Thesis for the Degree of Master

Robust Lane Detection Method Using Bird's-eye View Transform

버드아이뷰 변환을 통한 안정적 차선검출기법

December 2010

Department of Electronic Engineering

Graduate School of Soongsil University

by

QING LIN

Thesis for the Degree of Master

Robust Lane Detection Method Using Bird's-eye View Transform

버드아이뷰 변환을 통한 안정적 차선검출기법

December 2010

Department of Electronic Engineering

Graduate School of Soongsil University

by

QING LIN

Thesis for the Degree of Master

Robust Lane Detection Method Using Bird's-eye View Transform

A thesis supervisor: Professor Hern Soo Hahn

Thesis submitted in partial fulfillment of the requirements
for the Degree of Master

December 2010

Department of Electronic Engineering

Graduate School of Soongsil University

by

QING LIN

To approve the submitted thesis for the Degree of Master
by Qing Lin

Committee Chair	Hyung Tai Cha	(signature)
Committee	Min Chul Hong	(signature)
Committee	Hern Soo Hahn	(signature)

December 2010

Graduate School of Soongsil University

ACKNOWLEDGEMENTS

On the completion of this master thesis, I have spent 2 years in the electronic engineering department in Soongsil university. There are lots of people who devote their time and energy generously to help me persue my master studies in a foreign country. First of all, special thanks should be delivered to my supervisor – Prof. Hernsoo Hahn. He not only gives me a wonderful instruction to the research world, but also teaches me how to become a specialist with a global mind.In addition, Prof. Youngjun Han also gives me much help, so do my seniors Jaehyoung Yu, Jaedo Kim, Young-Suk Ji and my labmates Sungji Han, Yonghee Hong, and ByeongYeol Kim. I should express my sincere thankfulness to all of them for their generous helps in my studies and everyday life in a foreign country.

Secondly, cordial thanks should be given to my dear parents, their concerns and supports always give me astonishing powers whenever I encountered difficulties in a foreign country. Moreover, they have been coping with everything in the family so as to allow me to concentrate on my studies. I really appreciate everything they have done for me from the bottom of my heart.

And last but not least, I want to thank Prof. Maoyong Cao in China, for his kind arrangement of my faculty position in Shandong Univ. of Science and Technology, as well as my colleagues and friends in China, who gave me many helps and advices while I am away from homeland.

Table of Contents

List of Figures

List of Tables

ABSTRACT

Robust Lane Detection Method Using Bird's-eye View Transform

LIN, QING
Department of Electronic Engineering
Graduate School
Soongsil University

This thesis presents a real-time lane detection method based on bird's-eye view. Unlike many existing methods that pay much attention on the post-processing stage to fit lane-model among a great deal of outliers, the proposed method aims at removing those outliers as much as possible at feature extraction stage, so that the searching space at post-processing stage can be greatly reduced. To achieve this goal, a multi-channel Haar-like filter is developed to get lane-mark patterns, and a directional feature-points tracing algorithm is proposed to extract directional feature-point links. Based on these links, a simplified parabolic-model is fitted using template -matching. To identify lane-mark types, a Bayesian probability model is employed. Experimental results show that the proposed method can detect lanes effectively in various road environments, and the algorithm can achieve an accuracy rate around 90% at an average speed of 8ms/frame at the image size of 320×240.

국문초록

버드아이뷰 변환을 통한 안정적 차선검출기법

전자공학과 림 청

指 導 敎 授 한헌수

이 논문에서는 버드-아이뷰 기반 실시간 차선 검출 기법을 보인다. 존재하는 많은 기법들은 전처리 단계에서 수많은 잡음들 중에서 차선을 찾기 위해 복잡한 계산을 하는데 본 논문에서 제안하는 기법의 목적은 특징 추출 단계에서 가능한 많은 잡음을 제거하여 전처리 단계에서 검색 영역을 획기적으로 줄이는 것이다. 목표를 달성하기 위해 다중 채널의 Haar-like 필터로 차선의 패턴을 얻고, 지향성 특징점 추적 알고리즘을 제안하여 지향성 특징점 연결을 제안한다.

이 연결 기법을 기반으로 간이화된 포물선 모델은 템플릿 매칭을 사용하여 구해진다. 차선 종류를 식별하기 위해 베이시안 확률 모델을 이용하였다. 실험 결과는 제안하는 방법이 다양한 도로 환경에 서도 효과적으로 검출할 수 있는 것을 보인다. 제안하는 알고리즘은 해상도 320×240 영상으로 8ms/frame의 속도에서 약90%의 정확도를 보였다.

Chapter 1 Introduction

1.1 Motivation

In recent years, the topic of safety-driving has received more and more attentions all over the world. According to statistic report, traffic accidents can cause injury to more than 10 million people per year around the world. Yet the major cause is largely the improper driving caused by driver's inattention and fatigue. In order to avoid traffic accidents, many researchers are developing driving-assistance systems for adding more safety to driving.

One of the major tasks of driving-assistance system is to sense the road environment, and detect traffic entities like lane-marks, traffic-signs and nearby vehicles. Among them, the detection of lanes plays a very important role as it provides the correct path for driving on the road. In this thesis, a real-time lane detection method is discussed, based on which a lane detection system is developed.

The lane detection system is composed of sensing hardware and signal processing software. For sensing hardware, various sensing equipments can be used, such as radar, infrared camera and CCD (charge-couples device) camera. For consideration of both the cost and performance, the CCD camera is adopted to sense the road environment. For image processing software, an efficient lane detection algorithm is developed to identify lane-mark positions and types in captured images.

1.2 Related work

Many approaches for lane detection have been proposed, and most of them follow a similar flow which is composed of two stages[1]. In the first stage, lane-mark features are extracted from the road images. These features can be edges, textures, colors or motion vectors. Next, a post-processing stage starts to fit the lane-mark feature points to a certain kind of lane model.

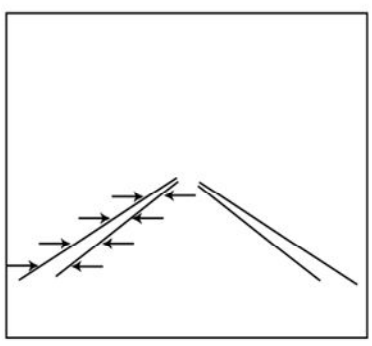
1.2.1 Lane detection in two views

Some researchers run these two stages directly on the input image in a perspective-view, while many others have found it much more helpful to operate on a bird's-eye view transform of the original perspective-view image. Generally, detecting lanes directly on the input image will waste no time over transforming the input image into bird's-eye view space. However, as introduced in [2], at least two disadvantages will come with apparent perspective effects: non-constant lane mark width, and non-even distance representation for pixels at different locations of image [Fig. 1-1]. A removal of such perspective effect using bird's-eye view transform will greatly help lane detection at both feature extraction stage and lane-model fitting stage.

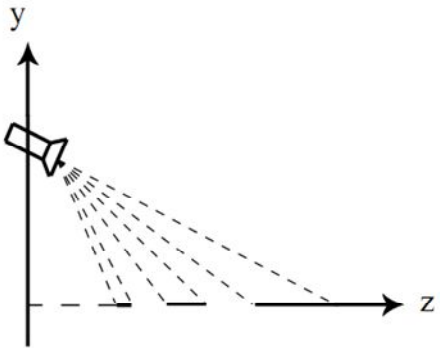
For feature extraction stage, on a bird's-eye view image, the left and right lane-marks appear in parallel bars with the same width

from the bottom to the top of image. This uniform and particular pattern can be easily extracted by using simple filters that can operate very fast. The size and shape of the filter mask can be fixed rather than changing from bottom to the top of image. While on the perspective-view image, because of the perspective effect, the extraction of lane-mark features should be based on searching for patterns with different size at different locations of the image. Therefore, it is generally much easier to extract lane-mark features on bird's-eye view compared to perspective-view.

For lane-model fitting stage, lane-mark boundaries on the bird's-eye view can be well represented by simple curve model like parabola. While on the perspective-view, there are almost no simple curve model can be directly used to represent the lane properly. Many methods have to use hybrid model or deformable model which introduce more complex parameters into the fitting process.



(1) Non constant lane-mark



(2) Non even distance for different pixels

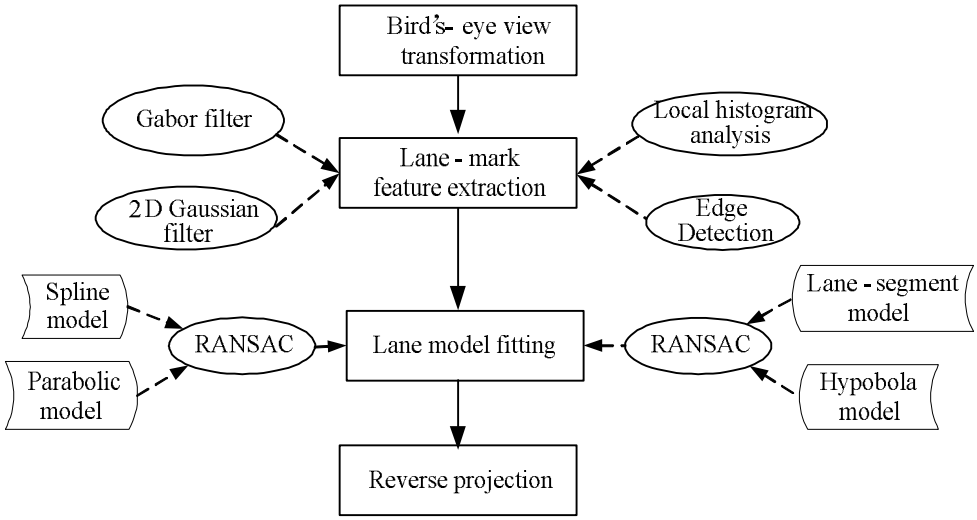
[Fig. 1-1] Perspective effects in the original input image

Since bird's-eye view brings many benefits for lane-detection compared with perspective-view, many researchers choose to detect lane-marks and fit lane-models on the bird's-eye view, and then map the fitted-lanes back to the perspective-view.

1.2.2 Lane-detection on bird's-eye view

Due to the convenience of extracting lane-mark features and fitting lane-models on a bird's-eye view, many lane detection methods using bird's-eye view have been developed, and various lane-models and fitting algorithms have been proposed.

K. Kaliyaperumal[3] detected lanes in radar images based on a deformable template model and Metropolis fitting algorithm. Y.Wang[4] proposed a B-snake lane model and used iterative mean square error minimization to fit B-snake parameters. Zu Kim[5] performed lane detection by using cubic-spline model, while C. Lipski[7] used a lane-segment model, both of them used RANSAC (RANdom Sample Consensus) algorithm for feature points fitting. Parabolic model is used by Joel C. McCall[1] with a statistical and motion based fitting algorithm. In addition, Qing Li[7] proposed a lane detection algorithm which uses adaptive randomized Hough transform to fit the lane boundary without using any pre-defined road model. [Fig. 1-2] shows a general lane-detection flowchart based on bird's-eye view.



[Fig. 1-2] General lane detection flowchart based on bird's-eye view

Instead of doing too much careful processing at lane-mark feature extraction stage, most of these methods tend to detect lane-mark features coarsely, while depending on the complex fitting algorithm to remove a great deal of outliers for the determination of actual lane-mark boundaries. On one hand, iterative fitting with coarsely extracted feature points can be more robust in various road conditions for the guarantee of true lane feature existence in coarse feature sets. While on the other hand, large amount of feature points enlarges the searching space of iterative fitting algorithm dramatically, resulting in a great increase of the computation cost at the post-processing stage.

Heavy computation load at the post-processing stage turns out to be a very serious problem for the real-time application of lane

detection algorithm. Tabel 1-1 lists some of the famous algorithm that operates on bird's-eye view. It can be observed that many of these algorithms just focused on detecting local lane-mark features based on local intensity difference, while didn't consider global distribution feature of lane-mark patterns. On the bird's-eye view image in rough road conditions, many local patterns caused by nearby vehicles, traffic sings, and guardrails will be very similar to that of lane-mark patterns. These noise patterns can not be filtered out by simply checking the local features at the feature extraction stage. These noise feature points enlarge the feature space for the next fitting stage.

[Table 1-1] Time performance of lane detection algorithms

Author	Method	Speed	Experiment platform
Yue Wang[4] (2004)	Edge map B-snake model MMSE fitting	500ms/frame	Typical PC Intel Pentium 3 1GHz
Cláudio Rosito[8] (2005)	Edge map linear-parabolic model Hough fitting	60ms/frame	Typical PC Intel Pentium 4 1.8GHz
Zu Kim[5] (2006)	ANN classifier cubic spline model RANSAC fitting	30ms/frame ~70ms/frame	Laptop PC Intel Pentium M 2GHz
Christian Lipski[6] (2008)	Local histogram patch lane-segment model RANSAC fitting	100ms/frame	Typical PC Intel Pentium 4 2GHz
Mohamed Aly[9] (2009)	2D Gaussian filter Bezier spline model RANSAC fitting	20ms/frame	Typical PC Intel Core2 2.4GHz

To find a correct lane-model parameters among such a noisy feature space, many of these algorithms have to use heavy fitting algorithm like Hough transform, MMSE(Minimization of Mean Square Error) and RANSAC(RANdom Sample Consensus). As a result, those heavy fitting algorithm increase the computation cost dramatically. The the processing times of these algorithms are generally over 20ms/frame on a typical PC platform. This turns out to be a processing speed which is not suitable for a fast embedded application.

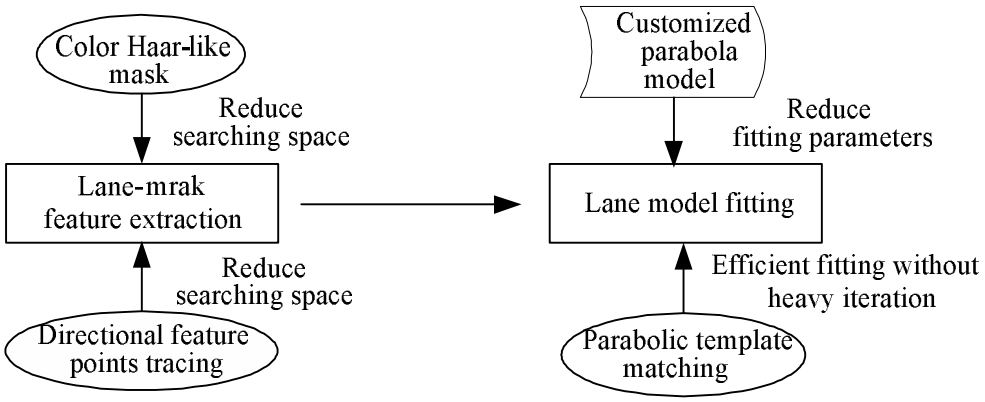
1.3 Overview of the proposed method

In order to limit the size of the feature space to reduce the computation cost at the post-processing stage, the method proposed in this thesis places a higher value at lane-mark feature extraction stage, that is to identify true lane-mark feature-points accurately among various noises on the road surface, so as to remove outlier feature points as much as possible at the feature extraction stage.

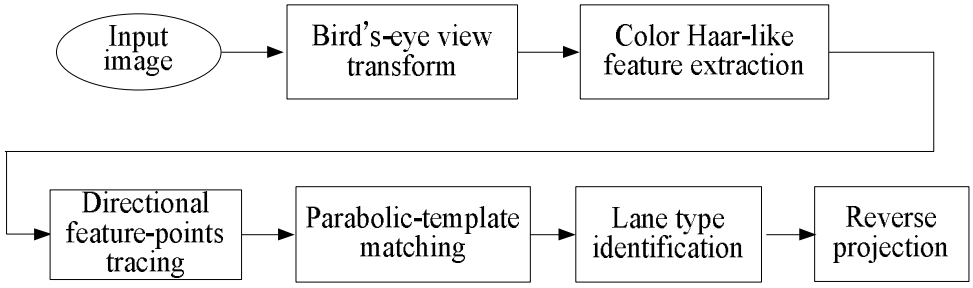
In this thesis, a kind of directional feature-point link is proposed as a new lane-mark feature, and a directional feature-points tracing algorithm is developed to search for this kind of directional feature-point links in a candidate feature-points set obtained using multi-channel Haar-like filter.

At post-processing stage, a simplified parabolic model is fitted based on directional feature-point links found in the feature extraction

stage. The simplified parabolic model can be fitted using just two parameters based on a parabolic template matching algorithm, which is an efficient fitting algorithm without heavy iterations. [Fig. 1-3] shows an overview of the proposed lane-detection method.



[Fig. 1-3] Overview of the lane-detection method



[Fig. 1-4] General flowchart of the lane-detection method

A general flowchart of the proposed lane-detection method is shown in [Fig. 1-4]. The whole detection algorithm is composed of 6 major modules.

(1) Bird's-eye view transform is used to transform the input image from perspective-view to bird's-eye view so as to remove the perspective effects of the on-board camera.

(2) On bird's-eye view images, a Haar-like filter is applied to the multiple channels of bird's-eye view image to extract lane-mark patterns in specific colors. The feature points obtained in this step constitutes candidate feature-points set to be input to the next feature-points tracing step.

(3) Inside the candidate feature-points set, directional feature-points tracing algorithm is applied to find directional feature-point links. Global features like orientation and connectivity length are checked to remove noise feature points.

(4) Based on extracted directional feature-point links, parabolic templates are generated and matched to the feature-points so as to find out the best matching curve.

(5) After the best-matching curve is obtained, the color and continuity of lane-marks are estimated to identify the type of lane-mark along the best-matching curve.

(6) In the final step, the best-matching curve found on the bird's-eye view image is projected back to the original perspective view to label the lane-boundaries in the original image.

Chapter 2 Bird's-eye View Transform

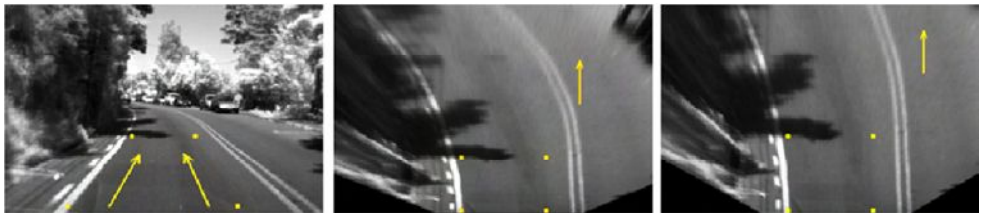
In many vision-based lane detection systems, the camera is mounted on the upper center of the vehicle's windshield and looking ahead to the road. Under this image acquisition conditions, the road image captured by camera appears in a perspective view. Bird's-eye view transform is capable of remapping the input image from the perspective view to a scaled view from the top towards an assumed planar road surface. From this remapped view, perspective effects can be approximately removed. There are generally two methods can be used to generate a bird's-eye view.

2.1 Warp perspective mapping

As in [5], a four-points correspondence can be used for the mapping from the perspective view into the bird's-eye view image. The mapping is achieved by selecting four points P_W on the ground plane when calibrating camera, and by using the planar ground assumption. P_W in the input image will be viewed as P_I. The mapping from P_I to P_W can be achieved by an affine matrix A as follows:

$$P_w = A \cdot P_I \qquad (2.1)$$

The calculation of A using the above equation can be easily achieved. Then, the whole input image can be mapped, pixel-by-pixel, using A. One main benefit of this warp perspective mapping (WPM) is that the used distance scale can be adjusted by selecting different sets of four corresponding points. This proved to be useful for detecting discontinuous lane-marks as well as for further forward looking situations. Another benefit is that no intrinsic or extrinsic parameters of camera(s) are needed. [Fig. 2-1] shows an example of using WPM. [Fig. 2-1](2)(3) are bird's-eye image obtained by using WPM but with different distance scales. Four points correspondence (points shown in yellow) is established by calibration.



(1) Input image in perspective-view

(2) Warpping image using small distance scale

(3) Warpping image using larger distance scale

[Fig. 2-1] An example of bird's-eye view transform using WPM

2.2 Inverse perspective mapping

Inverse perspective mapping (IPM) as introduced in [10] is another way to generate a bird's-eye view from the input image [2, 11]. The knowledge of the camera parameters (extrinsic and intrinsic) is

required for the application of the IPM transform:

(1) Viewpoint: camera position in world coordinate system $C=\{l,h,d\}$.

(2) Viewing direction: optical axis is defined by two angles:

γ_0: the angle formed by the projection of the optical axis on the xz plane, as shown in [Fig. 2-2](1),

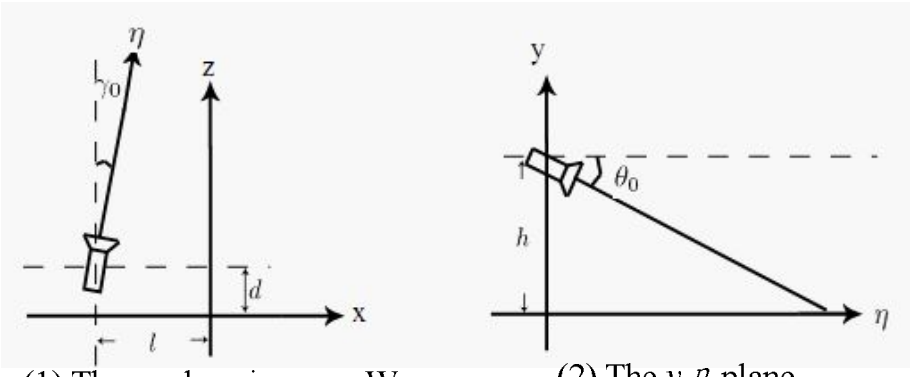
θ_0: the angle formed by the optical axis and axis η, as shown in [Fig. 2-2](2).

(3) Aperture: camera angular aperture is $2a_u$ in row direction and $2a_v$ in column direction. While :

$$\partial_u = \arctan(u_0 / F) \qquad \partial_V = \arctan(v_0 / F) \qquad (2.2)$$

(u_0, v_0) is the camera's focal center, and F is focal length.

(4) Resolution: camera resolution is n × m.



(1) The xz plane in space W (2) The y η plane

[Fig. 2-2] IPM model for bird's-eye view transform

Mathematically, IPM can be modeled as a projection from a 3D Euclidean space W, containing elements $(x, y, z) \in R^3$, onto a planar (2D) subspace of R^3, denoted by I, with elements $(u, v) \in R^2$. The transformation from I to W is given as:

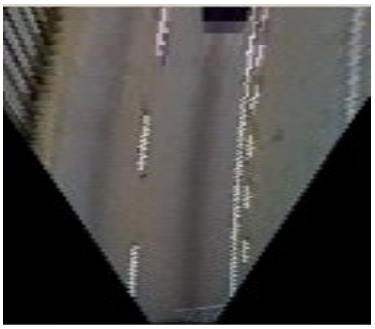
$$\begin{aligned} x(u,v) &= h \cdot \cot\{(\theta_0 - \partial_u) + u \cdot 2\partial_u / (m-1)\} \cdot \cos\{(\gamma_0 - \partial_v) + v \cdot 2\partial_v / (n-1) + 1\} \\ y(u,v) &= 0 \\ z(u,v) &= h \cdot \cot\{(\theta_0 - \partial_u) + u \cdot 2\partial_u / (m-1)\} \cdot \sin\{(\gamma_0 - \partial_v) + v \cdot 2\partial_v / (n-1) + d\} \end{aligned} \quad (2.3)$$

while the mapping from W to I is:

$$\begin{aligned} u(x,0,z) &= \frac{[\arctan\{h \sin\gamma(x,0,z)/(z-d)\} - (\theta_0 - \partial_u)] \cdot (m-1)}{2\partial_u} \\ v(x,0,z) &= \frac{[\arctan\{(z-d)/(x-l)\} - (\gamma_0 - \partial_v)] \cdot (n-1)}{2\partial_v} \end{aligned} \quad (2.4)$$

By using the above group of equations and filling the required camera parameters, the transform between perspective view and bird's-eye view domains can be easily carried out. [Fig. 2-3] shows an example of bird's-eye view transform using IPM.

(1) Original image



(2) Bird's-eye view image using IPM

[Fig. 2-3] An example of bird's-eye view transform using IPM

In the development of lane-detection system, we use a combination of the two methods. When doing experiments with different video clips, it is usually not an easy task to collect all the camera parameters associated with the video. In this case, WPM method is used to generate the bird's-eye view. The four points are selected manually from the left and right lane boundaries in the input image, In addition, the scale of bird's-eye view image can be adjusted during experiment by simply selecting another set of four points. However, it is obvious that doing bird's-eye view transform in this way is not very accurate. Therefore, when testing the system with a mounted camera in a vehicle, IPM method is used to generate an accurate bird's-eye view image using specific camera parameters.

Chapter 3 Lane-mark Feature Extraction

3.1 Multi-channel Haar-like filter

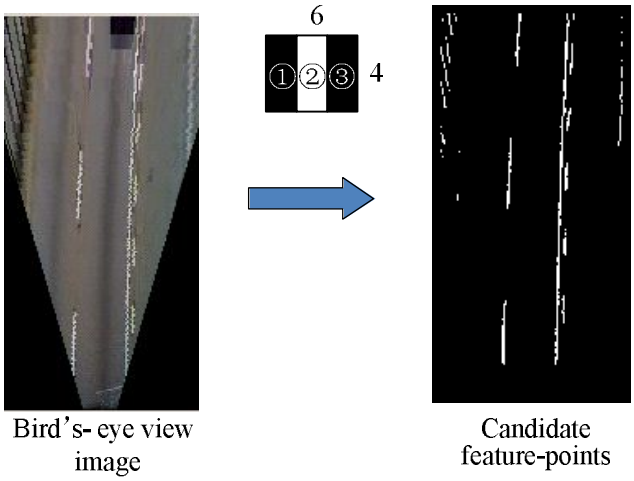
On transformed bird's-eye view image, lane-marks appear like thin rectangle patterns of local intensity difference of the road surface covering a certain length. To detect this kind of pattern on bird's-eye view image, Haar-like filter[12] is a very good choice.

As in [Fig. 3-1] shows, a 6×4 Haar-like filter is applied on the bird's-eye view image. The filter response is calculated using the following rules:

$$f(x,y)=\begin{cases} 1 & if\ \sum P_1(x,y)/\sum P_2(x,y) < T\ and\ \sum P_3(x,y)/\sum P_2(x,y) < T \\ 0 & else \end{cases} \quad (3.1)$$

Since the size of the Haar-like mask is fixed and small, the filter response at each pixel can be calculated directly without generating the integral image. In order to make the filter response more adaptive to various illumination conditions, instead of checking the intensity sum difference between the adjacent rectangle areas, the intensity sum ratio between adjacent rectangles are calculated and compared to a ratio threshold T. As in (3.1), $\sum P_1(x,y)/\sum P_2(x,y)$ is the intensity sum ratio between the left black box(labeled as ① in [Fig. 3-1]) and the middle white box(labeled as ② in [Fig. 3-1]), while

$\sum P_3(x,y) / \sum P_2(x,y)$ is the intensity sum ratio between the right black box(labeled as ③ in [Fig. 3-1]) and the middle white box(labeled as ② in [Fig. 3-1]). Generally, the ratio threshold T should less than 1.



[Fig. 3-1] Extract candidate feature-points using Haar-like filter

Rather than simply applying this Haar-like filter on gray-level image, it is applied to multiple channels

separately to take lane-mark color into consideration. If the input image is an RGB channel image, then the Haar-like filter is applied to R and B channels separately. As the general lane-mark colors in Korea are yellow, blue and white, and these three kinds of colors usually have dominant intensity values in R and B channels of RGB image. On the other hand, if the input image is an YUV channel image, then the Haar-like filter is applied to Y, U, V channels separately, since yellow and blue colors are much easier to identify on U and V channels.

By applying Haar-like filters to multiple channels of the bird's-eye view image separately, and then make the final response decision by considering the filter response in each separate channel, the candidate feature-points can be extracted more effectively.

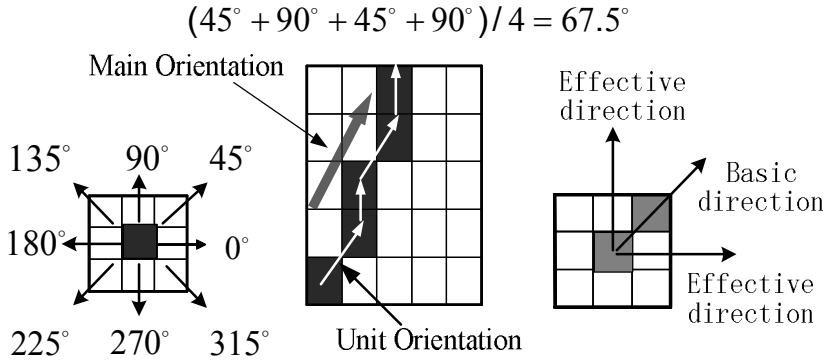
3.2 Directional feature-points tracing

After candidate feature-points are extracted, a directional feature-point tracing (DFPT) algorithm is used to further remove noise feature-points. Pixel-tracing is a very effective method to track object boundaries in certain geometries. Different kinds of pixel-tracing algorithm have been proposed[13][14], these methods used different pixel-tracing functions to detect certain types of geometries in noisy image environment. Here the general pixel-tracing algorithm is extended to identify lane-mark geometries.

In general, lane-mark feature-points should approximately lie on line-segments, which have specific directions and length on bird's-eye view image, DFPT algorithm is able to link the feature-points into line-segments so that noise feature-points link can be removed by checking the directions and length.

Feature-point links are defined as a vector which contains 5 parameters $< ID, P_{head}, P_{tail}, L, \alpha, >$. 'ID' is the label of this link, 'P_{head}' and 'P_{tail}' is the starting point and end point, 'L' is the length, and 'a' is the direction of this link. The direction of link is described by unit-direction and main-direction. The unit-direction is defined

according to the orientation of one pixel's 8 neighborhoods, as shown in [Fig. 3-2]. Based on the definition of unit-direction, the main-direction of edge-link is defined as the mean value of all unit directions between every two neighboring pixels: $\alpha = \sum U_i /(L-1)$, where $\sum U_i$ represents the sum of all unit directions. An example of main-direction calculation is also shown in [Fig. 3-2].

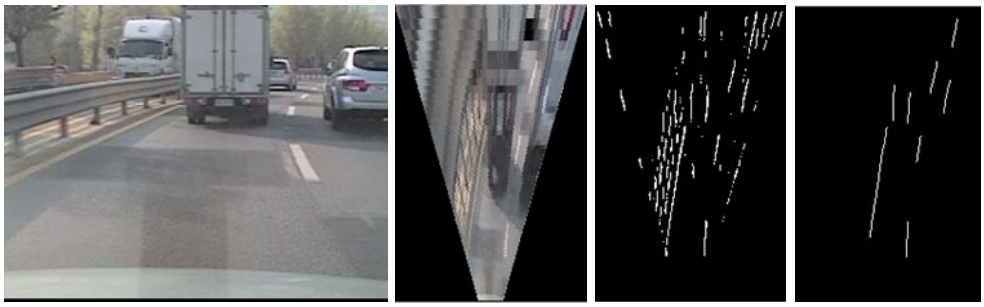


[Fig. 3-2] The definition of link direction

DFPT algorithm is developed based on an 8-neighborhood pixel-tracing algorithm. The DFPT algorithm is composed of two major steps: starting point scan and pixel tracing. Firstly, a raster scan is performed starting from bottom of image looking for starting points of a new link. When a starting point is found, pixel tracing starts to trace all the feature-points associated with the starting point. Basic and effective unit-directions are introduced to guide the tracing. Basic unit-direction is defined as the unit direction between the first two pixels in the link, which is regarded as the basic

orientation of this link. Effective unit-direction is the unit direction that is within ±45° to the basic unit-direction, as is shown in [Fig. 3-2].

The pixel-tracing process continues until one of the following two conditions are met: (1) No more connected pixels can be found. (2) The unit-direction of the next neighboring pixel falls out of effective direction range. In either case, the current link will terminate, and a new link will be initialized in the starting point scan process. Finally, when no more unlabeled starting points can be found, the whole tracing process will terminate, and the main-direction and length of each link is checked. The main-direction of edge-link should lie between 80° to 100° on the bird′-eye view image. The length of link should be larger than a threshold value as well. All links that violate this rule are discarded as noise. Some results of proposed DFPT algorithm are shown in [Fig. 3-3].



(1) Original image (2) Bird's-eye view (3) Haar filter result (4) DFPT result

[Fig. 3-3] Result of DFPT algorithm

Comparing with the tracing algorithm based on chain-code as the one used in [15], the proposed DFPT algorithm introduced unit-direction and main-direction to describe link path in a much finer way, which is able to track many link paths within the valid lane-mark direction range that chain-code based tracing algorithm can not capture.

3.3 Directional gap closing

Due to noise and improper threshold in the Haar-like filter, some lane-mark patterns may be broke up into many small disconnected segments. Since the lengths of these small segments are usually under the link length threshold, they can be easily filtered out as noise links, which cause the loss of real lane-mark feature points.

To deal with this problem, a directional gap closing (DGC) algorithm is added to produce more complete links. After feature-point links with valid orientations and a minimum length are obtained, the DGC step begins. In this step, links are extended by adding new pixels according to the link orientation to fill the possible gaps which split a complete link. The maximum number of added points is determined by a user-defined value. Generally, 5 pixels are enough to fill the gaps between splitted links.

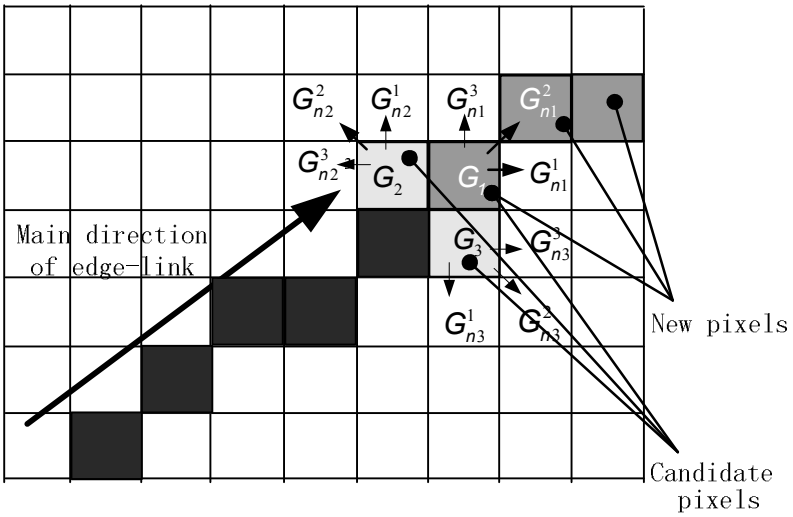
New points are selected from the neighboring points of the starting point and end point of one link along the link orientations. As [Fig. 3-4] illustrates, for a given link, three points G_1, G_2, and G_3,

which lie in the effective direction around tail point are considered as candidates for new points. If the Haar-like filter response of G_1, G_2 and G_3 are larger than a gap-closing threshold, then for each of these three candidate points, a gap-closing score is calculated as: $M_i = G_i + \max(G_{n1}^i, G_{n2}^i, G_{n3}^i)$, where G_i indicates the filter response at the i_{th} candidate points, while $G_{n1}^i, G_{n2}^i, G_{n3}^i$ refer to the filter response at the three neighboring points around i_{th} candidate points. The sum of the candidate point's filter response and the maximum filter response of this point's three neighbors is calculated as a gap-closing score M_i. The point with the maximum gap-closing score is finally selected as the new point to be added to the link. This link extension process will continue until one of the three following conditions is satisfied:

(1) The point in another link is detected in the neighborhood of newly added points. This means two links meet each other, so that they are merged into one complete link.

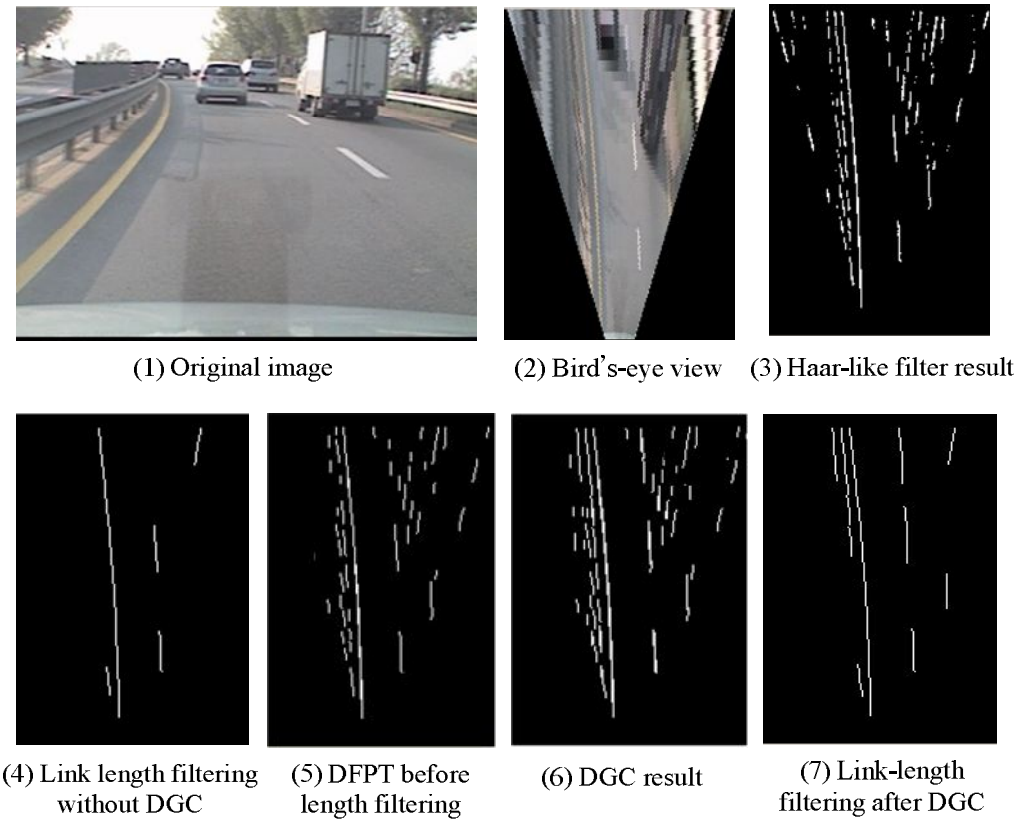
(2) The maximum number of new points is reached.

(3) There are no more points with filter response larger than the gap-closing threshold can be found.



[Fig. 3-4] Directional gap closing algorithm

After DGC step, those disconnected small segments which belong to one lane-mark can be linked together as one link. Then link length is checked to discard those small links with lengths below threshold. The effect of this DGC algorithm is shown in [Fig. 3-5]. Comparing this DGC algorithm with normal dilation operation, the proposed algorithm searches candidate new points along the direction of links, and these candidate points are evaluated by a gap-closing function to estimate their quality for recovery. Without these measures, as using a normal dilation operation, lots of noise pixels will be added during gap closing process.



[Fig. 3-5] The effect of DGC algorithm

Chapter 4 Lane Fitting and Type Identify

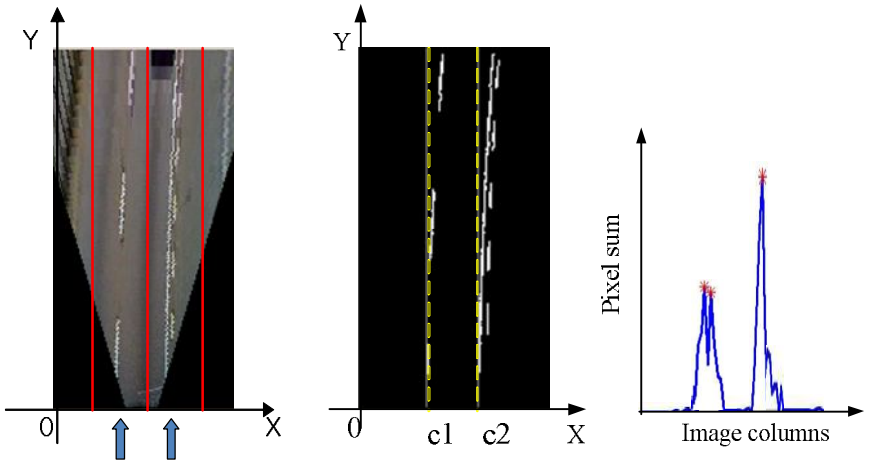
4.1 Parabolic-template initialization

After lane-mark feature-points are well extracted, the lane-model fitting process will start. On the bird's-eye view image, lanes can be well approximated by parabolic curve[16]. If image coordinate system is defined as in [Fig. 4-1], then the standard parabolic equation can be given as: $x = ay^2 + by + c$. However, the linear parameter b is found to be around zero most of the time, hence the parabolic model is simplified as: $x = ay^2 + c$. There are only two parameters need to be fitted in this simplified model, a linear parameter 'c' and a curvature parameter 'a'.

First, linear parameter 'c' is estimated as an initialization for the parabolic-templates. As shown in [Fig. 4-1](1), the initial searching range for 'c' is defined as from the width/5 to 4*width/5, and then at each position from width/5 to 4*width/5, the feature-points that lie on each column is accumulated vertically, and local maxima in the left and right part of the image is found on the vertical projection histogram as in [Fig. 4-1](3), the position on 'X' axis with the maximum feature-points accumulation is regarded as linear parameter 'c', this is shown in [Fig. 4-1](2).

As long as linear parameter 'c' is determined in the first initial frame, then in the following consecutive frames, the searching range

of 'c' will be set around the position of 'c' in the previous frame.



[Fig. 4-1] Initialization of the linear parameter

4.2 Parabolic-template matching

The next step is to estimate curvature parameter 'a'. This is done via a parabolic-template matching process. Based on the position of linear parameter 'c' on the 'X' axis, a bunch of parabolic-templates are generated by varying curvature parameter 'a' in the equation $x = ay^2 + c$. The variation range of 'a' is set from -0.0004 to 0.0004 in a unit step of 0.00001. For every value of 'a' in this range, a specific parabolic-curve can be generated, and a matching score is calculated based on the distance from every feature-point to this

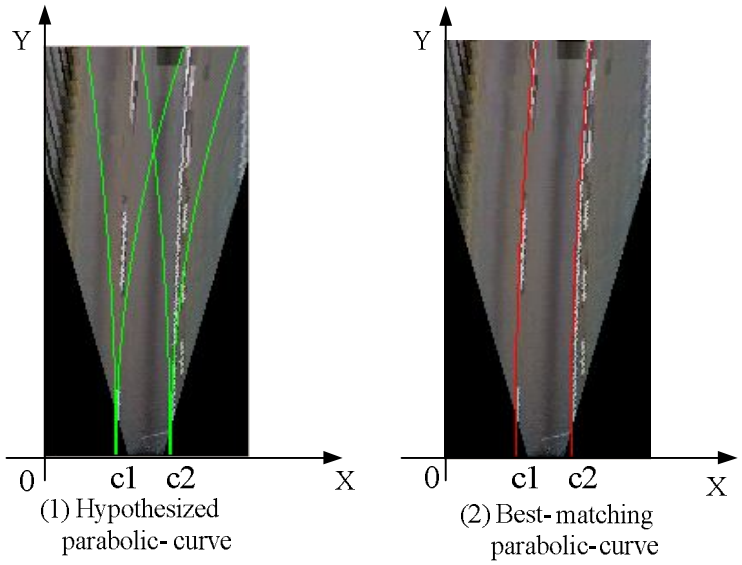
hypothesized curve.

To calculate the matching score, a weight function e(x) should be calculated first. The definition of e(x) is given in (4.1), d is the horizontal distance between each feature-point and the curve. For one feature-point, one weighted value e(x) will be calculated based on d, the larger the distance is, the smaller weight will be. Therefore, the weight function e(x) gives feature-point which near the curve higher weight, while gives those lie far to the curve lower weight. The sum of all weighted value of each feature-point is regarded as the matching score. The higher the score value is, the more feature-points located near this curve.

$$\mathrm{e(x)}=\begin{cases} 2 & if\ x = a_k y^2 + c \\ 1/\mathrm{d} & if\ x \neq a_k y^2 + c \end{cases} \qquad d = | x - (a_k y^2 + c) | \tag{4.1}$$

$$s_k = \sum_{i=1}^{n} e(x_i) \tag{4.2}$$

Finally, the curve with the largest matching score is regarded as the best matching curve. An example is shown in [Fig. 4-2].



[Fig. 4-2] Parabolic-template matching

4.3 Lane curvature estimation

In the defined lane-model: $x = ay^2 + c$, when curvature parameter 'a' equals to zero, the fitted curve is a straight line. Otherwise, the fitted curve is a left-turning(a<0) or right-turning(a>0) curve.

When vehicles drive on the road, straight lane will appear most of the time, and curve lane only appears on a specific portion of the road. Due to this fact, in order to make the lane-model fitting more robust to noise, a lane-curvature estimation part is added rather than output the best-matching curve directly.

A curve-energy score is calculated by comparing a curve evaluation score S_C and a line evaluation score S_L. The definition of

S_C and S_L is given in (4.3)(4.4). They are defined based on two weight function C(x) and L(x). The definition of C(x) is the same with the one used in the matching score calculation. L(x) is defined by the distance between feature-points and a straight line 'x = c'. The sum of L(x) at every feature-point is calculated as a line evaluation score.

$$C(x)=\begin{cases}2 & if\ x = ay^2 + c \\ 1/d & if\ x \neq ay^2 + c\end{cases} \quad d = | x - (ay^2 + c) | \quad s_c = \sum_{i=1}^{k} C(x_i) \qquad (4.3)$$

$$L(x)=\begin{cases}2 & if\ x = c \\ 1/d & if\ x \neq c\end{cases} \quad d = | x - c | \quad s_L = \sum_{i=1}^{n} L(x_i) \qquad (4.4)$$

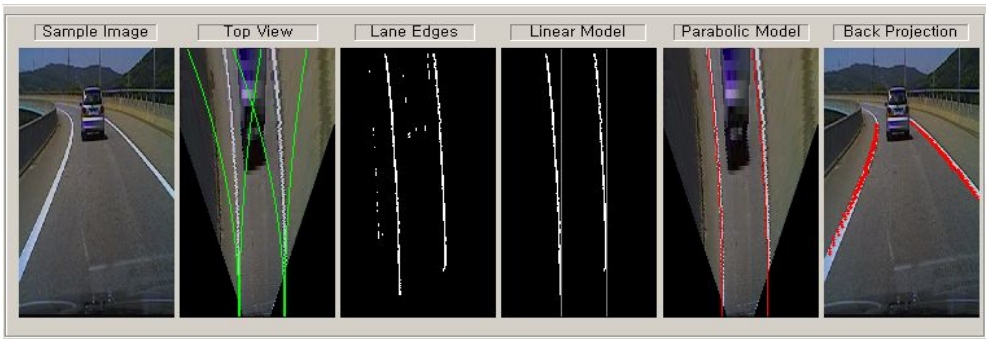
$$E = \sum_{i=1}^{k} C(x_i) / \sum_{i=1}^{n} L(x_i) \qquad (4.5)$$

The ratio between S_C and S_L is defined as the curve-energy score E. If the value of E becomes larger than 1, then the fitted curve is more likely to be a curve, while E becomes smaller than 1, then there is more chance to be a straight line. If the value of E is larger than a threshold, then it is assumed that curve lane is detected, otherwise straight lane is detected.

Moreover, since the curvature of lane usually does not change suddenly, therefore, lane curvature in the previous frames can be used to estimate curvature in the following consecutive frames. The matching range of parabolic-template will be set around the detected

curve in the previous frame, and the detected curvature will be compared with the curvature values in the history list to identify abnormal values. These inter-frame curvature estimation will make the lane-curvature detection more stable under rough conditions.

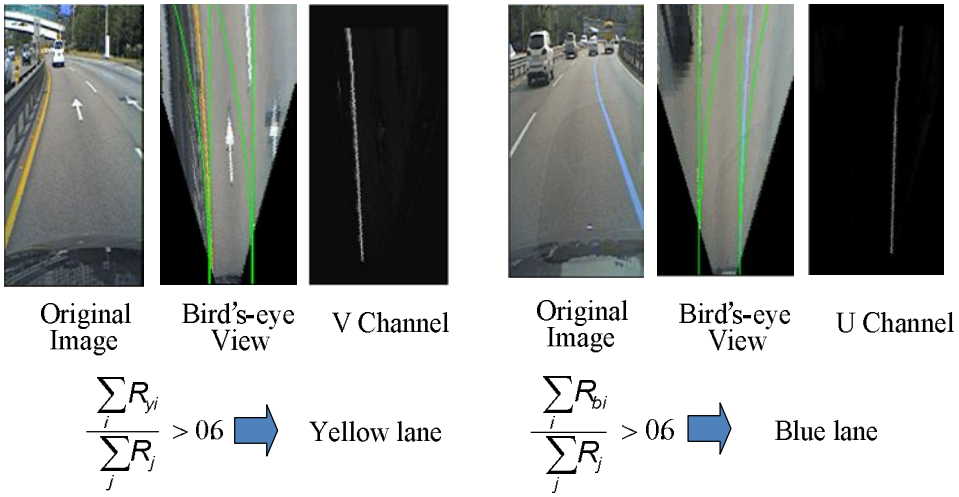
Finally, when both the linear part and curvature part of the parabolic curve can be determined, the parabolic curve can be projected from bird's-eye view into the perspective-view image. An example of the whole process is shown in [Fig. 4-3].



[Fig. 4-3] An example of the whole detection process

4.4 Lane type identification

Lane types can be identified by specific color and continuity. In Korea, there are generally three kinds of lane-mark colors: white, yellow and blue. These three colors are much easier to identify in the YUV color space. Therefore, color identification is done in the YUV color space. By using adaptive thresholds calculated from local histogram, yellow, blue dominant regions can be easily extracted from the U and V channels, as is shown in [Fig. 4-4]. Checking the number of feature-points in U channel first, and then calculating R_{yi} / R_i, R_{yi} indicates the number of feature-points in U channel, while R_i means the total number of feature-points. If this ratio is larger than a threshold, it is assumed yellow lane-mark is detected. If yellow lane-mark is not detected, then check the existence of blue lane-mark in the same way on V channel. If the lane-mark is neither yellow nor blue, then it is assumed to be white lane-mark.



[Fig. 4-4] Lane-mark color identification

In terms of the continuity of lane-mark, it can be classified as solid lane-marks and dashed lane-marks. In order to identify the continuity of lane-mark, a Bayesian probability model is employed. Given the best matching line-template L, two posterior probability functions (4.6) and (4.7) are involved to estimate the type of lane-mark, where $P(\mathbf{L} \mid Solid)$ and $P(\mathbf{L} \mid Dash)$ are the likelihood of solid and dashed lane-marks, while $P(Solid)$ and $P(Dash)$ are the corresponding prior probabilities.

$$P(Solid \mid \mathbf{L}) = P(Solid)P(\mathbf{L} \mid \text{Solid}) \quad (4.6)$$

$$P(Dash \mid \mathbf{L}) = P(Dash)P(\mathbf{L} \mid Dash) \quad (4.7)$$

Therefore, the given best matching line L can be classified as solid or dashed by using the following Bayes decision rule:

$$\mathbf{L} \in \begin{cases} solid & \mathrm{P}(solid \mid \mathbf{L}) \geq \mathrm{P}(dash \mid \mathbf{L}) \\ dash & \mathrm{P}(solid \mid \mathbf{L}) < \mathrm{P}(dash \mid \mathbf{L}) \end{cases} \quad (4.8)$$

The prior probability $\mathrm{P}(Solid)$ and $\mathrm{P}(Dash)$ can be estimated using some prior knowledge. For example, by using lane-mark colors appear on a specific type of road, like most yellow and blue lane-marks in Korea are solid lane-marks. The likelihood is calculated based on the ratio between the number of overlapped feature-points on the fitted curve and the real length of the fitted curve. As is shown in (4.9), $\sum E_t$ is the number of feature-points on the fitted curve, and $\sum L_j$ is the real length of the fitted curve.

$$\mathrm{P}(\mathbf{L} \mid Solid) = \sum_t E_t / \sum_j L_j \quad (4.9)$$

Chapter 5 Experimental Results

5.1 Speed performance evaluation

The proposed lane detection algorithm has been implemented in Visual C++ 6.0 on a PC running WindowsXP system. For video clips with 320×240 image size, the processing time is around 6~10ms/frame with an Intel Core2 1.86GHZ processor and 2GB DDR memory. Larger images will be down-sampled to 320×240 to keep the processing speed at a similar level. [Table 5-1] lists the comparison of speed performance with some famous algorithms.

[Table 5-1] Speed performance evaluation

Author	Method	Speed	Experiment platform
Yue Wang (2004)	Edge matching B-snake model MMSE fitting	500ms/frame	Typical PC Intel Pentium 3 1GHz
Cláudio Rosito (2005)	Edge distribution linear-parabolic model Hough with QEM	60ms/frame	Typical PC Intel Pentium 4 1.8GHz
Zu Kim (2006)	ANN classifier cubic spline model RANSAC	30~70ms/frame	Laptop PC Intel Pentium M 2GHz
Christian Lipski (2008)	Local histogram patch lane-segment model RANSAC fitting	100ms/frame	Typical PC Intel Pentium 4 2GHz
Mohamed Aly (2009)	2D Gaussian filter Bezier spline model RANSAC fitting	20ms/frame	Typical PC Intel Core2 2.4GHz
Proposed method	Haar-like filter Parabolic model Template matching	6~10ms/frame	Typical PC Intel Core2 1.86GHz

The proposed algorithm has also been ported onto the embedded board TMS320DM6437 with CCS3.3(Code Composer Studio). TMS320DM6437 board has a 600MHZ Processor and 128MB memory, the ported program can achieve a processing speed of 30~50ms/frame on this board, which means the algorithm is able to process 20~33 frames/sec for a fast embeded application.

5.2 Accuracy performance evaluation

To evaluate the accuracy performance of the proposed lane detection algorithm, the testing results on 5 video clips are presented here. The five video clips cover various road conditions including different lane types, all kinds of lane variation situations (emerging/terminating/merging/switching/intersecting), complex road surfaces (clutter markings, shadows), significant obstacles (vehicles running on the road), and various illumination conditions(tunnels, night-time).

For the accuracy evaluation experiment, we labeled the lane-mark center manually in the testing frames as ground-truth data. Then the output of the detection algorithm is compared with the ground-truth data. The comparison results are evaluated based on the rules illustrated in [Fig. 5-1]. For parallel deviation from the center of lane-mark, 5 pixels deviation can be accepted, while for angular deviation, 5 degree deviation can be accepted. Based on this evaluation rule, a list of the evaluation results are shown in

[Table 5-2].

As is shown in [Table 5-2], more than 500 frames are selected in each video clip for the testing. "Good" testing results mean the detected lane-mark positions are acceptable and the types of lane-marks(including color and continuity) are correct. Otherwise, the testing results are reported as "Bad" results. The accuracy rate is calculated using the percentage of good frames among the total testing frames.

In addition to the quantitative testing results listed in [Table 5-2], some qualitative detection results from the above 5 video clips are shown in [Fig. 5-2] to [Fig. 5-4].

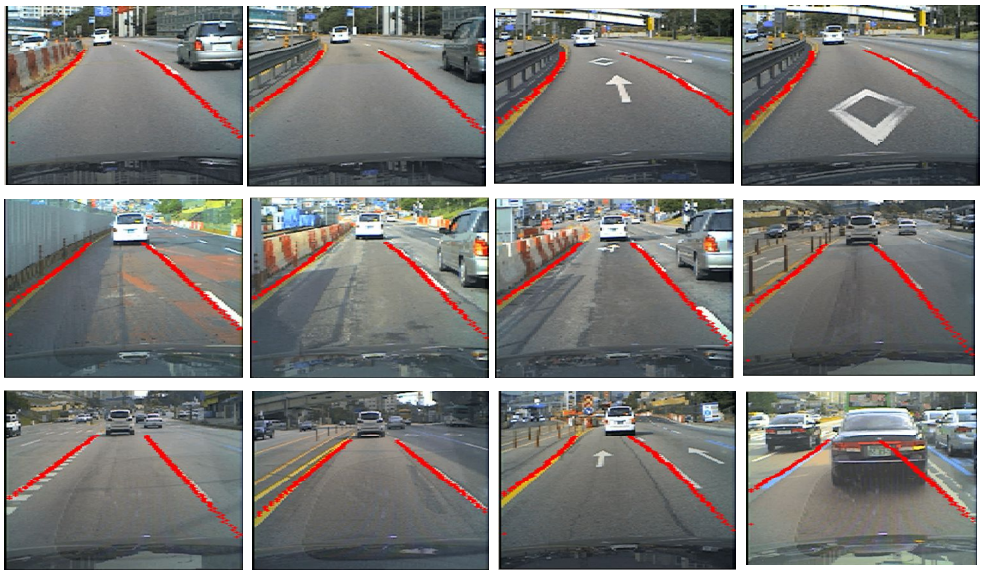
(1) Parallel Deviation (2) Angular Deviation

[Fig. 5-1] Accuracy evaluation rules

[Table 5-2] Accuracy testing results

Test Video ID	Good	Bad	Total	Accuracy
1	484	96	530	91.3%
2	518	42	560	92.5%
3	442	88	530	83.4%
4	503	77	580	86.7%
5	466	84	550	84.7%

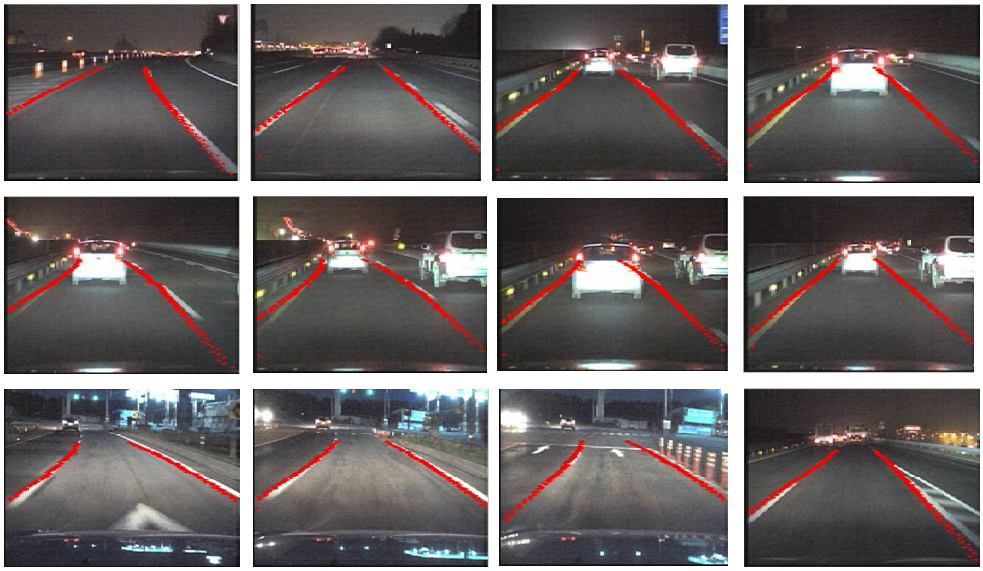
[Fig. 5-2] includes some frames from testing video-clip 3#. Video-clip 3# is captured on the city road and is very challenging. It contains various lane shapes, rough road surfaces, complex road-side facilities, different types of lane-marks, various traffic-signs on road surface, and crowded vehicles on the road. It can be observed that the proposed lane-detection algorithm can work under such complex road environment.



[Fig. 5-2] Results from video-clip 3#

[Fig. 5-3] shows the detection results from video-clip 1#. Video-clip 1# is captured on the highway, which contains some high curvature lane as well as various tunnels. [Fig. 5-4] shows some detection results at night-time, in which the illumination conditions are much different from that at day-time.

[Fig. 5-3] Results from video-clip 4#



[Fig. 5-4] Night-time testing results

Chapter 6 Conclusions

In this paper, a lane detection method based on bird's-eye view is presented. Compared with other lane-detection methods using bird's-eye view, the proposed algorithm pays much attention at the feature extraction stage to achieve a much higher processing speed. By using multi-channel Haar-like filter and the directional feature-point tracing algorithm, directional feature-points links which belong to lane-marks can be extracted very effectively in noisy environment. Most of the noise and outliers can be removed at this stage, this helps to limit the searching space for the following fitting process, so that the computation cost for line-template matching can be greatly reduced. Furthermore, when applying the proposed detection algorithm to consecutive video frames, inter-frame level estimation can be involved to get optimized lane parameters by accumulating and analyzing lane parameters that are obtained in the single-frame level. The output of the proposed lane detection method is able to provide accurate information about lane positions and lane types for further applications like lane departure warning, lane tracking or forward vehicle detection.

References

[1] J. C. McCall and M. M. Trivedi. “Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation.”. IEEE Trans. Intell. Transp. System.Vol. 7. No. 1. pp.20-37. 2006.

[2] M. Bertozzi and A. Broggi. “GOLD - A parallel real-time stereo vision system for generic obstacle and lane detection”. IEEE Trans. Image Processing. Vol. 7. pp.62 - 81. 1998.

[3] K. Kaliyaperumal, S. Lakshmanan, K. Kluge. “An algorithm for detecting roads and obstacles in radar images”. IEEE Trans. Veh. Technology. Vol. 50. No. 1. pp.170-182. 2001.

[4] Y. Wang, E. K. Teoh, D. Shen. “Lane detection and tracking using B-snake.” Image Vis. Computing. Vol. 22. No. 4. pp.269-280. 2004.

[5] Z. Kim. “Robust lane detection and tracking in challenging scenarios”. IEEE Trans. on Intell. Transp. System. Vol. 8. No. 1. pp. 16-26. 2008.

[6] C. Lipski, B. Scholz, K. Berger, C. Linz, T. Stich. “A fast and robust approach to lane marking detection and lane tracking”. Proc. IEEE Southwest Symp. on Image Analy. and Interpretation. USA. pp.57-60. 2008.

[7] Q. Li, N. Zheng, and H. Cheng. “Springrobot: A prototype autonomous vehicle and its algorithms for lane detection”. IEEE Trans. Intell. Transp. System. Vol. 5. No. 4. pp. 300-308. 2004.

[8] C. R. Jung and C. R. Kelber. “Lane following and lane departure

using a linear-parabolic model". Image Vis. Computing. Vol.23. No.13. pp.1192-1202. 2005.

[9] M. Aly. "Real time detection of lane markers in urban streets". Proc. IEEE Intelli. Vehicles Symposium. pp. 7-12. 2008.

[10] M. A. Mallot, H. H. B Hulthoff, J. J. Little, and S. Bohrer. "Inverse perspective mapping simplifies optical flow computation and obstacle detection". Biological Cybernetics. Vol. 64. pp. 177 - 185. 1991.

[11] A. M. Muad, A. Hussain, S. A. Samad, M. M. Mustaffa, and B. Y. Majlis. "Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system". Proc. IEEE Conf. TENCON. Vol. 1. pp.207 - 210. 2004.

[12] P. Viola and M. J. Jones. "Robust real-time face detection". International J. of Computer Vision. Vol. 57. No. 2. pp.137-154. 2004.

[13] O.Ghita and P.Whelan. "Computational approach for edge-linking". J. Electron. Imaging. Vol. 11. No. 4. pp.479-485. 2002.

[14] J.Stahl and S. Wang. "Edge grouping combining boundary and region information". IEEE Trans. Image Processing. Vol. 16. No. 10. pp. 2590-2606. 2007.

[15] J. Yu, Y. Han, H. Hahn. "A scheme of extracting forward vehicle area using the acquired lane and road area information". Journal of Korean Institute of Intell. System. Vol. 18. No. 6. pp. 797-807. 2008.

[16] J. Park , J. Lee and K. Jhang. "A lane-curve detection based on an LCF". Pattern Recognit. Letter. Vol. 24. No. 14. pp.2301-2313. 2003.