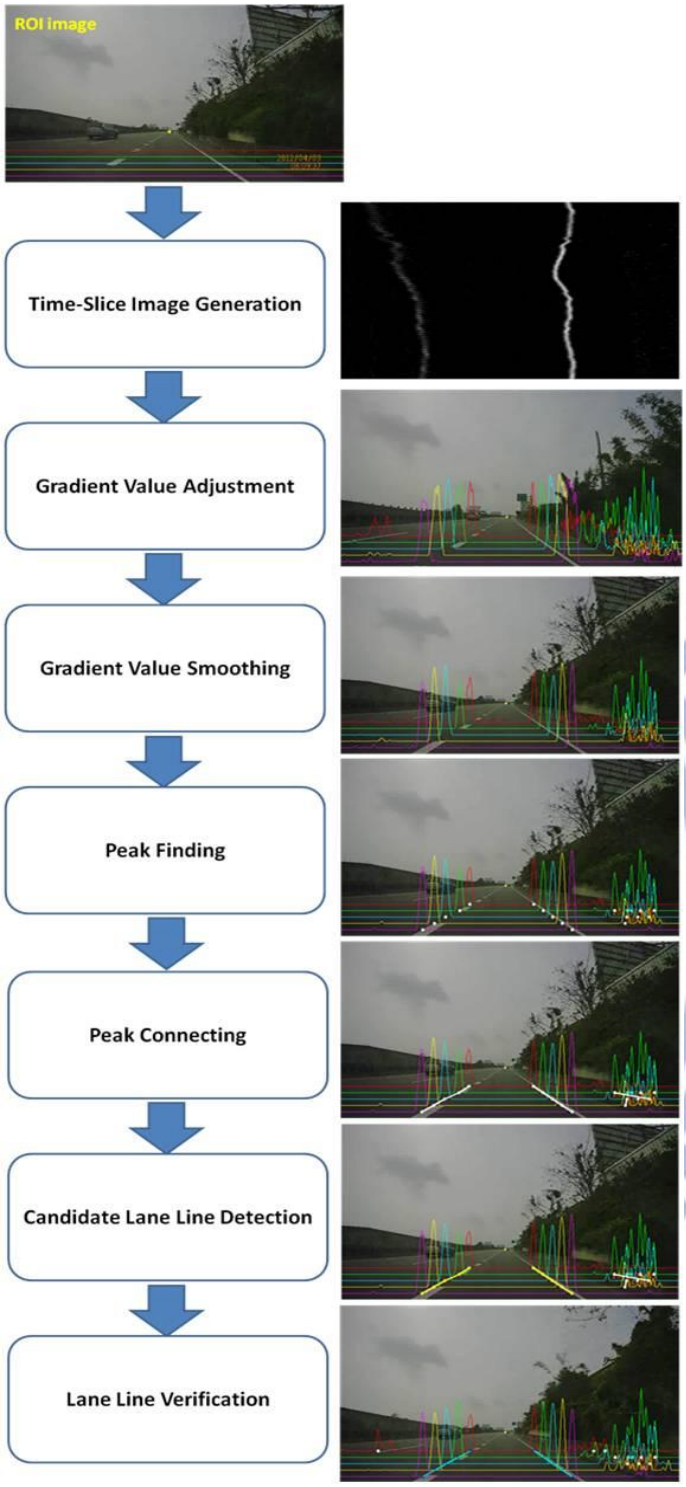


Figure 3-13 : Flowchart of *Lane Line Detection and Verification* module.

3.4.1 Time-Slice Image Generation

Inspired by [14], time-slice image generation greatly assists in the detection and tracking of lane lines, especially when the vehicle runs from the straight lane lines to

curve lane lines or the condition of changing lane happens, which are still challenging tasks for state-of-the-art works. Supposing that a video sequence totally have F frames, and each frame f_i (i = 1, …, F) is a $W \times H$ image. Then we record the specific row R_{row} of pixels from each frame in time, and thus we generate a $W \times F$ time-slice image, as shown in Figure 3-14. Besides, the index of each row of time-slice image is f, which means the frame number.

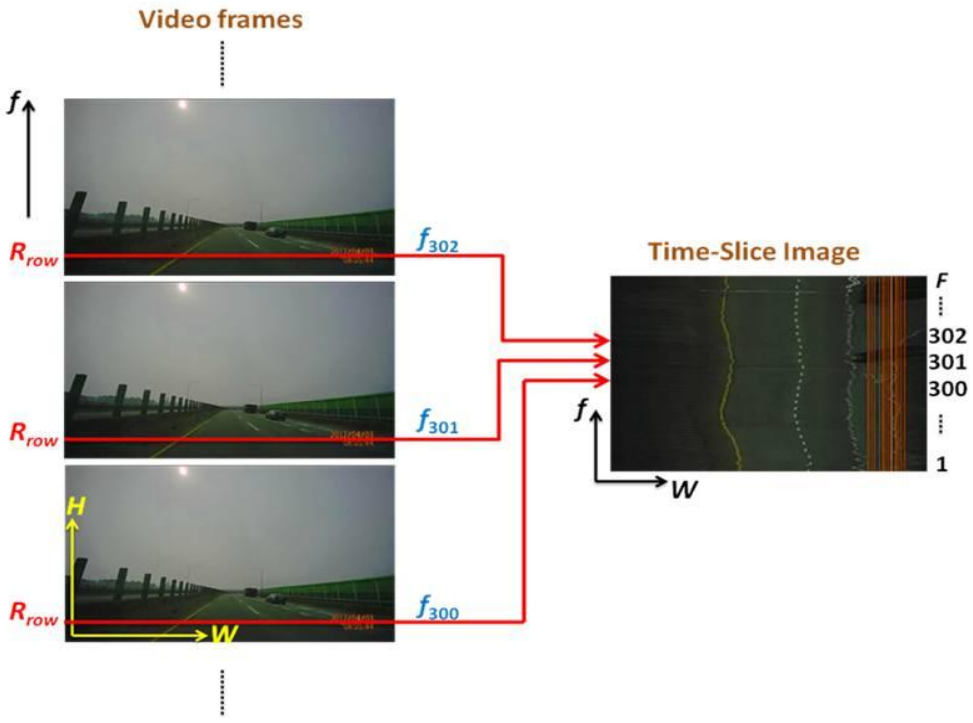


Figure 3-14 : Illustration of time slice generation.

3.4.2 Gradient Value Adjustment

Due to the discontinuousness of the dashed lane lines, detecting dashed lane lines in a single image becomes difficult and Hough transformation based lane line detection technique cannot work well. In order to overcome this obstacle, Borkar *et al.* [14] propose the temporal blurring algorithm, which adds the time conception to

generate an average image, giving the dashed lane lines the appearance of a near continuous line by connecting them. The concept of temporal blurring is to take only a few frames from the past in the averaging. The average image is defined as follows:

$$\text{AverageImage} = \sum_{i=0}^{N_{\text{past}}} \frac{I(n-i)}{N_{\text{past}}} \tag{13}$$

where I is the intensity of current frame, n is the index of current frame, and N_{past} is the number of frame from the past. Hence, the detection of these dashed lane lines becomes easier since they appear as a connected lines in the image. An example of temporal blurring is shown in Figure 3-15.

Figure 3-15 : An example of temporal blurring. (a) Original image. (b) Average image.

The temporal blurring algorithm is a good method to facilitate the detection of dashed lane lines. It is an important issue to determine how many frames should be used for in the averaging process. If too many frames are used, the perceived width of the lane lines will be altered; otherwise, if we use only a few frames, the effect of temporal blurring will become unobvious. In brief, this method is dependent on the moving speed of the vehicle. In our work, a method capable of detecting the dashed lane lines without considering the vehicle speed is proposed.

Five rows are selected as our ROIs (N_{row} = 5), and the gradient value of each ROI

is recorded. Figure 3-16 shows the gradient histogram of each ROI in the image with different colors. However, due to the discontinuousness of dashed lane lines, not every ROIs can get the gradient information of lane lines in current frame. As shown in Figure 3-16, the gradient value of the left dashed lane line is only recorded on the first and third ROIs. For resolving this obstacle, we propose the gradient value adjustment algorithm to retain the lane line information. The detail of this algorithm is described in Algorithm 1. For each ROI, we first set an accumulative gradient histogram to record the change of value. Then, for each pixel, we compare its gradient value in current frame with the value of corresponding pixel in accumulative histogram. If the current value is less than the accumulative value, we will decrease the accumulative value by one. This method can avoid decreasing the value too fast to retain the position of the dashed lane line. Otherwise, if the current value is larger than the accumulative value, the current value will substitute for the accumulative value. Afterward, repeating the above steps until all ROIs are examined. We also need N_{past} frames from the past to obtain the completed information of dashed lane lines. However, the advantage of our algorithm is that we do not have to give N_{past} in advance, N_{past} is automatically determined in the process. (N_{past} = 3 in our experiment averagely.) The result of gradient value adjustment is illustrated in Figure 3-17.

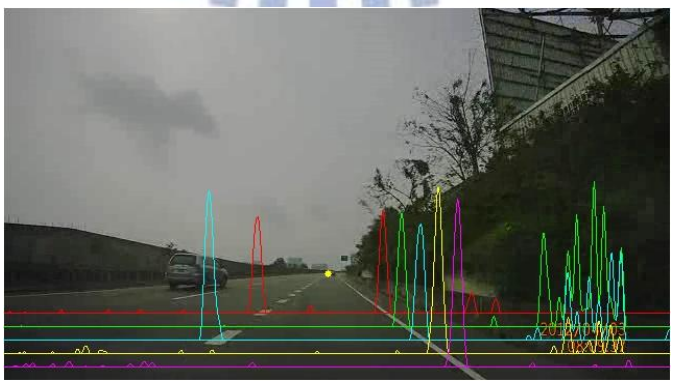


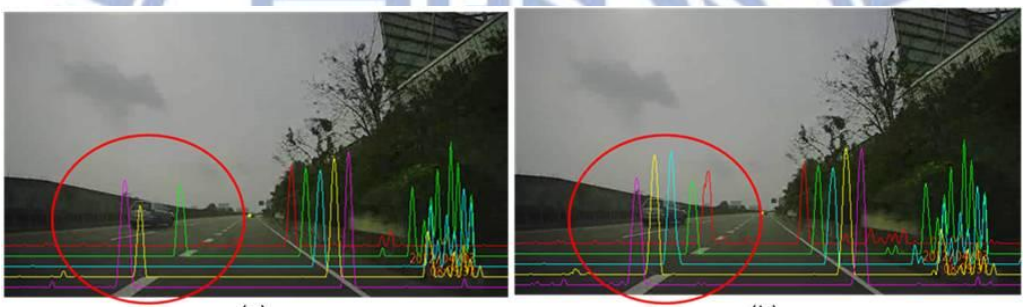
Figure 3-16 : The gradient histogram of each ROI shown with different colors.

Algorithm 1: Gradient Value Adjustment

Input: The accumulative gradient value of each ROI (h) and
the gradient value of each ROI in current frame (g)

Output: The gradient value of each ROI after adjusting

```
1   for each ROI do
2       for j = 0 to width //each pixel on the ROI
3           if (g_j < h_j) then h_j = h_j -1;
4               if (h_j < 0) then h_j = 0;
5               endif
6           endif
7           else h_j = g_j;
8       end for
9   end for
```



(a) (b)

Figure 3-17 : The result of gradient value adjustment. (a) Original gradient histogram of each ROI which sometime does not include the information of dashed lane line. (b) The gradient value adjustment algorithm compensates the detection of dashed lane line. The red cycle shows the final result of dashed lane line.

3.4.3 Gradient Value Smoothing

As shown in Figure 3-17, there are several “hills” in the gradient histogram of each ROI. A formal definition of the “hill” [41] can be given as: A range over which the values increase first and decrease next without any internal ripples in the histogram. As illustrated in Figure 3-18, the peak point (P_p) is the point which has maximum gradient value in a hill. Under the ideal case, the position of a lane line

corresponds to the position of a peak point for each ROI. However, the lateral moving of the vehicle results in many hills in the gradient histogram on each ROI, as shown in Figure 3-19(a), this phenomenon causes too many peak points.

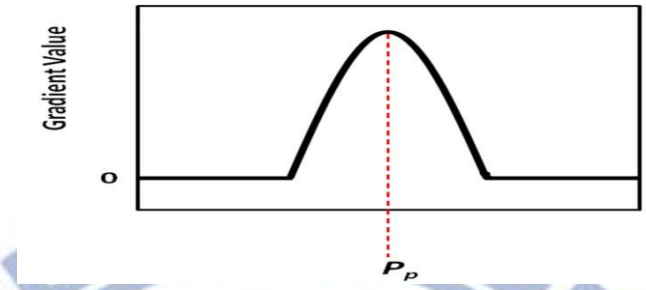


Figure 3-18 : The hill and peak point (P_p) in the histogram [41].

As described in Section 3.2.2, for each ROI, in order to remove the noise on the gradient histogram, we apply 1-D Gaussian smoothing filter in the *x*-direction. Table 1 shows the 1-D *x* component kernel that we apply [37]. After smoothing, the internal ripples in the histogram are removed and the maximal peak is retained, as shown in Figure 3-19(b).

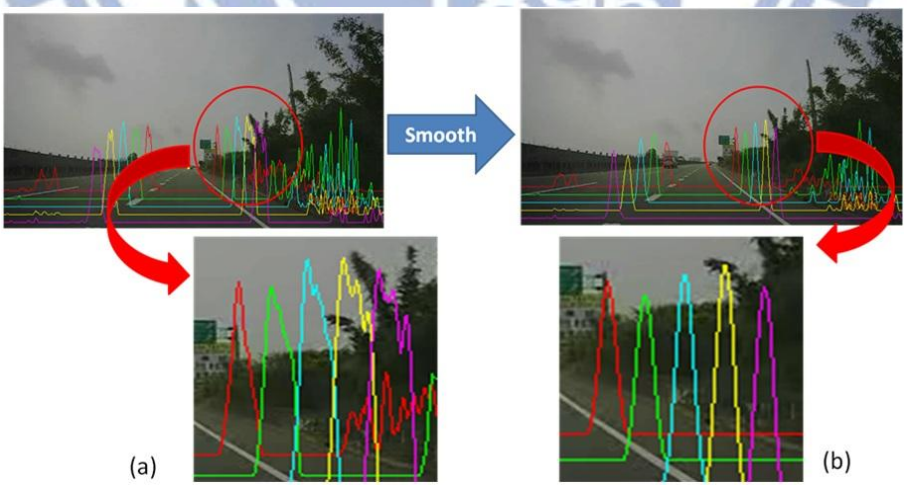


Figure 3-19 : The result of gradient value smoothing. (a) Original gradient histogram with many internal ripples. (b) The gradient histogram only with the maximal peak after gradient value smoothing.

Table 1 : The 1-D convolution kernel of Gaussian Filter [37].

.006	.061	.242	.383	.242	.061	.006

3.4.4 Peak Finding

Since a lane line typically owns the high contrast (or high edge response) to the neighboring road surface for attracting drivers' attentions. Here, we apply the peak finding algorithm [41] to extract the feature points of lane lines in the image. Once we obtain a peak point, we apply further restrictions to determine whether it corresponds to a lane line. That is, the value of a peak point must be larger than an adaptive threshold Th_{val} which is defined as:

$$Th_{val} = gv_{max} / 2 \tag{14}$$

where gv_{max} is the maximum gradient value of a ROI. Besides, the distance between this peak point and the neighboring peak points should be larger than a distance threshold (Th_d). We set the value of Th_d to 20 in the experiment. If two neighboring peaks are within a distance of Th_d, the one with small value is discarded. After applying the peak finding algorithm, we obtain the feature points of lane lines, as shown in Figure 3-20.

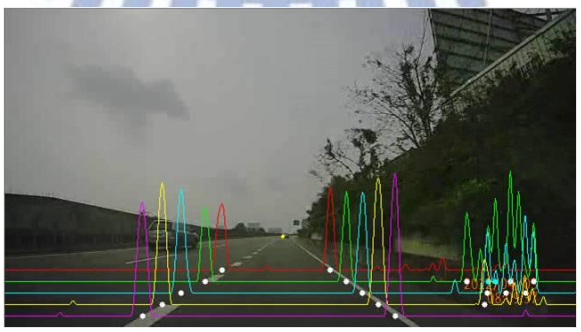


Figure 3-20 : The result of peak finding algorithm. Found (picked) peak points, represented by the white points.

3.4.5 Peak Connecting

After obtaining the peak points, we produce lane line candidates by connecting the similar peak points. First, a line segment L_i is generated by a peak point and the vanishing point and another line segment L_j is a horizontal line. We calculate the angle between L_i and L_j for each peak point. For example, in Figure 3-21(a), those blue points represent the peak points obtained in peak finding step, and the angle of each point is also shown. As we know, for a straight lane line, the angles among the points are almost the same or similar. Based on this property, for each pair of ROI (ROI_i and ROI_{i+1}), we select two peak points with the smallest angle in ROI_i and ROI_{i+1}, respectively. That is, P_1 and P_3. The angle difference of two peak points is within a threshold Ang_{thres} (In our example, Ang_{thres} = 5), the pair of two peak points is recorded and then the peak point with small angle is discarded. We repeat the above steps until there is no peak point to match in ROI_i and ROI_{i+1}. Now we go back to our example, since the angle of P_1 and P_3 is similar ($|30 - 29| = 1 < Ang_{thres}$), we record this pair and then abandon P3 which has the small angle. Next round, P1 and P4 are selected. However, the angle difference of these two points is too large ($|30 - 90| = 60 > Ang_{thres}$), we do nothing and abandon P1 which has the smaller angle. In round 3, P2 and P4 are selected. As the same as round 2, we abandon P4 ($|90 - 121| = 31 > Ang_{thres}$). In round 4, P2 and P5 are selected. We find that their angles are similar ($|121 - 120| = 1 < Ang_{thres}$), thus this pair is recorded. Afterward, there is no peak point to be matched, and then we stop. The final result of this example is shown in Figure 3-21(b). After scanning all ROIs and doing peak connecting algorithm, the result of real road image is shown in Figure 3-22.

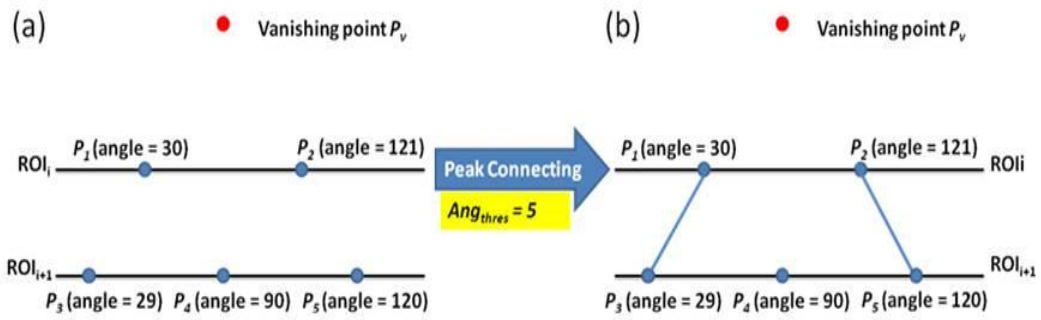


Figure 3-21 : An example of peak connecting algorithm. (a) 5 peak points (blue) and the vanishing point (red). (b) Peak connecting result of (a). Each blue line segment means that two peak points are similar.

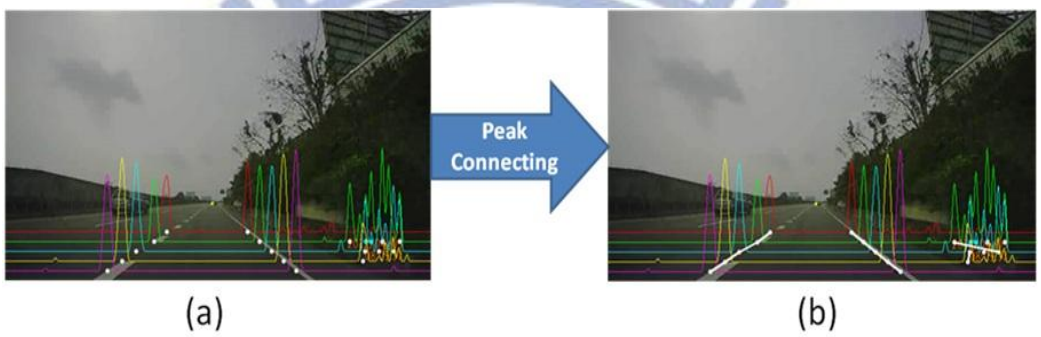


Figure 3-22 : The result of peak connecting on the real road image. (a) Peak point image. (The peak points are represented by the white points.) (b) Peak connecting image. (The white line segments imply the relationship within the similar peak points.)

3.4.6 Candidate Lane Line Detection

After peak connecting step, we acquire several line segments in the image. Nevertheless, under our assumption, a lane line must be satisfied that all ROIs can find a corresponding point. We group these line segments into a line by observing their start point and end point. Next, we choose those lines which can pass through all ROIs as our candidate lane lines and put them into our candidate list. Figure 3-23 shows the detected candidate lane lines which are painted with yellow color.

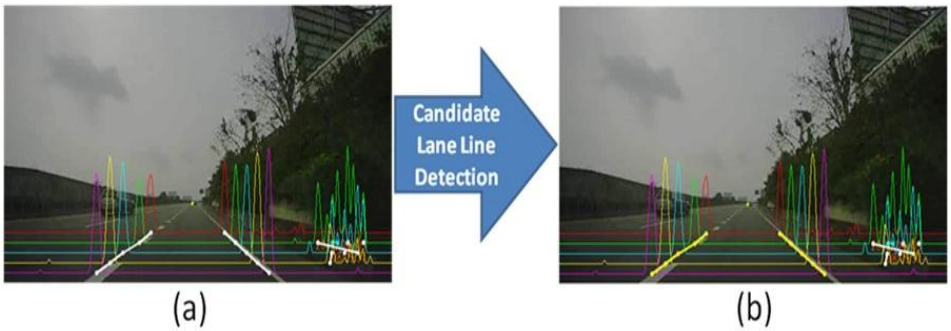


Figure 3-23 : The result of candidate lane line detection. (a) Peak connecting image. (b) Two candidate lane lines are obtained in current frame (yellow lines).

3.4.7 Lane Line Verification

We verify all candidate lane lines in our candidate list through tracking them in *M* frames. Once a lane line is detected in current frame, it should be found in the subsequent frames; otherwise, this lane line may be a false one. Suppose a candidate lane line is detected for *M* frames, it is a valid lane line and then we can put it into our tracking list. Figure 3-24 shows the result of lane line verification on the real road image, we paint the valid lane lines with cyan color. In car video sequence, we may face with the different driving environments, such as various markings on the road, as shown in Figure 3-25(a). In this case, some false candidate lane lines are generated, as shown in Figure 3-25(b) (the yellow lines). Nevertheless, Figure 3-25(c) shows that our lane line verification algorithm can eliminate the false candidate lane lines effectively.

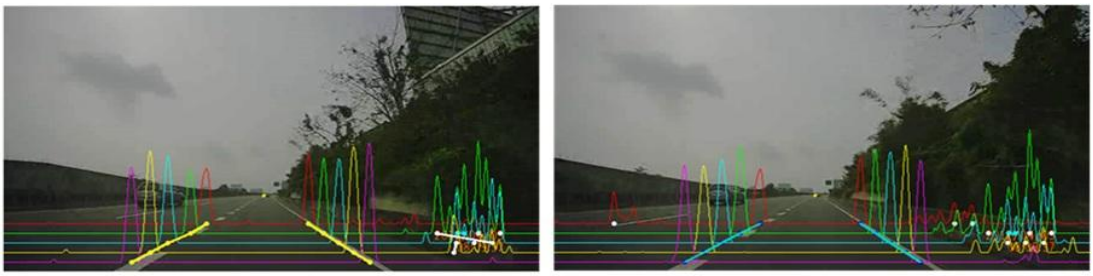


Figure 3-24 : The result of lane line verification. (a) Two candidate lane lines are drawn with yellow color. (b) Final results of lane lines are drawn with cyan color.

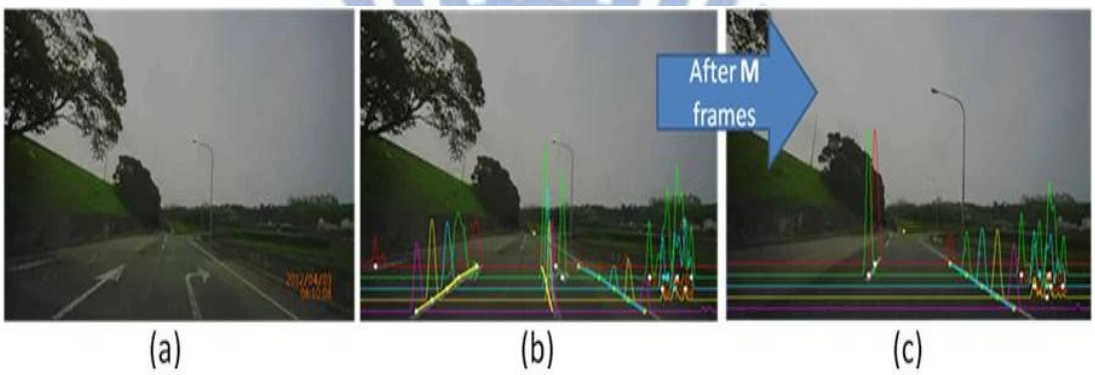


Figure 3-25 : Elimination of the false candidate lane lines in lane line verification. (a) Original image. (b) Two false candidate lane lines are generated since the presence of arrow markings on the road. (c) Within *M* frames, these two false candidate lane lines are not detected consecutively and thus be eliminated.

3.5 Lane Line Tracking

As illustrated in Figure 3-26, when a new frame comes, if there are several lane lines in the tracking list and candidate list, we first track the lane lines in the tracking list, and then track the candidate lane lines in the candidate list. Once a candidate lane line passes the limitation in lane line verification, as described in Section 3.4.7, this candidate lane line can be put into the tracking list. We describe the tracking algorithm about lane line tracking and candidate lane line tracking in the following.

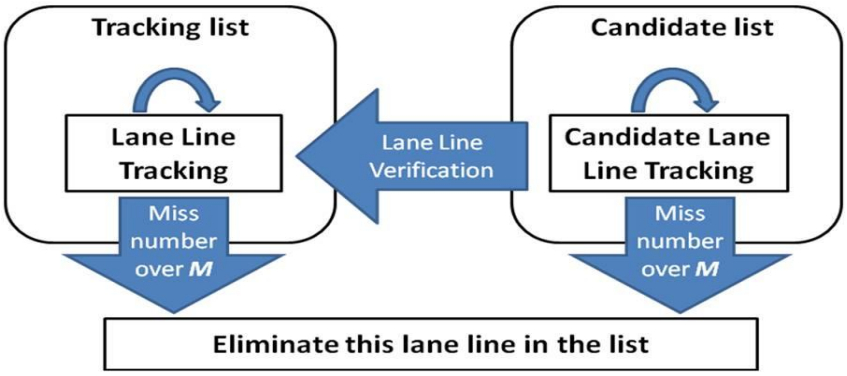


Figure 3-26 : The relationship between the candidate list and tracking list.

Since the position of a lane line (candidate) seldom changes between two consecutive frames, we can track a lane line (candidate) by calculating the moving velocity (*vel*) and moving acceleration (*acc*) and then predicting the possible position in a new frame. Nevertheless, we need to obtain the position of a lane line (candidate) in the previous frames for calculating *vel* and *acc*. Therefore, we do this prediction procedure when 2 frames have been processed; otherwise, we directly track the lane line (candidate) by finding the surrounding area of the current position in a new frame. Here, we take the time-slice image generated by ROI_5, as shown in Figure 3-27(b), to give an example of the prediction procedure, as shown in Figure 3-27(c). In this example, the *x*-coordinate of the current point is x_3. To calculate *vel* and *acc*, we apply the formulas defined from Eq. (15) to Eq. (17) where t_i (i = 1, 2, 3, 4) is the time index. Afterward, we obtain the predicted *x*-position (x_4) of the point in a new frame from two perspectives: one is obtained from Eq. (18) without the consideration of *acc*, another is obtained from Eq. (19). As described in Eq. (20), we combine two possible positions above by averaging them and obtain the final predicted position x_4. For a new frame, we apply the peak finding algorithm, as mentioned in Section 3.4.4, by this predicted position to track the corresponding position of a lane line (candidate) on a ROI. When we can find the corresponding points of a lane line (candidate) on all ROIs, this lane line (candidate) is seen as “detected” in the new frame; otherwise, we

mark it as “miss”. If a lane line (candidate) consecutively misses for *M* frames, that is, the lane line (candidate) is mis-traced for too many frames, it is discarded from the tracking list or the candidate list and no longer tracked, as illustrated in Figure 3-26.

By the way, once the position of a lane line (candidate) in the tracking list (candidate list) is obtained in a new frame, we set the gradient value around this position to 0 on the gradient histogram of each ROI. This proceduce can avoid obtaining the same lane lines between the tracking list and candidate list.

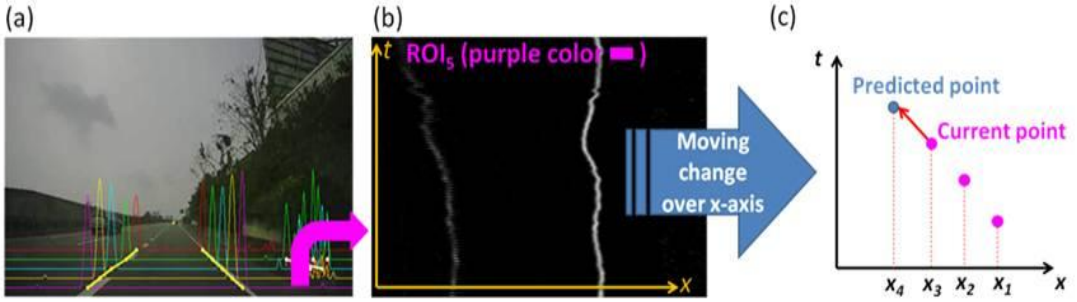


Figure 3-27 : The conception of prediction procedure. (a) Two candidate lane lines are drawn with yellow color. (b) The time-slice image generated by ROI_5 is used to track the lane lines. (c) An example of the moving change over *x*-axis. Here, x_3 is the current point, and x_4 is the predicted point.

$$vel_1 = \frac{x_2 - x_1}{t_2 - t_1} = x_2 - x_1 \tag{15}$$

$$vel_2 = \frac{x_3 - x_2}{t_3 - t_2} = x_3 - x_2 \tag{16}$$

$$acc = \frac{vel_2 - vel_1}{t_2 - t_1} = vel_2 - vel_1 \tag{17}$$

$$x_{4\_no\_acc} = x_3 + vel_2(t_4 - t_3) = x_3 + vel_2 \tag{18}$$

$$x_{4\_with\_acc} = x_3 + vel_2(t_4 - t_3) + \frac{1}{2}acc(t_4 - t_3)^2 = x_3 + vel_2 + \frac{1}{2}acc \tag{19}$$

$$x_4 = \frac{(x_{4\_no\_acc} + x_{4\_with\_acc})}{2} \tag{20}$$

Chapter 4. Experimental Results and Discussions

In this chapter, we present the experiments for the lane line detection and discuss the results. In the Section 4.1, we introduce the environments and datasets used in our experiments. Section 4.2 introduces the evaluation methods. The experimental results of the lane line detection and the problems occurring in the experiments are discussed in Section 4.3.

4.1 Experimental Environments and Datasets

All algorithms are implemented in C programming language and OpenCV (Open Source Computer Vision) Library, and all experiments are performed on a general PC with Genuine Intel® U7300 1.30GHZ CPU and Microsoft® Windows 7 professional operation system. We mount the camera on the upper center of windshield of the vehicle to capture the road images. Here, we have several video clips for test, but only three ground-truth video clips for evaluation. All these clips are captured on the highways with the solid/dashed lane lines on straight/curved roads. Moreover, in ground-truth video clips, Clip#1 contains the intermediate case of driving from straight lane lines to curve lane lines. Here, we only analyze the performance of the curve lane lines. Clip#2 is the lane changing case and Clip#3 is the straight lane lines case. The resolution of each video frame is 640×360 and the clip lengths (in frames) are listed in Table 2. Some sample images of the car videos are shown in Figure 4-1. Four modules are included in our system, i.e., *Pre-processing, Vanishing Point Computation and Row of Interest (ROI) Setting*, *Lane Line Detection and Verification*, and *Lane Line Tracking*. The processing time of each module is listed in Table 3. It

can be seen that the *Vanishing Point Computation and Row of Interest (ROI) Setting* module takes the most execution time in our system. The cause is from the OTSU Binarization step which takes 30ms to find an optimal threshold for each frame. In addition, there is no complicated process involved in our system, thus the processing speed our system can achieve is up to 21 *fps (frame per second)*.

Figure 4-1 : Sample road images in different cases. (a) Intermediate case where the type of lane lines is from the straight to the curve then back to straight. (b) Lane changing case from left to right. (c) Straight lane lines case.

Table 2 : Total number of frames of each ground-truth video clip.

Video Clip	Number of frame	Total frames
Clip#1 (intermediate case)	130	8452
Clip#2 (lane changing case)	1236	
Clip#3 (straight lane lines case)	7086	

Table 3 : Processing time of each module in our system.

Modules	Processing time
Pre-processing	6.59 ms
Vanishing Point Computation and Row of Interest (ROI) Setting	38.8 ms
Lane Line Detection and Verification	1.28 ms
Lane Line Tracking	0.09 ms

4.2 Evaluation Method

To evaluate the performance of our system, we refer to two performance indices in [35]: the *missing rate (MR)* and *false detection rate (FR)*, as defined by Eq. (21).

$$MR = \frac{N_{\text{real}} - N_C}{N_{\text{real}}} \quad \text{and} \quad FR = \frac{N_D - N_C}{N_D} \tag{21}$$

where N_{real} is the number of ground-truth lane lines existing in the road scenes to be analyzed, N_C is the number of correctly detected lane lines, and N_D is the number of detected lane lines.

As the same as the measurement method in [35], we first draw the ground truth of a lane line manually to decide whether it is correctly detected. For each pixel in the detected lane line, its corresponding point in the ground-truth lane line is the one with the same *y*-coordinate. Then, we calculate their *x*-coordinate difference as their distance. After scanning all pixels in the detected lane line, their average distance can be obtained. If the average distance is less than half of a lane line width, the lane line is labeled to “correct”.

4.3 Experimental Results

Here, we implement several related methods on the intermediate case, the lane changing case and the straight lane lines case, and then we show the experimental results. We discuss the problems occurring in the experiment. Since the intensity of a lane line is brighter than the neighboring road surface in the image, Nadra *et al.* [36] utilize the top-hat transformation, which is one of the morphological operations, in the pre-processing step to extract the clear regions in the image regardless of background variations, as shown in Figure 4-2(b). For the contrast enhancement, as shown in Figure 4-2(c), they give a threshold based on the top-hat transformed image to enhance the intensity of the pixels. However, the threshold is hard to define. Another question is how to select the structure element of top-hat transformation for lane line detection. If we apply the 3×3 structure element and set the threshold as 6 in the top-hat transformation to test our dataset, some lane lines may be destroyed, as shown in Figure 4-3(b). Besides, due to too many noise pixels included in the enhancement image, as shown in Figure 4-2(c), the recognition rate of lane lines decreases. By the way, before pre-processing, they divide the image into left and right parts, as shown in Figure 4-2(a), as the regions of interest. This procedure makes the system only can detect the condition where the lane lines are located in the left and right parts. When doing the lane changing, the lane line crossing the middle part is not detected.

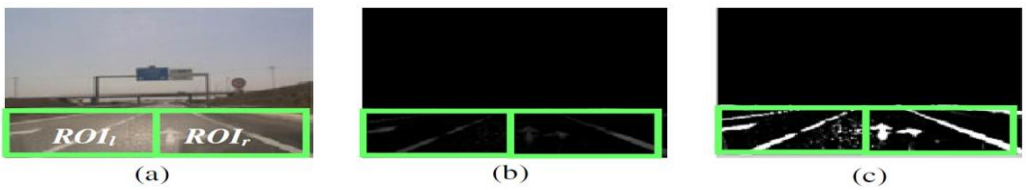


Figure 4-2 : The results of pre-processing step in [36]. (a) Original image. (b) Image after top-hat transformation. (c) Image after contrast enhancement.

(a) (b)

Figure 4-3 : The result after the top-hat transformation and contrast enhancement. (a) Original image. (b) Image after top-hat transformation and contrast enhancement where the red cycle shows the destroyed part of the lane line.

Hence, for reducing the noise pixels in the textured areas, we apply an additional constraint "structure tensor" to find those line-structure pixels on the contrast-enhanced image. By observing the two eigenvalues of the structure matrix *SM* which is computed over a small window of size (*2q*+1) around each candidate pixel (*x*, *y*) and defined by Eq. (22) [37].

$$SM = \sum_{i=x-q}^{x+q} \sum_{j=y-q}^{y+q} \nabla g(i,j) \cdot (\nabla g(i,j))^T \tag{22}$$

Depending on the two eigenvalues of the matrix *SM*, called λ_1 and λ_2 ($\lambda_1 \geqq \lambda_2$), the area can be classified into *textured* (both λ_1 and λ_2 are large), *linear* ($\lambda_1 >> \lambda_2$), and *flat* (both λ_1 and λ_2 are small). On the straight lane lines, the *linear* case will apply to retain the pixels only if $\lambda_1 > \beta\ \lambda_2$ where β is a constant. Figure 4-4 shows that the effectiveness of this method in removing the word pixels in the bottom right corner of the image and the pixels of the grass or the wall. Nevertheless, the disadvantage of this method is that it takes more than 300ms, which is not suitable for the real-time requirement. For the reason of above paragraphs, we do not apply the top-hat transformation and line-structure constraint in our system.

As described in Section 3.3.2, the Hough transformation is one of the most common algorithms to detect the straight lines in an image. For various types of lane lines, there should be many lane line models used to describe them. The straight and

curve are the common lane line models. In this thesis, we first detect and record the lane lines from the straight, and then track them in subsequent frames. Figure 4-5 shows the result of lane line detection when a new lane line appears in the middle of two originally detected lane lines. Result of the intermediate case of driving from the straight lane lines to curve lane lines then back to straight lane lines is shown in Figure 4-6. Figure 4-7 and Figure 4-8 show the result of the lane changing case and the straight lane line case, respectively.

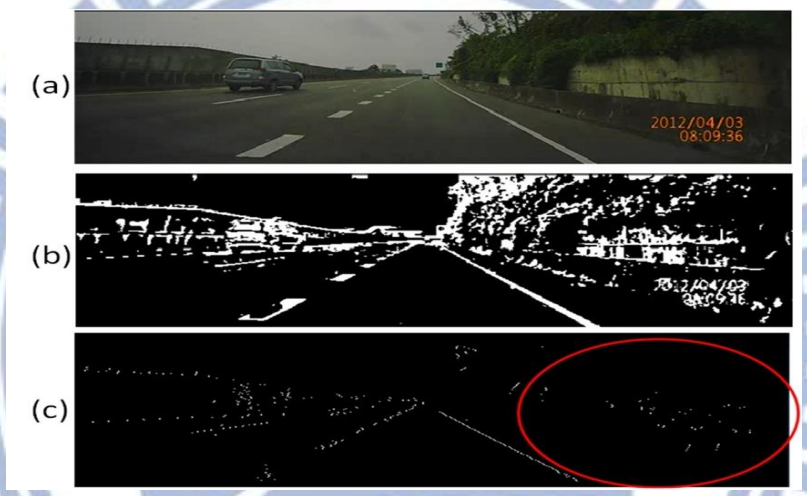


Figure 4-4 : The result of line-structure constraint. (a) Original image. (b) Image after top-hat transformation and contrast enhancement. (c) Image after the line-structure constraint on (b). The red cycle shows the effectiveness of noise removal.

Figure 4-5 : A new lane line appears in the middle of two originally detected lane lines. (a) Original images. (b) Output images.

Figure 4-6 : The intermediate case of driving from straight lane lines to curve lane lines then back to straight lane lines. (a) Original images. (b) Output images.

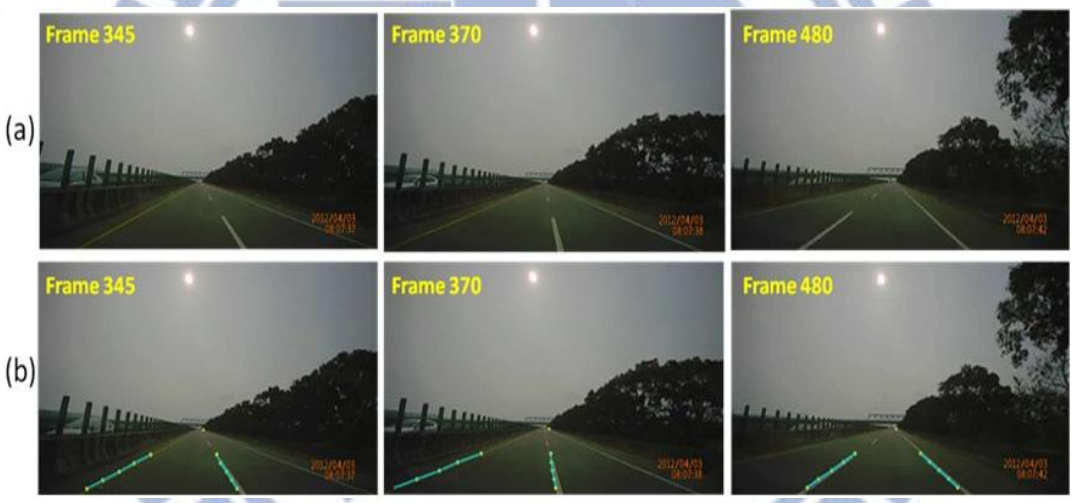


Figure 4-7 : The lane changing case. (a) Original images. (b) Output images.

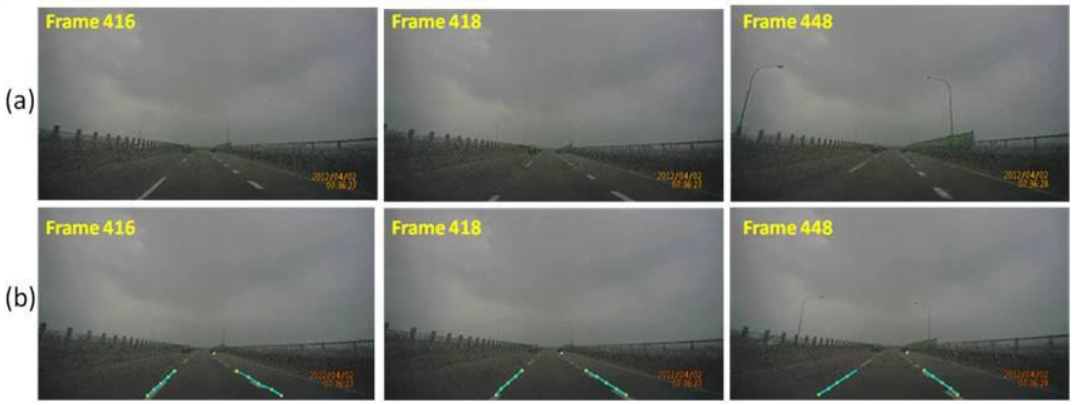


Figure 4-8 : The straight lane line case. (a) Original images. (b) Output images.

Although our algorithm applying the time-slice images to track the lane line points is feasible, there are two problems arising in our experiments. Problem 1 is that the number of times the gradient value smoothing step is performed affects the declining speed of the gradient histogram. As described in Section 3.4.3, we smooth the movement of gradient histogram and reduce the generation of the wrong peak points when the lateral moving of the vehicle occurs. Nevertheless, the more number of times of smoothing, the more quickly the gradient histogram decreases. This phenomenon makes the detection of the dashed lane lines go wrong easily. Figure 4-9 shows an example of incorrect lane line detection due to too many times of smoothing steps. We focus on the left lane line and use four times of smoothing step. Since its internal distance is farther than the right lane line, we loss its information on the first ROI (red color) at frame 458, which is just 3 frames after frame 455. Since from frame 458 to frame 464, the next left lane line segment does not pass our first ROI yet, we loss the left lane line information on the second, third, fourth, and fifth ROI in the image, respectively. Although the new left lane line segment passes our ROIs at frame 465, the left lane line in the tracking list has been destroyed since it misses for too many frames. On the other hand, as shown in Figure 4-10(a) and (b), we apply 1 time smoothing and 4 times smoothing in the gradient histogram, respectively. The gradient histogram after 1 times smoothing generates many peaks in the ROIs, which makes the result of lane line detection go wrong easily. The yellow regions show the difference between Figure 4-10(a) and (b).

Figure 4-9 : Problem 1 caused by the smoothing step. (a) Original images. (b) Output image where the left lane line is incorrect at the frame 465.

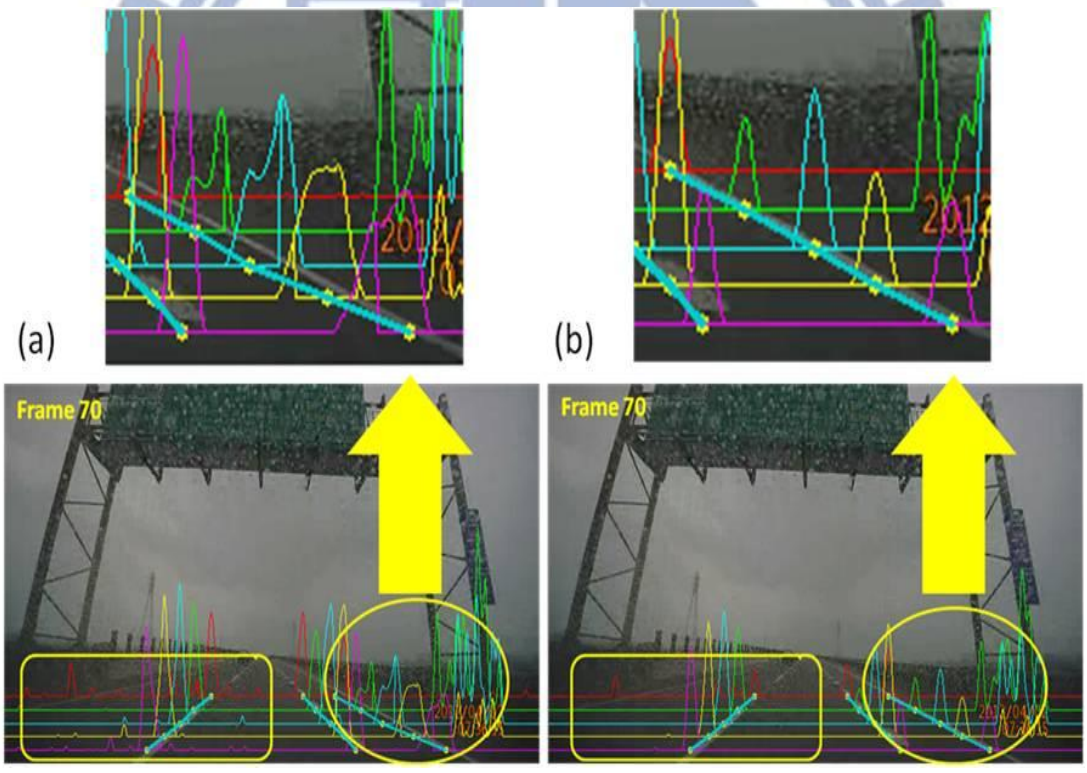


Figure 4-10 : The results under the different times of the smoothing step. (a) After 1 time of smoothing step, our system generates an incorrect lane line because of too many peaks in the gradient histogram. (b) After 4 times of smoothing step, the result of lane line detection is almost correct because of the proper gradient histogram.

Problem 2 is how to check whether the shape of a lane line is correct when tracking both on the straight lane lines and curve lane lines. Even though we already

obtain a lane line positions on each ROI in the detection step, the new point positions on each ROI tracked correctly cannot be ensured because of the noise. The noises may come from the leading vehicles (*c.f.* Figure 4-11 (a)), the words (*c.f.* Figure 4-11 (b)), and the reflection of the windshield (*c.f.* Figure 4-11 (c)). They cause errors in lane line detection. Furthermore, if we find out that several tracked points of a lane line are wrong, how to modify this lane line to be correct is also one of the challenges in our system.

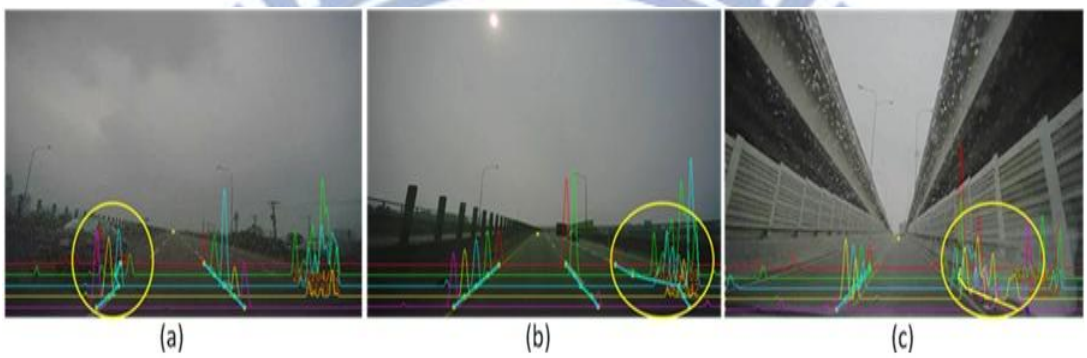


Figure 4-11 : Problem 2 caused by the noises. (a) Noise from the leading vehicle. (b) Noise from the words. (c) Noise from the reflection of the windshield.

In the end, we also implement the method proposed by Nadra *et al.* [36] in our experiments. To compare with this method by the statistic analyses, three performance indices are used, i.e., *missing rate*, *false detection rate*, and *accuracy*. The definitions of *missing rate (MR)* and *false detection rate (FR)* are described by Eq. (21) in Section 4.2. Table 4 lists the detailed performance comparisons between [36] and our method. From the view of *accuracy*, the method [36] performs better than our method for the Clip#2 (lane changing case) and Clip#3 (straight lane line case) since we have not resolved the Problem 1 and Problem 2 yet. But for the Clip#1 (intermediate case), their method performs worse than our method because their method needs to delimit a region for the curve fitting method. However, the performance of their system does

not work well on our dataset because of two reasons. One reason is that they select the maximum voting line by the Hough transformation as the lane line in the left and right region of interest, respectively. But they do not mention about how to select the better region of interest in their system for different car images. The other reason is that they only detect the lane lines for each single frame, and do not include the tracking conception. Looking back at the performance of our system, since the challenges mentioned before still need to be conquered, there is still room for improvement. In conclusion, if we can overcome the problems arising in our experimental results, our system utilizing the time-slice images to detect and track the lane lines will become feasible and robust.

Table 4 : Performance comparisons between Nadra *et al.*[36] and our method.

Clips	**# of frames**	**Methods**	**Missing frames**	**False frames**	***MR* (%)**	***FR* (%)**	**Accuracy (%)**
Clip#1	130	Nadra *et al.*[36]	10	70	7.69	53.85	38.46
		Our method	54	0	41.54	0	58.46
Clip#2	1236	Nadra *et al.*[36]	48	884	3.88	71.52	24.6
		Our method	56	939	4.53	75.97	19.5
Clip#3	7086	Nadra *et al.*[36]	146	3003	2.06	42.38	55.56
		Our method	3119	494	44.02	6.97	49.01

Chapter 5. Conclusions and Future Works

In this thesis, we propose a lane line detection and tracking system utilizing the time-slice images. The system architecture consists of four modules, including (1) *Pre-processing*, (2) *Vanishing Point Computation and Row of Interest (ROI) Setting*, (3) *Lane Line Detection and Verification*, and (4) *Lane Line Tracking*. *Pre-processing* is performed by RGB to grayscale, image smoothing, image normalization, and edge detection for obtaining the edge feature of lane lines. *Vanishing Point Computation and Row of Interest (ROI) Setting* consists of Otsu binarization, Hough transformation, vanishing point computation, and ROI setting for locating the vanishing point and delimiting our ROIs. Time-slice image generation, gradient value adjustment, gradient value smoothing, peak finding, peak connecting, candidate lane line detection, and lane line verification are utilized for extracting the lane lines in the image in *Lane Line Detection and Verification*. The gradient value adjustment algorithm is proposed to overcome the sparseness problem in detecting the dashed lane lines. At last, *Lane Line Tracking* applies the prediction procedure to track a lane line in the time slice images. Since we consider the location information of a lane line from previous images to constrain the probable lane detection in the current image and only process on the ROIs instead of the whole image, the processing time of an image is reduced. The testing images of the car video clips are captured from the camera mounted on the upper center of windshield of the vehicle and we focus on the intermediate case of driving from the straight lane lines to curve lane lines and the lane changing case. The experimental results show that our proposed methods can improve the recognition of the lane line. However, there are still two problems needed to be resolved, as

described in Section 4.3.

Thus, some interesting issues based on time-slice images for the lane lines extraction are worthy of further investigation. For the future works, we have some suggestions:

(1) While the vanishing point is determined, how many rows in the image should be selected as our ROIs? More ROIs can facilitate the lane line detection and describe the shape of the lane lines in detail, but the processing time increases accordingly.

(2) After selecting the new lane line points on each ROI, how to set constrains to check whether the shape of the lane line is correct? Though utilizing the time-slice images to track the lane line points is an intuitive approach, the positions of the lane lines easily suffer from the noises such as vehicles and shadows in the image. Hence, more features should be taken into consideration for eliminating noises in the image and to extract the lane lines more efficiently.

(3) To what extent of smoothing in the gradient histogram is suitable for the system to detect the feature points of the lane lines?

(4) If a system not only detects the lane line positions, but also recognizes the types of lane lines such as solid or dashed, single or double, yellow or white, the drivers can have better understanding about the driving environment. The drivers also can protect themselves in advance of a possible accident on the road.

(5) In this thesis, we only discuss the intermediate case of driving from the straight lane lines to curve lane lines or the lane changing case. Other cases under the different weather conditions or different scenarios should be taken into consideration for constructing a more robust lane line detection system.

Bibliography

[1] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, “Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations,” Vehicle System Dynamics, vol. 44, no. 7, pp. 569–590, July 2006.

[2] S. Ezell, “Explaining International IT Application Leaderhip : Intelligent Transportation Systems”, The Information Technology & Innovation Foundation, January 2010. http://www.itif.org/files/2010-1-27-ITS_Leadership.pdf

[3] J. C. McCall and M. M. Trivedi, “Video-based lane detection estimation and tracking for driver assistance: Survey, system, and evaluation,” IEEE Transactions on Intelligent Transportation System, vol. 7, no. 1, pp. 20-37, March 2006.

[4] M. Aly, “Real time detection of lane markers in urban streets,” in Proc. IEEE Intelligent Vehicles Symposium, pp. 7-12, June 4–6, 2008.

[5] D. Khosla, “Accurate estimation of forward path geometry using two-clothoid road model,” IEEE Intelligent Vehicles Symposium, vol. 1, pp. 154-159, June 17-21, 2002.

[6] S. Nedevschi, R. Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga, and C. Pocol, “3D lane detection system based on stereovision,” in Proc. IEEE Intelligent Transportation Systems Conference, Washington, DC, pp. 161-166, Oct. 3-6, 2004.

[7] Y. Otsuka, S. Muramatsu, H. Takenaga, Y. Kobayashi, and T. Monj, “Multitype lane markers recognition using local edge direction,” in Proc. IEEE Intelligent Vehicles Symposium, pp. 604-609, Jun. 2002.

[8] C. Rasmussen, “Combining laser range, color, and texture cues for autonomous

road following," in Proc. IEEE Int. Conf. Robot. Autom., pp. 4320-4325, Aug. 2002.

[9] R. Tapia-Espinoza and M. Torres-Torriti, "A comparison of gradient versus color and texture analysis for lane detection and tracking," in Proc. Latin Amer. Robot. Symp., pp. 1-6, Oct. 2009.

[10] C. Kreucher and S. Lakshmanan, "LANA: A lane extraction algorithm that uses frequency domain features," IEEE Transactions on Robotics and Automation, vol. 15, no. 2, pp. 343-350, Apr. 1999.

[11] Q. Li, N. Zheng, and H. Cheng, "Springrobot: A prototype autonomous vehicle and its algorithms for lane detection," IEEE Trans. on Intelligent Transportation Systems, vol. 5, no. 4, pp. 300-308, Dec. 2004.

[12] D. Kang and M. Jung, "Road lane segmentation using dynamic programming for active safety vehicles," Pattern Recognit. Lett., vol. 24, no. 16, pp. 3177-3185, Dec. 2003.

[13] 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.

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

[15] N. Apostoloff and A. Zelinsky, "Robust vision based lane tracking using multiple cues and particle filtering," in Proc. IEEE Intell. Veh. Symp., pp. 558–563, Jun. 2003.

[16] Z. Kim, "Robust lane detection and tracking in challenging scenarios," IEEE Trans. Intell. Transp. Syst., vol. 9, no. 1, pp. 16-26, Mar. 2008.

[17] A. Broggi, "Robust real-time lane and road detection in critical shadow conditions," Proceedings of the IEEE International Symposium on Computer

Vision, Coral Gables, Florida, pp. 353-358, November 19-21, 1995.

[18] A. Broggi, and S. Berte, “Vision-based road detection in automotive systems: A real-time expectation-driven approach,” Journal of Artificial Intelligence Research, vol. 3, pp. 325-348, 1995.

[19] M. Bertozzi, and A. Broggi, “GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection,” IEEE Trans. on Image Processing, vol.7, no.1, pp.62-81, Jan. 1998.

[20] Y. He, H. Wang, and B. Zhang, “Color-based road detection in urban traffic scenes,” IEEE Transactions on Intelligent Transportation System, vol. 5, no. 4, pp. 309-318, Dec. 2004.

[21] H.Y. Cheng, B.S. Jeng, P.T. Tseng, and K.C. Fan, “Lane Detection with Moving Vehicles in the Traffic Scenes,” IEEE Transactions on Intelligent Transportation System, vol. 7, no. 4, pp. 571-582, Dec. 2006.

[22] Y. R. Huang, Y. L. Pan, "Fast Algorithm for Structured and Unstructured Road Detection", IEEE International Congress on Image and Signal Processing, pp. 1-5, 2009.

[23] L.W. Tsai, J.W. Hsieh, C.H. Chuang, and K.C. Fan, “Lane detection using directional random walks,” IEEE Intelligent Vehicles Symposium, pp.303-306, June 4-6, 2008.

[24] Y. U. Yim, and S. Y. Oh, “Three-feature based automatic lane detection algorithm (TFALDA) for autonomous driving,” IEEE Transactions on Intelligent Transportation System, vol. 4, pp. 219–225, Dec. 2003.

[25] K. Kluge, and S. Lakshmanan, “A deformable-template approach to lane detection,” in Proceedings of the IEEE Intelligent Vehicles Symposium, Detroit, pp. 54–59, September 25-26, 1995.

[26] Canny, J., “A computational approach to edge detection,” IEEE Trans. Pattern

Analysis and Machine Intelligence, vol.8, no.6, pp.679-698, Nov. 1986.

[27] Steven M. Kay, Fundamentals of Statistical Processing, Volume I: Estimation Theory, Prentice Hall Signal Processing Series, 1993.

[28] J. Wang, Y. Wu, Z. Liang, and Y. Xi, "Lane detection based on random hough transform on region of interesting," Proc. IEEE International Conference on Information and Automation (ICIA), pp. 1735-1740, 2010.

[29] Y. Wang, D. Shen, and E. K. Teoh, "Lane Detection Using Catmull-Rom Spline," in Proc. IEEE Intelligent Vehicles Symposium, pp. 51-57, Oct. 1998.

[30] J. W. Park, J. W. Lee, and K. Y. Jhang, "A Lane-Curve Detection Based on An LCF," Pattern Recognition Letters, vol. 24, no. 14, pp. 2301-2313, Oct. 2003.

[31] C. Xu, and J. L. Prince, "Snakes, shapes, and gradient vector flow," IEEE Transactions on Image Processing, vol. 7, no. 3, pp. 359-369, 1998.

[32] E. Catmull, and R. Rom, "A class of local interpolating splines," in Computer Aided Geometric Design, New York, pp. 317-326, 1974.

[33] N. Otsu, "A threshold selection method from gray level histograms," IEEE Transactions on Systems, Man and Cybernetics, vol. 9, pp. 62-66, 1979.

[34] P. V. C. Hough, "Method and means for recognizing complex patterns," U.S. Patent 3,069,654, Dec. 18, 1962.

[35] S. Y. Chen, J. W. Hsieh, and D. Y. Chen, "Jointing Edge Labeling and Geometrical Constraint for Lane Detection and its Application to Suspicious Driving Behavior Analysis," Journal of Information Science and Engineering (JISE), vol. 27, no. 2, pp. 715-732, March 2011.

[36] B. R. Nadra, H. Mohamed, and B.A. Hanene, "A Comparative Study of Vision-based Lane Detection Methods", In Advanced Concepts for Intelligent Vision Systems (ACIVS), Belgium, pp. 46-57, 2011.

[37] B. Jahne, "Digital Image Processing," *Springer Verlag*, 2002.

[38] R. C. Gonzalez, and R. E. Woods, Digital Image Processing (3rd Edition), *Prentice Hall*, 2008. http://folk.uib.no/eha070/mat262/lectures%202011/DIP3ELecture11_2011.pdf

[39] M. E. Wall, A. Rechtsteiner, and L. Rocha, "Singular Value Decomposition and Principal Component Analysis", A Practical Approach to Microarray Data Analysis. (Berrar DP, Dubitzky W, Granzow M, eds.), Kluwer: Norwell, pp. 91-109, Mar 2003.

[40] D. Leykekhman, "Lecture 9. Linear Least Squares. Using SVD Decomposition", 2008. http://www.math.uconn.edu/~leykekhman/courses/MATH3795/Lectures/Lecture_9_Linear_least_squares_SVD.pdf

[41] S. S. Huang, C. J. Chen, P. Y. Hsiao, and L. C. Fu, "On-Board Vision System for Lane Recognition and Front-Vehicle Detection to Enhance Driver's Awareness," IEEE Intl. Conf. on Robotics and Automation, New Orleans, Los Angeles, USA, pp. 2456-2461, 2004.

[42] H.A. Mallot, H.H. Biilthoff, J.J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," in Biological Cybernetics, vol. 64, no. 3, pp. 177-185, 1991.

[43] V. Cantoni, L. Lombardi, M. Porta, and N. Sicard, "Vanishing Point Detection: Representation Analysis and New Approaches," in Proceedings of 11th IEEE International Conference on Image Analysis and Processing, pp. 90-94, Sept. 26-28, 2001.

[44] C. C. Wang, S. S. Huang, and L. C. Fu, "Driver assistance system for lane detection and vehicle recognition with night vision," IEEE Intl. Conf. on Intelligent Robots and Systems, Alberta, Canada, pp. 3530-3535, 2005.