Intelligent road control system using advanced image processing techniques

Dingxin Ouyang
The University of Toledo

Follow this and additional works at: http://utdr.utoledo.edu/theses-dissertations

Recommended Citation

This Thesis is brought to you for free and open access by The University of Toledo Digital Repository. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of The University of Toledo Digital Repository. For more information, please see the repository's About page.
A Thesis
entitled

Intelligent Road Control System Using Advanced Image Processing Techniques

by

Dingxin Ouyang

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the

Master of Science Degree in Electrical Engineering

Dr. Ezzatollah Salari, Committee Chair

Dr. Junghwan Kim, Committee Member

Dr. Jackson Carvalho, Committee Member

Dr. Patricia R. Komuniecki, Dean
College of Graduate Studies

The University of Toledo

December 2012
Copyright 2012, Dingxin Ouyang

This document is copyrighted material. Under copyright law, no parts of this document may be reproduced without the expressed permission of the author.
An Abstract of

Intelligent Road Control System Using Advanced Image Processing Techniques

By

Dingxin Ouyang

Submitted to the Graduate Faculty as partial fulfillment of the requirements for the Master of Science Degree in Electrical Engineering

The University of Toledo
December 2012

Over the past few years, Support Vector Machine (SVM) has been widely used in data classification field and has already been proved as an optimal solution for both linear and nonlinear classification problems. Since the image segmentation can be considered as a type of classification, SVM can be designed as an efficient image segmentation tool. This thesis aims to develop a SVM based intelligent road transportation control system, which involved three modules: pavement inspection, vehicle tracking, and collision warning. In the pavement inspection part, the SVM is used to extract the pavement from the background in a given image. The Radon transform is then applied to the pure pavement image to classify the crack to a particular type. In the vehicle tracking part, SVM trained by Gabor and edge features is involved to segment the first frame of a given video, which captured by an in-car camera. Another Wavelet feature based SVM is utilized to tracking this specific vehicle. In the collision warning part, the Time to Collision (TTC) is calculated by the scale change method. By the comparison between the TTC and a predefined threshold value, the Forward Collision Warning (FCW) system is designed, which can inform the driver to push the brake to avoid crash. Although the
traditional image processing methods can fulfill all the three tasks above, limited success has been accomplished due to the low accuracy of image segment result. The proposed SVM algorithm can be trained by the proper feature, such as RGB feature, Gabor feature, Wavelet feature, etc., which makes the system appear to be more effective and computationally more efficient.
Acknowledgements

I sincerely thank my advisor Dr. Ezzatollah Salari for giving me the opportunity to pursue my research under his guidance. It has been a great learning experience working with him. I would also like to express my deepest gratitude towards him for offering instructive advice at all times, treating me with endurance, and constantly pushing me to do better every time.

Last, but not the least, I thank my family, and friends who have been a constant source of encouragement and support for me throughout my journey in the Master’s Degree program.
# Table of Contents

Abstract .................................................................................................................................................. iii

Acknowledgements ................................................................................................................................. v

Table of Contents .................................................................................................................................. vi

List of Tables ......................................................................................................................................... viii

List of Figures ...................................................................................................................................... ix

1 Introduction ....................................................................................................................................... 1
   1.1 Background .............................................................................................................................. 1
   1.2 Motivation of the Research ................................................................................................. 2
   1.3 Organization of the Thesis ..................................................................................................... 3

2 Literature Review ................................................................................................................................. 4
   2.1 Exitng Automated Pavement Distress Detection Algorithms ............................................. 4
   2.2 Exitng Vehicle Detection and Tracking Algorithms ......................................................... 5
   2.3 Exitng Forward Collision Warning and Lane Departure Warning Algorithms ................. 6

3 Pavement Distress Detection and Classification using Support Vector Machine ................. 7
   3.1 Pavement Segmentation using a Support Vector Machine ................................................. 8
   3.2 Crack Extraction Using Fractal Theory based on Thresholding ........................................ 13
   3.3 Crack Classification using the Radon Transform ................................................................. 14

4 Real Time Vehicle Detection and Tracking Using Support Vector Machine ......................... 22
4.1 Vehicle Detection Using the Gabor Feature Based SVM ............................. 25
4.2 Vehicle Tracking Using Wavelet Feature Based SVM ............................ 27

5 Camera-based Forward Collision Warning and Lane Departure Warning Using Support Vector Machine ............................................................................................................. 30
   5.1 The Proposed Algorithm ........................................................................ 30
   5.2 Forward Collision Warning .................................................................... 31
   5.3 Lane Departure Warning ........................................................................ 35

6 Simulation Result ............................................................................................ 39
   6.1 Simulation result for the SVM based pavement Inspection System ....... 39
   6.2 Simulation result for the Vehicle Detection and Tracking System ........ 42
   6.3 Simulation result for the SVM based FCW and LDW ......................... 44

7 Conclusion and Future Work ........................................................................... 47

References ............................................................................................................ 50

Appendix A Source Code for SVM Image Segmentation Program ................ 54
Appendix B Source Code for Vehicle Tracking ............................................... 56
Appendix C Source Code for Lane Detection .................................................... 63
Appendix D Gabor Filter Bank Generator ......................................................... 67
Appendix E Source Code for Wavelet Transform ........................................... 69
List of Tables

3.1 Crack classify guideline .............................................................................................. 21
5.1 Departure decision guideline ...................................................................................... 38
6.1 SVM training features ................................................................................................. 39
List of Figures

3-1 Flowchart of the pavement inspection system.............................................................. 8
3-2 Geometric interpretation of SVM................................................................................. 9
3-3 Transversal Crack and its Radon Transform.............................................................. 17
3-4 Longitudinal Crack and its Radon Transform ............................................................ 17
3-5 Diagonal Crack at 45° and its Radon Transform........................................................ 17
3-6 Diagonal Crack at 135° and its Radon Transform...................................................... 18
3-7 Block Crack and its Radon Transform ....................................................................... 18
3-8 Alligator Crack and its Radon Transform ............................................................... 18
4-1 Flowchart of the vehicle detection and tracking......................................................... 24
4-2 Edge map of a vehicle image...................................................................................... 25
4-3 Example Gabor Filter banks .................................................................................... 26
4-4 Example of a 2-level wavelet transform..................................................................... 28
5-1 (a) Flowchart of the Forward Collision Warning system (b) Flowchart of the Lane
Departure Warning system ............................................................................................... 31
5-2 Lane detection result................................................................................................... 37
6-1 (a, b) A pavement image and segmentation result..................................................... 40
6-2 (a, b, c, d) Results for the pavement inspection......................................................... 41
6-3 Vehicle detection result ............................................................................................. 42
6-4 40th frame tracking result........................................................................................ 43
6-5 80th frame tracking result................................................................. 43

6-6 (a) 40th frame of the video. (b) 60th frame of the video. (c) 80th frame of the video. (d) 120th frame of the video. ................................................................. 45

6-7 Lane Departure Warning result ................................................................. 46
Chapter 1

Introduction

1.1 Background

The road system plays a crucial role in our complex world today as a result of its major impact on the society and economy. A good quality road transportation system is a necessity for every human activity. However, no matter how well the road is constructed, it will eventually deteriorate due to stress. The traditional road inspection procedure visually inspects pavement segments and evaluates them subjectively by human experts. This process of human inspection and classifying based on samples and experience involves a great deal of labor costs. For this reason, there is an urgent need for an automatic procedure that can be used for inspection of the road and detections of the cracks on the pavement. On the other hand, vehicle detection and tracking is a primary problem in Intelligent Transportation System (ITS), which has been developed to make the existing traffic infrastructure more efficient and assist the driver to avoid the traffic accident. Over 10 million people are injured yearly worldwide in road accidents. These include two to three million severely injured and 400,000 fatalities. The financial damage of accidents is estimates as 1%-3% of world GDP. Rear-end and lateral collisions constitute a significant proportion of the total accidents.
1.2 Motivation of the Research

This documentation proposed a Support Vector Machine (SVM) based road control system which involved three modules, such as pavement inspection system, vehicle detection and tracking system, Forward Collision Warning (FCW) and Lane Departure Warning (LDW) system. Traditional radar or laser based pavement inspection and vehicle detection systems have been researched for a long time. However, due to the noisy pavement surfaces and the complex vehicle video, limited success was accomplished. For the pavement inspection, although the existing methods can detect the crack and classify it, all these kind of methods suffer from the background, which is surrounding the pure pavement. Thus, in our algorithm, SVM as a pavement extraction tool has been applied before the crack classification. For the vehicle detection and tracking, Gabor feature and Wavelet feature based SVMs were applied for detection and tracking, respectively. The proposed algorithm using the video capture by an in-car camera as an input, SVM is applied to extract vehicle from background. For the FCW and LDW, scale change method was applied to calculate the Time to Collision without knowing the distance or the velocity information. Thus, the trigger of FCW can be designed through the comparison result between TTC and a predefine threshold value. The LDW system takes the advantage that lane is constituted by several line components, which can be easily detected by Hough transform. The warning system will be activated when the host vehicle is crossing the line without the use of blinkers.
1.3 Organization of the Thesis

The primary goal of this thesis is to develop a SVM based intelligent road transportation control system, which involved three modules: pavement inspection, vehicle tracking and collision warning. The rest of the thesis is organized as follows:

In Chapter 2, a brief review of some of the commonly used pavement inspection, vehicle detection, vehicle tracking, Forward Collision Warning, and Lane Departure Warning methods are presented.

In Chapter 3, investigations are made to develop a SVM based pavement inspection system.

In Chapter 4, a SVM based vehicle detection and tracking algorithm is proposed.

In Chapter 5, a SVM based Forward Collision Warning and a Hough transform based Lane Departure Warning systems were presented respectively.

In Chapter 6, simulation results of the above algorithms are given illustrating the effectiveness of the proposed systems.

In Chapter 7, conclusions are drawn.
Chapter 2

Literature Review

During the past few years, several technologies have been developed using a variety of concepts and approaches for pavement inspection, vehicle detection, vehicle tracking, FCW and LDW.

2.1 Existing Automated Pavement Distress Detection Algorithms

There has been a significant amount of research during the past two decades in developing automated pavement inspection systems. Xu and Huang [1] developed a pavement crack inspection system in which an image is divided into small cells and a cell is then classified as either crack or non-crack seeds based on its local characteristics. A cluster of seeds is identified as an actual crack. Cheng et al. [2] described a neural network-based thresholding method to segment and classify pavement images that can be implemented in real time. Yamaguchi et al. [3] proposed the use of an improved percolation model as a technique to detect pavement cracks. Abdel-Qader et al. [4] conducted a comparison study of crack detection in bridges involving a fast Haar transform, fast Fourier transform, and Sobel and Canny operators. Similarly, Tsai et al. [5] presented a critical assessment of various segmentation algorithms for pavement distress detection and classification.
In recent years, the use of a Discrete Wavelet transform (DWT) has also been explored in pavement inspection. Specifically, Wang et al. [6] proposed a pavement distress detection algorithm based on DWT. The magnitude of the wavelet coefficients represents the level of distress. However, this kind of system suffers from a critical problem, such that it can only deal with a pavement image that does not have complicated background. In other words, if the road image includes some background components like the trees, or cars, the performance of these methods will deteriorate significantly.

2.2 Existing Vehicle Detection and Tracking Algorithms

There has been a significant amount of research in developing automated vehicle detection and tracking systems. Avidan [7] designed Support Vector Tracking system, which utilizes an offline-learned support vector machine as the classifier and embeds it into an optical-flow based tracker. Broggi et al. [8] developed a multi-resolution vehicle detection system which uses the symmetry features to locate the vehicle. Bertozzi et al. [9] described a stereo vision-based vehicle detection system that can be implemented on the ARGO vehicle. Zhu et al. [10] defined an object tracking system using SVM regression method. Rasekhi et al. [11] developed a supervised learning method based on wavelet transform to perform the airplane tracking.

Lim et al. [12] demonstrated monocular lane-vehicle detection and tracking system comprising of lane boundary detection, lane region tracking and vehicle detection with vertical asymmetry measurement. A critical survey of recent vision-based on-road vehicle detection system was presented by Sun et al. [13]. Song et al. [14] presented a monocular machine vision system capable of detecting vehicles in front or behind of a vehicle. Bensrhair et al. [15] presented a comparison of the model vehicle detection
system and the stereo vision based vehicle detection system. Zhang et al. [16] proposed a framework to analyze the traffic video sequence using unsupervised vehicle detection method.

2.3. Existing Forward Collision Warning and Lane Departure Warning Algorithms

Marius M. Balas [17] presented a new method of management and optimization for the highway traffic, based on the Constant Time to Collision Criterion. Sumio Motoyama et al. [18] developed a Lane Departure Warning System for a series production car. The system has functions to predict vehicle’s departure from the lane with a CCD camera and warn the driver with visual and sound messages, plus steering wheel vibration. Eric Raphael et al. [19] discussed the development of a camera-based FCA system that uses a camera in place of a radar device for sensing rear-end crash situations and provided an overview of the system, including how the system detects vehicles, tracks vehicles, projects collision course trajectories, and estimates Time to Collision (TTC) using image scale change. Said Mammar [20] developed a Distance to Line Crossing (DLC) based computation of time to line crossing with different computation methods. K. Lee et al. [21] demonstrated a Collision warning / collision avoidance evaluation process based on a performance metric which is commonly used in signal detection and information retrieval under unbalanced data population.
Chapter 3

Pavement Distress Detection and Classification using Support Vector Machine

In this chapter, we propose a new pavement detection algorithm based on the Support Vector Machine (SVM), which is a tool with a set of related supervised learning methods that analyze data and recognize patterns, used for classification and regression analysis. The proposed method consists of three modules: pavement segmentation, crack extraction, and crack classification. A Support Vector Machine (SVM) is used to separate a pavement surface from its complicated background. A fractal based thresholding technique is applied to extract the pavement cracks. Finally, a Radon transform is introduced to analyze the linear structure and classify the image into various crack types. The overall scheme is summarized in the following diagram.
3.1. Pavement Segmentation using a Support Vector Machine

The Support Vector Machine (SVM) is a popular data classification technique, which was proposed by Vapnik and his group at AT&T BELL Laboratories. Generally speaking, the SVM is a supervised learning method that can analyze data and recognize patterns. It has been widely used in human face recognition, handwriting recognition, vehicle license plate recognition and can outperform other competing methods in most cases [22].
The main concept of the SVM is to construct a hyper-plane or a set of hyper-planes in a high or infinite dimensional space, and use them to classify the data. Among the possible hyper-planes, SVMs select the one where the distance of the hyper-plane from the closest data point is as large as possible. The distance between this data point and the hyper-plane is known as the margin.

Figure 3-2 shows the geometric interpretation of the SVM, the figure on the left presents a large margin whereas the image on the right displays a small margin. As a result, the hyper-plane on the left is more desirable than the one on the right for the purposes of classification.

The challenge in SVM classification is to find a hyper-plane with a maximal margin. Let's define a training sample \( D_i = (x_i, y_i) \), with the input data \( x_i \in \mathbb{R}^N \) and the corresponding target \( y_i \in \{-1, +1\} \), and construct a hyper-plane classifier,

\[
g(x) = w \cdot x - b
\]  

(3.1)

Where \( w \) is the weight vector and \( b \) is the bias. Note, \( g(x_i) \geq 1 \) if input vector \( x_i \) is in class 1, otherwise, \( g(x_i) \leq -1 \) if \( x_i \) is in class -1. If the training data are linearly separable, we can select two hyper-planes in a way that there are no points between them.
and then try to maximize their distance. The distance between these two hyper-planes can easily be obtained as \( 2/\|\mathbf{w}\| \). As a result, minimizing the value of \( \|\mathbf{w}\| \) becomes the primary goal. Hence, the problem becomes a quadratic programming optimization problem as follows:

\[
\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t.} \quad y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1
\]  

(3.2)

The solution to this problem is given by the saddle point of the Lagrange function:

\[
L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \{y_i[\mathbf{w} \cdot \mathbf{x}_i - b] - 1\}
\]  

(3.3)

where \( \alpha_i \) are the Lagrange multipliers. Classical Lagrangian duality enables the primal problem to be transformed to its dual form, which is easier to solve. The dual form is given by:

\[
\max_{\alpha} \bar{L}(\alpha) = \max \left\{ \min_{\mathbf{w}, b} L(\mathbf{w}, b, \alpha) \right\}
\]  

(3.4)

The solution to the dual problem is given by:

\[
\bar{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j
\]  

(3.5)

\[
= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i, \mathbf{x}_j)
\]

\[
\text{s.t. } \alpha_i \geq 0, \quad \sum_{i}^{n} \alpha_i y_i = 0
\]

where a linear kernel is used, i.e., \( K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j \) and the optimal weight vector is obtained as,

\[
\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i
\]  

(3.6)
Therefore, the decision function will be

\[ g(x) = \text{sign} \left( \sum \alpha_i y_i x_i \cdot x + b \right) \]  

(3.7)

There are other kinds of kernels as listed below which can be used in SVM to solve various problems:

Polynomial kernel: \( K(x, y) = [(x \cdot y) + 1]d \)

Radial basis kernel: \( K(x, y) = \exp(-|x - y|^2 / d^2) \)

Sigmoid kernel: \( K(x, y) = \tanh(a(x \cdot y) + b) \)

In addition to linear classification, a SVM can be applied to nonlinear classification problems. When applying a SVM in non-linear problems, nonlinear mapping is used to map the data into a high-dimensional feature space, where the data can be linearly classified [23]. To implement the SVM for pavement segmentation, both the color as well as the textural information are used. The color information is specified by RGB values and the texture is obtained by the application of the Gabor filter as explained in the following.

The use of SVM for pavement segmentation requires an initial training phase for the estimation of the optimal parameters. The training phase involves the extraction of the features both from the background and the pavement. For the proposed approach, we have chosen two kinds of features: the color feature and the texture feature.

The RGB value is one of the dominant factors in the color image. The RGB color model is an additive color model in which red, green, and blue lights are added together in various ways to reproduce a broad array of colors. Each point has a set of values called the RGB value which is a number between 0 and 255. As a result, the RGB value has been used as the color feature.
The texture feature is another crucial factor in the pavement image segmentation. For texture feature extraction, we apply a Gabor filter to the image [24]. In the spatial domain, a 2D Gabor is a special form of the Gaussian kernel function modulated by a sinusoidal plane wave. It is defined as follows:

$$g(x, y) = h(x, y) \exp(2\pi j W x)$$

$$= \frac{1}{2\pi \sigma_x \sigma_y} \exp \left(0.5 \left( \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right) \right) \exp (2\pi j W x)$$

where $\sigma_x^2$ and $\sigma_y^2$ are the variances in the $x$ and $y$ directions, and $W$ is the modulation frequency [25].

To extract the texture feature, the image is first transformed from the RGB color space to the Ycbcr color space. In this specific color space, the Y value represents the luminance component and the $C_b$, $C_r$ are the chrominance components of the image. The Gabor filter is applied to the Y component in four dominant orientation values, 30, 60, 90, 120 degrees, and its mean value is used as a measure of the texture.

To train the SVM classifier, we have taken 20 sample points from the background and 20 sample points from the pavement, a total of 40 points to form the training sample. Each feature vector is organized with four components: the RGB values along with output of the Gabor filter representing the texture. As a result, the dataset is a 4*40 matrix. The trained support vector machine is then used to classify the original color image into two groups: the background group and the pavement group. The pavement segment is further processed for crack extraction.
3.2 Crack Extraction Using Fractal Theory based on Thresholding

Thresholding is one of the most common methods of image segmentation. During this process, individual pixels in original images are labeled as object pixels if their value is greater than some given threshold value, and labeled as background pixels if their value is less than the threshold value.

In the proposed approach, we use fractal theory based a thresholding method. The crack on the pavement can be considered to be several single cracks which show self-similarity. This kind of self-similarity is a central concept of fractal geometry, and it can make the cracks effectively represented by fractals and thus pavement crack images can be segmented using a fractal method. We adopt the thresholding scheme as introduced in reference [26] and briefly described as follows. An important aspect of fractal objects is their surface area. Considering all points at distance $r$ from the grey level surface, the area of a fractal surface $A(r)$ is obtained as,

$$A(r) = \frac{V(r)}{2r} \quad (3.9)$$

Where $V(r)$ is the volume corresponding to the surface thickness $2r$. To find $V(r)$, first, the concept of upper surface $u(i,j,r)$ and lower surface $b(i,j,r)$ are introduced. Given the gray level function $f(i,j) = u(i,j,0) = b(i,j,0)$, the upper and the lower surfaces for various values of $r$ are updated as follows:

$$u(i,j,r + 1) = \max \left\{ u(i,j,r) + 1, \max_{|(m,n) - (i,j)| \leq 1} [u(m,n,r)] \right\} \quad (3.10)$$

$$b(i,j,r + 1) = \max \left\{ b(i,j,r) + 1, \min_{|(m,n) - (i,j)| \leq 1} [b(m,n,r)] \right\} \quad (3.11)$$

where the point $(m,n)$ is an immediate neighbor of $(i,j)$. The volume is then computed from $u$ and $b$ functions by,
The area of a fractal surface behaves according to the expression:

$$A(r) = kr^D$$  \hspace{1cm} (3.13)

Taking the logarithm of the equation above, we get:

$$\log A(r) = D \log r + \log k$$  \hspace{1cm} (3.14)

In the above equation, the value of $D$ and $k$ can be calculated by least-square fitting. The parameter $k$ reflects the change of the surface area on different scales and can be used as a local threshold to segment the image.

### 3.3 Crack Classification using the Radon Transform

Radon transform is one of the most efficient methods to detect the line in a given image, which can be used to analyze the crack image and accomplish the classification. Applying the Radon transform on an image $f(x,y)$ for a given set of angles can be considered as computing the projection of the image along the given angles. The resulting projection is the sum of the intensities of the pixels in each direction, for example a line integral. The result is a new image $R(\rho, \theta)$ [27].

Radon transform has proven to be a useful tool for extracting linear features in an image. In this method, the input image $g(x,y)$ with linear structures is transformed into a new space $R(\rho,\theta)$ of the line parameters. A common definition of the Radon transform is expressed as the following line integral,

$$R(\rho,\theta) = \int \int g(x,y) \delta(\rho - x \cos \theta - y \sin \theta) dx dy$$  \hspace{1cm} (3.15)
Where $\rho$ and $\theta$ represents the parameters of the line as described in the following equation and is the Dirac delta function.

$$\rho = x\cos\theta + y\sin\theta$$  \hspace{1cm} (3.16)

All lines in the normal Radon transform can be described by choosing that $0 \leq \theta \leq 2\pi$ and $\rho \geq 0$, but often other limits are used. If negative values of $\rho$ are introduced, the parameter domain is bounded by $0 \leq \theta \leq \pi$ and $-\rho_{\text{max}} \leq \rho \leq \rho_{\text{max}}$, where $\rho_{\text{max}}$ is positive and finite when any discrete implementation is considered. Both limits of the parameter domain are valid, and they can be achieved by the following conversion:

$$R(\rho, \theta) = R(-\rho, \theta + \pi)$$  \hspace{1cm} (3.17)

The proposed approach can detect and classify five types of crack images, such as a transversal crack, longitudinal crack, diagonal crack, block crack, and alligator crack. In order to classify the crack image into a specific type of crack, we apply a linear Radon transform to the binary image obtained from the fractal thresholding, and analyze the results to find its relationship with the crack image.

To better understand the relationship of the Radon transform result and the crack image, we apply the Radon transform to the six sample crack images and analyze the results. The set of figures below show this procedure. The left side of the figures are the original sample cracks, and the right side show their corresponding Radon transform images. The angle of a crack is defined as the angle between the direction of the crack and the lateral direction of the pavement.

Figure 3k3 is a transversal crack, the transform image has a peak at $90^\circ$ and the x coordinate of the peak determines the position of the crack. Figure 3k4 indicates a longitudinal crack where the peak in the transform image is at $0^\circ$ and $180^\circ$. Figures 3-5
and 3-6 are a pair of cracks called diagonal crack, and the peaks in the Radon transform results are at 45° and 135°, respectively. For block and alligator cracks, the number of peaks increases to at least four. Block cracks form two groups of peaks closely located at 0° and 90°, respectively. A peak array is defined as having at least two peaks at a certain angle. For the block cracks shown in Figure 3-7, there should be one peak array at 0° and another one at 90°. For the alligator cracks, there are two peak arrays at 50° and at 130°, as shown in Figure 3-8. In summary, if there are two or three peaks, the cracks are the combined single cracks of the longitudinal, transverse, or diagonal cracks. If there are four or more peaks, one needs to determine their patterns for the existence of a block or alligator crack.
Figure 3-3 Transversal Crack and its Radon Transform

Figure 3-4 Longitudinal Crack and its Radon Transform

Figure 3-5 Diagonal Crack at 45° and its Radon Transform
Figure 3-6 Diagonal Crack at 135° and its Radon Transform

Figure 3-7 Block Crack and its Radon Transform

Figure 3-8 Alligator Crack and its Radon Transform
Generally, the number of peaks in the Radon domain corresponds to the number of cracks, so this number can be used to determine if the crack is a single crack or multiple cracks. The angle of the projection, $\theta$, is perpendicular to the orientation of a crack. Therefore, $\theta$ is used to classify a single crack as a longitudinal, transversal, or diagonal crack. The coordinate $x$ is the shortest distance between the center and the crack and it is used to locate the position of a crack. Since a Radon transform integrates the crack along its dominant orientation, the peak value has a strong relationship with the length of a crack. The classification algorithm is explained in detail as follows:

First, we define a threshold to determine the peaks in the Radon transform. In this case, we select the threshold value by setting a peaks search range between $(0.6*p)$ and $(p)$. Here, $p$ represents the Radon transform maximum value. All peaks in a small window are grouped to form a single peak. The process continues until all of the peaks are found and grouped together in a window. The procedure for crack classification is summarized as follows:

1). If the number of windows is equal to one, there is a single crack. If $1^\circ < \theta < 10^\circ$ or $145^\circ < \theta < 180^\circ$, this may indicate the presence of a longitudinal crack; else if $35^\circ < \theta < 80^\circ$, this may indicate the presence of an obtuse angle diagonal; similarly, if $80^\circ < \theta < 100^\circ$, this may indicate the presence of transversal cracks, otherwise it may indicate the presence of an acute angle diagonal crack.

2). If the number of windows is equal to two or three, there are two or three cracks, classify each of them according to Step 1.
3). If the number of windows is equal to four and two of the windows are located between $(1^\circ, 10^\circ)$ and $(145^\circ, 180^\circ)$ and the other two between $80^\circ$ and $100^\circ$, the cracks are block cracks. Otherwise, return to Step 1 to classify them as four single cracks.

4). If the number of windows is equal to or greater than five, they are more likely to be block cracks or alligator cracks. Go to Step 3 to determine whether they are block cracks. If there are two arrays of windows not located in the ranges of $(1^\circ, 10^\circ)$ and $(145^\circ, 180^\circ)$ or $(80^\circ, 100^\circ)$, they are considered to be alligator cracks.

5). If the number of windows is larger than 10, they are most likely alligator cracks.

The crack type classify rule is shown in Table 3.1
Table 3.1 Crack classify guideline

<table>
<thead>
<tr>
<th>Crack type</th>
<th>Number of windows</th>
<th>Angle of the projection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longitudinal crack</td>
<td>1</td>
<td>$1^\circ &lt; \theta &lt; 10^\circ$</td>
</tr>
<tr>
<td>Longitudinal crack</td>
<td>1</td>
<td>$145^\circ &lt; \theta &lt; 180^\circ$</td>
</tr>
<tr>
<td>Obtuse angle diagonal crack</td>
<td>1</td>
<td>$35^\circ &lt; \theta &lt; 80^\circ$</td>
</tr>
<tr>
<td>Acute angle diagonal crack</td>
<td>1</td>
<td>$11^\circ &lt; \theta &lt; 34^\circ$ or $101^\circ &lt; \theta &lt; 144^\circ$</td>
</tr>
<tr>
<td>Combination of single cracks</td>
<td>2</td>
<td>$1^\circ &lt; \theta &lt; 180^\circ$</td>
</tr>
<tr>
<td>Combination of single cracks</td>
<td>3</td>
<td>$1^\circ &lt; \theta &lt; 180^\circ$</td>
</tr>
<tr>
<td>Block cracks</td>
<td>4</td>
<td>2 of them between $(1^\circ, 10^\circ)$ and $(145^\circ, 180^\circ)$, 2 of them between $80^\circ$ and $100^\circ$</td>
</tr>
<tr>
<td>Combination of single cracks</td>
<td>4</td>
<td>Otherwise</td>
</tr>
<tr>
<td>Alligator cracks</td>
<td>5 or more than 5</td>
<td>Not in $(1^\circ, 10^\circ)$ and $(145^\circ, 180^\circ)$ or $(80^\circ, 100^\circ)$</td>
</tr>
<tr>
<td>Block cracks</td>
<td>5 or more than 5</td>
<td>Otherwise</td>
</tr>
<tr>
<td>Alligator cracks</td>
<td>More than 10</td>
<td>$1^\circ &lt; \theta &lt; 180^\circ$</td>
</tr>
</tbody>
</table>
Chapter 4

Real Time Vehicle Detection and Tracking Using Support Vector Machine

In this chapter, we present a vehicle detection and tracking system which is capable of identifying vehicles ahead and tracking them continuously with an in-car video camera. The fundamental problem here is to identify vehicles in changing environment and illumination. Although there have been numerous publications on general object recognition and tracking, or combination of them, not many of these techniques could successfully be applied in real-time to in-car video. The proposed method uses one of the most efficient classifiers, SVM to perform the vehicle detection and tracking in real time. In this case, we can train a classifier in advance to distinguish between a desired object and the background. Our system detects and tracks the vehicles from a video sequence taken by a forward looking camera mounted on a moving vehicle. The problem is then converted to an image segmentation problem in which the input video is analyzed frame by frame and the location of the tracking vehicle is found in each frame.

The proposed method can be divided into two modules: the detection and the tracking parts. For a given video, we can find all the vehicles in the first frame by using a SVM classifier trained by Gabor features as well as edge features, and assign a different label to each detected object. The Gabor filter is used to extract the features of vehicle and
non-vehicle images, and the SVM is trained with these two classes of input images. This method can deal with the variability problem including various vehicle models, the different weathers, and the different road conditions. Since the SVM is trained with a large amount of input data including all the variations, it can be used in multiple detection tasks.

In the tracking of a specific vehicle, a different SVM with reduced complexity is trained by wavelet features extracted from the first frame. Wavelet features provide a more robust and efficient system. The experimental results show that the proposed method acquires not only the accuracy needed for the detection and tracking, but can also be implemented more efficiently. The following diagram provides a summary of the proposed method.
Figure 4-1 Flowchart of the vehicle detection and tracking system
4.1 Vehicle Detection Using the Gabor Feature Based SVM

To train a two-class SVM classifier, we use two types of features, the edge feature as well as features extracted from the Gabor filter. We take advantage of the fact that vehicles are manmade objects that contain strong horizontal and vertical edges. As a result, the region with strong edge factors is more likely to indicate the presence of a vehicle in a given image. Therefore, the original vehicle and background training images are converted to an edge map image to obtain the first training set feature. Figure 4-2 shows the original and the result of edge map images for a vehicle.

![Figure 4-2 Edge map of a vehicle image](image)

To extract the second training set feature, a Gabor filter is applied. In the spatial domain, a 2D Gabor is a special form of the Gaussian kernel function modulated by a sinusoidal plane wave. It is defined as follows:

\[
h(x, y) = \exp \left\{ -\frac{1}{2} \left[ \frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} \right] \right\} \cos(2\pi u_0 x + \varphi) \tag{4.1}
\]

Where \( u_0 \) and \( \varphi \) are the frequency and phase of the sinusoidal plane wave along the x-axis, and \( \sigma_x^2 \) and \( \sigma_y^2 \) are the variances in the x and y directions. Figure 4-3 is an example of Gabor filter banks at various orientations, 0, 60, 120, 180 degrees.
Given an input image $I(x, y)$, Gabor feature extraction is performed by convolving $I(x, y)$ with a Gabor filter bank. Although the raw responses of the Gabor filter could be used directly as features, some kind of post-processing is usually applied, for example, Gabor-energy features, thresholded Gabor features, and moments of the Gabor features. In this case, we use the moment of the Gabor features, extracted from several windows of the input image. The inputs to the feature extraction sub-system are the hypothesized vehicle sub-images. First, each sub-image is scaled to a fixed size of 64 * 64 pixels. Then, it is subdivided into 9 overlapping 32 * 32 windows. The Gabor filters are then applied to each window separately. The motivation for extracting possibly redundant Gabor features from several overlapping windows is to compensate for errors in the hypothesis generation step (e.g., sub-images containing partially extracted vehicles or background information), making feature extraction more robust [28].
Both the edge feature and the Gabor feature provide the input data to train an SVM for the vehicle detection. It should be mentioned that Gabor features are relatively robust and have been successfully used in the past for face recognition and image retrieval. Similarly, vehicles do contain strong edges and lines at different orientation, thus, the edge feature could be very powerful for vehicle verification.

4.2 Vehicle Tracking Using Wavelet Feature Based SVM

Unlike the Fourier transform, in which basis functions are sinusoids and redundant, the wavelet transforms are based on short-duration waves, called wavelets, of different frequency and restricted duration. This characteristic makes them a favorable choice to provide us with the frequency as well as temporal information for a given signal. They can easily be implemented using multi-resolution techniques. The advantage of this approach is that some features which might not be detected at one resolution may be found at some other resolutions. The 2-D Fast Wavelet Transform (FWT) result in four sub-band images in each level, where LL, LH, HL, HH denote the approximation, vertical detail, horizontal detail, and diagonal detail sub-band images, respectively. The size of each of the four sub-band images is half of the original input image [29].

To extract the feature of vehicle and non-vehicle in a given image, the input image is decomposed into various sub-band images obtained by a 2-level Haar Wavelet Transform (WT). By decomposing an image using WT, the resolutions of the image are reduced. Because the computational complexity is drastically reduced by operation on a lower resolution than a high resolution, this algorithm is considerably more efficient than using the original image. Figure 4-4 is an example of a 2-level wavelet transform applied to a vehicle image.
It is obvious that the detection step provides the segmentation of the first frame into vehicle and non-vehicle parts. We then apply a 2-level wavelet transform to the vehicle part of the image. In the one level decomposition, the $LL_1$ sub-band corresponds to the significant visual characteristics of the vehicles; the $LH_1$ sub-band corresponds to the vertical features, such as the sides of the vehicles; the $HL_1$ sub-band corresponds to the horizontal features, such as the roof, underside, top of the grille and bumper area, and the $HH_1$ sub-band corresponds to the diagonal features, such as the corners of the vehicles' body. The decomposition can be further carried out on the $LL_1$ sub-band image, which results the second level wavelet transform, namely, the $LL_2$, the $LH_2$, the $HL_2$, and the $HH_2$ sub-band images.

In the proposed method, the SVM training sample is formed by two different features obtained within the second level wavelet transform domain. First one is the $LL_2$ sub-band, which contains most visual information of a given image. And the second one is the combination of other three second level sub-bands, which represent the edge features.
of an image. Meanwhile, the same procedure, i.e., a 2-level wavelet decomposition scheme is used to obtain the features for the non-vehicle part. Since each image is processed sequentially in the video, the search area can shrink to a relatively small area. Through the moving speed of the tracking object and the number of Frame Per Second (FPS) of the video, it is possible to calculate the longest distance that a tracking vehicle can move within each frame. As a result, a predefined search area surrounding the tracking vehicle is utilized to reduce the complexity of the vehicle tracking algorithm.
Chapter 5

Camera-based Forward Collision Warning and Lane Departure Warning Using Support Vector Machine

In this chapter, we propose a unique method that can fulfill both the FCW and LDW, only based on the video captured by an in-car camera. First, Support Vector Machine (SVM) is used to segment the first frame of a given video and locate the position of the vehicle in front of the host vehicle. Following this step, two warning systems are designed separately. For the FCW, Time to Collision (TTC), the most crucial factor of the decision, can be calculated through scale change without the information of the velocity or the distance. The warning system will be activated when the TTC is less than a predefined threshold value. For the LDW, the image processing method is used to analyze the lane position information using Hough transform and the LDW is activated if the vehicle-lane distance is exceeded the predefined threshold value and the blinkers had not been used.

5.1 The Proposed Algorithm

The proposed method involved two separate modules, FCW and LDW. For a given video, the position of vehicle in front of the host vehicle can be obtained by SVM in the first frame. The FCW calculate the TTC by scale change method without knowing any other information (velocity and distance), and then use TTC to trigger the warning. On
the other hand, we apply Hough transform to the first frame of the given video to extract line elements, which can be connected to form a lane on the road. When the distance between vehicle and lane is exceed the predefine threshold value, the LDW will be activated. The following is the flowcharts for both FCW and LDW, respectively.

![Flowchart of the Forward Collision Warning system](a)

![Flowchart of the Lane Departure Warning system](b)

Figure 5-1 (a) Flowchart of the Forward Collision Warning system (b) Flowchart of the Lane Departure Warning system

### 5.2 Forward Collision Warning

A Forward Collision Warning is issued when the Time to Collision is lower than a certain threshold (3 seconds in this case) [30]. The calculation of TTC can be divided into
two steps. First we calculate the momentary TTC based on the constant velocity model through scale change. Following this step, we obtained the final TTC from a model considering acceleration.

We define the momentary TTC as,

\[ T_m = -\frac{Z}{V} \]  

(5.1)

Where \( Z \) is the relative distance to the target and \( V \) is the relative velocity.

Since distance and relative speed are not natural vision measures, we will show that we can represent the momentary TTC as a function of scale-change in the image in a given sampling interval \( \Delta t \).

The perspective projection model of the camera gives:

\[ w_t = \frac{fW}{Z_t} \]  

(5.2)

Where \( w_t \) is the width of the target in the image at time \( t \), \( Z_t \) is the distance to the target, \( W \) is the vehicle width, and \( f \) is the camera focal length.

We define scale-change \( S \) as the ratio between the width in the image in two consecutive frames:

\[ S = \frac{w_1}{w_0} = \frac{fW/Z_1}{fW/Z_0} = \frac{Z_0}{Z_1} \]  

(5.3)

When the time interval between \( Z_1 \) and \( Z_0 \) is small we can write:

\[ Z_1 = Z_0 + V\Delta t \]  

(5.4)

Thus,

\[ S = \frac{Z_1 - V\Delta t}{Z_1} \]  

(5.5)
Extracting $Z_1/V$ from the equation above yields:

$$T_m = \frac{Z_1}{V} = \frac{\Delta t}{1 - S}$$  \hspace{1cm} (5.6)

From this equation, $T_m$ can be found based on scale change and time information.

The problem with the momentary TTC computation is that it neglects relative acceleration between the two vehicles. Relative acceleration will occur when the target vehicle performs a sudden stop or when the host vehicle is slowing down to avoid collision. Both cases are very important in an FCW application. Not detecting the host vehicle slowing down will give many false alarms (e.g. when nearing a stop light). Using the brake signal may not be enough since many times the driver might slow down by simply taking his or her foot off from the gas pedal.

Taking into account relative acceleration, the relative distance between the two vehicle as a function of time is given by:

$$Z = Z_0 + V_0 \Delta t + \frac{1}{2} a \Delta t^2$$  \hspace{1cm} (5.7)

The actual TTC denoted simply by $T$, is the time that $Z = 0$, thus:

$$T = \frac{-V_0 + \sqrt{V_0^2 - 2Z_0 a}}{a}$$  \hspace{1cm} (5.8)

Because the distance, speed, and acceleration cannot be obtained directly from the video, scale-change needs to be used instead to compute the actual TTC. The actual TTC can be determined by $T_m$ and its derivative, $\dot{T}_m$, where both can be computed from the scale-change in the image.

The momentary TTC is given by:

$$T_m = -\frac{Z}{V}$$  \hspace{1cm} (5.9)
Thus its derivative is:

\[ \dot{T}_m = \frac{-\dot{Z} \cdot V + \dot{V} \cdot Z}{V^2} \]  

(5.10)

Since \( \dot{Z} = V \) and \( \dot{V} = a \), therefore,

\[ \dot{T}_m = \frac{-V^2 + a \cdot Z}{V^2} = -1 + \frac{a \cdot Z}{V^2} = \frac{a \cdot Z}{V^2} - 1 \]  

(5.11)

Considering the following auxiliary variable,

\[ C = \dot{T}_m + 1 = \frac{a \cdot Z}{V^2} \]  

(5.12)

Because the \( T_m \) and its derivative are values taken from the current image thus \( Z \) and \( V \) can be substituting by \( Z_0 \) and \( V_0 \), therefore,

\[ T_m = -\frac{Z_0}{V_0} \]  

(5.13)

and

\[ C = \frac{a \cdot Z_0}{V_0^2} \]  

(5.14)

Extracting \( a \) from the equation (5.14),

\[ a = C \cdot \frac{V_0^2}{Z_0} \]  

(5.15)

Substituting for \( a \) in equation (5.8),

\[ T = \frac{-V_0 + \sqrt{V_0^2 + 2C \cdot V_0^2}}{a} \]  

(5.16)

\[ = \frac{-V_0 + V_0 \sqrt{1 + 2C}}{a} \]  

(5.17)

Using equation (5.15),

\[ T = \frac{-V_0 + V_0 \sqrt{1 + 2C}}{C \cdot \frac{V_0^2}{Z_0}} \]  

(5.18)
\[
= -1 + \sqrt{1 + 2C} \\
\frac{C \cdot \frac{v_0}{z_0}}{(5.19)}
\]

Considering equation (5.13),

\[
T = \frac{-1 + \sqrt{1 + 2C}}{c - T_m} \tag{5.20}
\]

\[
= -T_m \cdot \frac{-1 + \sqrt{1 + 2C}}{C} \tag{5.21}
\]

\[
= T_m \cdot \frac{1 - \sqrt{1 + 2C}}{C} \tag{5.22}
\]

where \( C \) is a function of \( \hat{T}_m \) as in equation 18. This shows that the actual TTC can be determined only by \( T_m \) and \( \hat{T}_m \).

5.3. Lane Departure Warning

The proposed LDW algorithm involves two parts, lane detection and departure warning. For the first step, line features are extracted through Hough transform method in a given image. Following this step, all the features of interest are connected to form the lanes on the road. Finally, an angle-threshold method was used to trigger the warning system.

The function of Hough transform is identification of lines in the image. In the Hough transform, the main idea is to consider the characteristics of the straight line not as position coordinate points \((x, y)\), but instead, in terms of its parameters, such as the slope parameter \( m \) and the intercept parameter \( b \). Therefore, a line \( y = mx + b \) can be represented as a point \((b, m)\) in the parameter space [31]. However, one faces the problem that vertical lines give rise to unbounded values of the parameters \( m \) and \( b \). For computational reasons, the polar coordinates, denoted by \( r \) and \( \theta \) are commonly used in
Hough transform. The parameter $r$ represents the distance between the line and the origin, while $\theta$ is the angle of the perpendicular line relative to the coordinate axis. Using the $r$ and $\theta$ parameters, the equation of the line can be written as,

$$y = \left(-\frac{\cos \theta}{\sin \theta}\right)x + \left(\frac{r}{\sin \theta}\right)$$

Therefore it is possible to associate with each line of the image a unique pair $(r, \theta)$, $\theta \in [0, \pi)$ and $r \in R$, or if $\theta \in [0, 2\pi)$ and $r \geq 0$.

The Hough transform can be used to identify the parameter of a curve, which best fits a set of given edge points. This edge description is commonly obtained from a feature detector such as the Roberts, Sobel or Canny edge detector. These might be noisy, i.e. they may contain multiple edge fragments corresponding to a single feature. The proposed algorithm uses the Canny edge operator to produce edge points from a set of lane marks in the image. Following this step, the line Hough transform is used to identify line segments and parameters.

The result of line detection procedure may contain some other objects besides the lane marks, such as vehicle or pedestrian, which is considered to be noise and should be eliminated by the thresholding method. Morphological filters are used to connect line segments to form a lane mark. Generally speaking, there are two sets of lane marks in a given frame within a video captured by an in-car camera, which form a lane for the host to move on. Figure 5-2 demonstrates the lane detection result in a given video.
For the departure warning system, we first locate the intersection points between the two lanes and the bottom part of the image, and define them as the left intersection point and right intersection point. Following this step, we measure the distance between left intersection point and the left side of the image; meanwhile, we measure the right distance in the same way. Then we will compare these two distance values to a predefine threshold value, which is 48 pixels in this given 480*320 resolution video. If the left distance is more than this threshold value, the host vehicle will departure to the left shortly, if the right distance is more than this threshold value, the host vehicle will departure to the right, otherwise, the host vehicle will move on this lane without departure. Both left departure and right departure will trigger the warning system, which can inform the driver to adjust the steering wheel to keep the vehicle in lane. Table 5.1 describes the departure decision rules.
Table 5.1 Departure decision guideline

<table>
<thead>
<tr>
<th>Departure</th>
<th>Left distance</th>
<th>Right distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left departure</td>
<td>More than 48 pixels</td>
<td>Less than 48 pixels</td>
</tr>
<tr>
<td>Right departure</td>
<td>Less than 48 pixels</td>
<td>More than 48 pixels</td>
</tr>
<tr>
<td>No departure</td>
<td>Less than 48 pixels</td>
<td>Less than 48 pixels</td>
</tr>
</tbody>
</table>
Chapter 6

Simulation Results

The most advantage of SVM is that we can choose the proper feature to train it. Table 6.1 lists the features that we have choose for different proposes, such as pavement extraction, vehicle extraction, and vehicle tracking.

<table>
<thead>
<tr>
<th>Application</th>
<th>Number of features</th>
<th>Type of Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pavement extraction</td>
<td>2</td>
<td>Color and Gabor features</td>
</tr>
<tr>
<td>Vehicle extraction</td>
<td>2</td>
<td>Edge and Gabor features</td>
</tr>
<tr>
<td>Vehicle tracking</td>
<td>2</td>
<td>Wavelet features (LL sub band and the combination of other three sub bands)</td>
</tr>
</tbody>
</table>

6.1 Simulation results for the SVM based pavement Inspection System

The proposed algorithm has been implemented in MATLAB, and its performance and simulation results are presented in this section. Figure 6-1 (a) represents an original
image in which 20 background samples and 20 pavement sample points are selected to train the SVM. Figure 6-1 (b) shows the result of the segmentation.

Figure 6-1 (a, b) A pavement image and segmentation result

To show the results of the crack extraction and classification, a longitudinal crack is considered. Figure 6-2 (a,b,c,d) represents an original image, image after thresholding, the results of the Radon transform, and the Radon transform peak values, respectively. In this example, the peaks are at 0° and 180° which corresponds to a longitudinal crack.
To measure the performance of the proposed method, a set of 50 images with different types of cracks is chosen for the experiment. For the pavement segmentation, the successful rate is based on the percentage of misclassified background pixels in the segmented image. The successful rate of the proposed segmentation algorithm is calculated to be about 95%. However, for the crack classification, the successful rate is estimated to be slightly over 90%.
6.2 Simulation results for the SVM based Vehicle Detection and Tracking System

The proposed algorithm has been implemented in MATLAB, and its performance and simulation results are presented in this section. The input video is taken from an in-car camera with 30FPS and time duration of 10 seconds. As a result, the total video has 300 frames. An SVM classifier using Gabor and edge features is used to segment the first frame of the video in order to extract the tracking vehicle from the background. Figure 6-3 shows the segmented result, the tracking vehicle is enclosed in a rectangular box.

![Vehicle detection result](image)

Figure 6-3 Vehicle detection result

The vehicle detection procedure provides the position of the tracking vehicle in the first frame. A second SVM classifier with reduced complexity is designed to track the vehicle. The tracking step utilizes the wavelet features to analyze the video frame by frame. In each frame, the location of the tracking object is found and labeled with a
rectangular box. For example, Figure 6-4 and 6-5 represent the results of tracking the vehicle in 40th and 80th frames of the video, respectively.

Figure 6-4 40th frame tracking result

Figure 6-5 80th frame tracking result
6.3 Simulation results for the SVM based FCW and LDW

The proposed algorithm has been implemented in MATLAB, and its performance and simulation results are presented in this section. The input video we used is taken from an in-car camera, which is 30FPS, and the length of it is 10 seconds. As a result, the total video has 300 frames. In the first frame of this video, we used the Gabor and edge features based SVM to segment the image, to extract the tracking vehicle from the background. Figure 6.6 (a, b, c, d) shows the tracking results of 40th, 80th, 120th, 160th frames. The tracking vehicle was labeled with a rectangle box and the TTC is presented at the right side of the image.
The predefined threshold value used in FCW is set at 3.0 seconds. As a result, the TTC for figure 6-6 (d) is 2.9 seconds which is less than the threshold leading to the activation of FCW.

The LDW is implemented as follows. The lane position is first obtained by the connection of the detected lane marks in each frame. A distance threshold value is set as 48 pixels for this particular case. The comparison between the left/right distances and the threshold value is carried on in each frame. In figure 6-7 (a), both the left and right distance are less than the threshold value, which means that the vehicle is not departure.
On the other hand, in the figure 6-7 (b) the left departures will happen, since the left distance is 133 pixels which is more than threshold value.

Figure 6-7 Lane Departure Warning result
Chapter 7

Conclusion and Future Work

This thesis presents a SVM based intelligent road transportation control system, which involved three modules: pavement inspection, vehicle tracking and collision warning.

For the pavement inspection system, the proposed method provides a robust pavement inspection method, especially in the image that not only contains a road segment, but also includes some other complicated background component. For the SVM, we have chosen the RGB values along with the texture measure obtained by the Gabor filter to form the input training data set. The cracks are then extracted from the pavement image by converting the image into a binary image using fractal theory. For the crack classification, we apply the Radon transform to the binary crack image, and analyze the image in the Radon domain to classify the crack to a specific crack type. Future research will investigate the use of other features in the color space and texture factor values extracted by a Law’s filter set for the training of the SVM. Further, the crack severity measurement can be obtained from a Radon transform, for example, the wider the crack, the larger the peak.
For the vehicle detection and tracking system, we presented a unique and effective method for vehicle detection and tracking using SVM classifier. This method can analyze a video obtained from an in-car camera, extract the vehicle in the first frame, and track it in the subsequent frames. Selection of an effective set of features for SVM is essential for its success. In our algorithm, we have used two different SVMs. The one used for vehicle detection, and the other used for vehicle tracking. In the detection part, the Gabor and the edge features are used as the input for SVM. It should be mentioned that vehicles do contain strong edges and lines at different orientations and scales, therefore, it can be effectively extracted by the Gabor filter and represented by the edge map. In the tracking part, a less complex SVM classifier based on wavelet features is designed to find the position of the vehicle. The SVM used in the tracking phase considers a specific vehicle to track, therefore, the use of wavelet features appear to be more effective and computationally more efficient. Future research will investigate the use of wavelet feature based SVM to track multiple vehicles in a given video. We can train a set of SVMs and design a parallel processing approach to fulfill this task.

For the collision warning system, we developed a camera based driving warning system which consists of two separate modules, FCW and LDW. This system can analyze the video captured through an in-car camera and warn the driver when the host vehicle is too close to the front vehicle or it is departure without intention. For the FCW modules, the SVM is applied to detect the vehicle position in a given video. The scale change method is further utilized to calculate the TTC, which is a crucial factor to trigger the warning system. For the LDW, the lane position is obtained by Hough transform, and the distance threshold method is designed to trigger the warning system. Future research
will investigate the use of feature based SVM to track multiple vehicles in a given video,
which can be achieved by train a set of SVMs to form a parallel paradigm.
References


tic;
close all;
clear all;
clc;
format compact;

%%

pic = imread('111.jpg');
figure;
imshow(pic);

%% Define the training sample

TrainData_background = zeros(20,3,'double');
TrainData_foreground = zeros(20,3,'double');
% Get background samples

msgbox('Please get 20 background samples','Background Samples','help');

pause;

for run = 1:20
    [x,y] = ginput(1);
    hold on;
    plot(x,y,'r*');
    x = uint8(x);
    y = uint8(y);
    TrainData_background(run,1) = pic(x,y,1);
    TrainData_background(run,2) = pic(x,y,2);
    TrainData_background(run,3) = pic(x,y,3);
end

% Get foreground samples

msgbox('Please get 20 foreground samples which is the part to be segmented','Foreground Samples','help');

pause;

for run = 1:20
    [x,y] = ginput(1);
    hold on;
    plot(x,y,'ro');
    x = uint8(x);
y = uint8(y);

TrainData_foreground(run,1) = pic(x,y,1);
TrainData_foreground(run,2) = pic(x,y,2);
TrainData_foreground(run,3) = pic(x,y,3);
end

% Let background be 0 & foreground 1

TrainLabel = [zeros(length(TrainData_background),1); ... ones(length(TrainData_foreground),1)];

%% Form the SVM based on LIBSVM

TrainData = [TrainData_background;TrainData_foreground];

model = svmtrain(TrainLabel, TrainData, 'kt 1 kd 1');

%% Image segmentation

preTrainLabel = svmpredict(TrainLabel, TrainData, model);
[m,n,k] = size(pic);
TestData = double(reshape(pic,m*n,k));
TestLabel = svmpredict(zeros(length(TestData),1), TestData, model);
ind = reshape([TestLabal, TestLabal, TestLabal], m, n, k);
ind = logical(ind);

pic_seg = pic;

pic_seg(~ind) = 0;

figure;

imshow(pic_seg);

figure;

subplot(1,2,1);

imshow(pic);

subplot(1,2,2);

imshow(pic_seg);
toc;
Appendix B

Source Code for Vehicle Tracking

close all;
clear all;
clc

h = [40 30]; %Define the object size
I = imread('frame0001.jpg'); R = I;
imshow(I); hold on
pt = ginput(1); y(1) = round(pt(2));
y(2) = round(pt(1));

%size(I) = 480*640*3
r = 10;
I(y(1), y(2) - r:y(2) - 3,:) = 255; I(y(1), y(2) + 3:y(2) + r,:) = 255; %Left side of the rectangle
I(y(1) - r:y(1) - 3, y(2), :) = 255; I(y(1) + 3:y(1) + r, y(2), :) = 255; %Right side of the rectangle
I(y(1), y(2), :) = 255;
I(y(1) + round(h(1)/2), y(2) - round(h(2)/2):y(2) + round(h(2)/2), :) = 255;
I(y(1) - round(h(1)/2), y(2) - round(h(2)/2):y(2) + round(h(2)/2), :) = 255;
I(y(1) - round(h(1)/2):y(1) + round(h(1)/2), y(2) - round(h(2)/2), :) = 255;
I(y(1)-round(h(1)/2):y(1)+round(h(1)/2),y(2)+round(h(2)/2),:)=255;

%Draw rectangle box

mov = avifile('Object Tracking.avi','fps',15,'quality',100);
F=im2frame(I);
mov = addframe(mov,F); %Add first frame

[Aim Aimer pic]=TemplateTrans(R,y(1),y(2),h);
x=y(2);xb=x;
y=y(1);yb=y;

%Define Search area

py=0;ty=12;
px=0;tx=12;
output(1)=100;
tic
num=1;
for Frame=1:120
    filename = sprintf('%3.3d.jpg', Frame);
    ImageName=strcat('frame0',filename);
    I =imread(ImageName);
    R=rgb2gray(I);
R = im2double(R);

[Temp detect] = Transform(R, y, x, h);

TeF = Temp.*Aim;

Out = abs(ifft2(TeF));

op = max(max(Out(1:64,1:64)));

output(Frame) = op;

[y1 x1] = find(Out == op);

y1 = y1(1) - 2;

x1 = x1(1) - 2;

if y1 > 32 & x1 > 32
    y1 = y1 - 64;
    x1 = x1 - 64;
elseif y1 > 32 & x1 < 32
    y1 = y1 - 64;
elseif y1 < 32 & x1 > 32
    x1 = x1 - 64;
end

y = y + y1;

x = x + x1;

num = num + 1;

if num > 3
    if Frame > 4
        if op < output(Frame - 2)
if op>1/2*output(Frame-2)

    [Gy,Gx]=modify(Aimer);
    y=y+Gy;
    x=x+Gx;
    Aimer=UpImage(R,y,x,h);
    [H1,L1]=size(Aimer);
    R1=round((64-H1)/2);
    R2=round((64-L1)/2);
    Amp=zeros(64,64);
    Amp(R1:R1+H1-1,R2:R2+L1-1)=Aimer;
    Aim=conj(fft2(Amp));
    num=0;

    end

end

end

end

%%%Redraw the rectangle box

I(y+round(h(1)/2),x-round(h(2)/2):x+round(h(2)/2),:)=255;
I(y-round(h(1)/2),x-round(h(2)/2):x+round(h(2)/2),:)=255;
I(y-round(h(1)/2);y+round(h(1)/2),x-round(h(2)/2),:)=255;
I(y-round(h(1)/2);y+round(h(1)/2),x+round(h(2)/2),:)=255;
I(y,x-r:x-3,:)=255;I(y,x+3:x+r,:)=255;
I(y-r:y-3,x,:)=255;I(y+3:y+r,x,:)=255;
I(y,x,:)=255;
xtp(Frame) = x;
ytp(Frame) = y;

%I=bitmapplot(ytp,xtp,I,struct('LineWidth',1,'Color',[1 0 0 1]));
imshow(I),drawnow
F = im2frame(I);
mov = addframe(mov,F);
Frame
end

t=Frame/toc

figure,plot(xtp,ytp,'-*'),title('tracking path'),grid on
clear all
Appendix C

Source Code for Lane Detection

```matlab
RGB = imread('111.jpg');
I = rgb2gray(RGB); % Convert to Grey level image
[x, y] = size(I);
BW = edge(I);
figure; imshow(I); title('Original Image')
figure; imshow(BW); title('Edge Detection Image')

rho_max = floor(sqrt(x^2 + y^2)) + 1;

accarray = zeros(rho_max, 180);

Theta = [0:pi/180:pi];

for n = 1:x,
    for m = 1:y
        if BW(n, m) == 1
```

63
for k=1:180
    rho=(m*cos(Theta(k)))+(n*sin(Theta(k)));
    rho_int=round(rho/2+rho_max/2);
    accarray(rho_int,k)=accarray(rho_int,k)+1;
end
end
end
end

%figure;colormap gray;
%imagesc(accarray);title('hough变换图像')
%xlabel('	heta'), ylabel('ho');

%accarray=uint8(accarray);
%figure;imshow(accarray);title('hough变换图像')
%xlabel('	heta'), ylabel('ho');
%axis on, axis normal, hold on;

K=1;
for rho_n=1:rho_max
    for theta_m=1:180
        if accarray(rho_n,theta_m)>=10
            case_accarray_n(K)=rho_n;
        end
    end
end
case_accarray_m(K)=theta_m;
K=K+1;
end
end
end

I_out=zeros(x,y);
I_jiao_class=zeros(x,y);
for n=1:x,
    for m=1:y
        if BW(n,m)==1
            for k=1:180
                rho=(m*cos(Theta(k)))+(n*sin(Theta(k)));
                rho_int=round(rho/2+rho_max/2);

                for a=1:K-1
                    if rho_int==case_accarray_n(a)&k==case_accarray_m(a) k==case_accarray_m(a)&rho_int==case_accarray_n(a)
                        I_out(n,m)=BW(n,m);
                        I_jiao_class(n,m)=k;
                    end
                end
            end
        end
    end
end
figure;imshow(I_out);title('ÀûÓ¶½ä»»È¡µÄͼÏñ');
[m,n]=size(I_out);
for i=1:ceil(2*m/7)
    for j=1:n
        I_out(i,j)=0;
    end
end
figure
imshow(I_out);

%=============hough transform toolbox==============%

% [H,T,R] = hough(BW,'RhoResolution',0.5,'ThetaResolution',0.5);
%
figure;imshow(H,'XData',T,'YData',R,'InitialMagnification','fit');title('hough±ä»»¾ØÕó')
% xlabel('theta'), ylabel('rho');
% axis on, axis normal, hold on;
Appendix D

Gabor Filter Bank Generator

function [filter_bank ang_s]=gabor_bank(bank_size,sigma_x,sigma_y,freq)

ang_s=fix(linspace(0,180,bank_size));
angs=deg2rad(ang_s);

filter_bank=cell(length(angs),1);
for i=1:length(angs),
    filter_bank{i}=gabor_fn(sigma_x,sigma_y,angs(i),freq);
end

function gb=gabor_fn(sigma_x,sigma_y,theta,freq)
% Sigma_x and Sigma_y must be integers
% For fingerprint enhancement, sigma_x and sigma_y should be the half of the wave
% length
% (1/(2*Freq)
sz_x=6*sigma_x+1;
sz_y=6*sigma_y+1;

[x y]=meshgrid(-fix(sz_x/2):fix(sz_x/2),fix(-sz_y/2):fix(sz_y/2));

% Rotation
x_theta=x*cos(theta)+y*sin(theta);

y_theta=-x*sin(theta)+y*cos(theta);

gb=exp(-.5*(x_theta.^2/sigma_x^2+y_theta.^2/sigma_y^2)).*cos(2*pi*freq*x_theta);
Appendix E

Source Code for Wavelet Transform

clear all;
close all;
load woman;  % Load image data

nLevel = 3;    % Number of decompositions
nColors = size(map,1);  % Number of colors in colormap
cA = cell(1,nLevel);  % Approximation coefficients
cH = cell(1,nLevel);  % Horizontal detail coefficients
cV = cell(1,nLevel);  % Vertical detail coefficients
cD = cell(1,nLevel);  % Diagonal detail coefficients
startImage = X;
for iLevel = 1:nLevel,
    [cA{iLevel},cH{iLevel},cV{iLevel},cD{iLevel}] = dwt2(startImage,'db1');
startImage = cA{iLevel};
end

figure;colormap(map);
imagesc(dwt2(startImage,'db1')); %THIS GIVES THE MAZED IMAGE INSTEAD OF THE TRANSFORMED IMAGE

figure;
tiledImage = wcodemat(cA{nLevel},nColors);
for iLevel = nLevel:-1:1,
tiledImage = [tiledImage wcodemat(cH{iLevel},nColors); ... wcodemat(cV{iLevel},nColors) wcodemat(cD{iLevel},nColors)];
end

figure;

imshow(tiledImage,map);

%reconstruct
fullRecon = cA{nLevel};
for iLevel = nLevel:-1:1,
fullRecon = idwt2(fullRecon,cH{iLevel},cV{iLevel},cD{iLevel},'db1');

end

partialRecon = cA{nLevel};

for iLevel = nLevel:-1:1,
    partialRecon = idwt2(partialRecon,[],[],[],'db1');
end

figure;

imshow([X fullRecon; partialRecon zeros(size(X))],map,...

'InitialMagnification',50);