Lane Departure Warning System

Chanin Charernwan

Chitisan Subtechitmanee

Bachelor of Engineering Program in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

2012

Thesis – Academic Year 2012

B.Eng. in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title: Lane Departure Warning System

Authors:

1. Mr. Chanin Charernwan Student ID : 52090005
2. Mr. Chitisan Subtechitmanee Student ID : 52090007

Approved for submission

.....

(Dr. Chaiwat Nuthong)

Advisor

INTERNATIONAL COLLEGE KMITL

.....

(Dr. Isara Anantavrasilp)

Co-Advisor

Date 19.09.2013

Lane Departure Warning System

Mr. Chanin Charernwan	52090005
Mr. Chitisan Subtechitmanee	52090007
Dr. Chaiwat Nuthong	Advisor
Dr. Isara Anantavrasilp	Co-Advisor

Academic Year 2012

ABSTRACT

Road accidents cause tremendous loss of lives and properties. Each year it has been reported about many deaths concerning road accidents and injured victims. There could be more accidents and losses, which are not reported officially. Many research in the past showed the causes of road accidents. One of the important cause, which has been reported is lane changing. For this reason, the project concerning reduction in road accident rate caused by lane changing is proposed. This project is to create a warning system application for drivers when they unintentionally change driving lane. The proposed application will be developed on iOS platform.

The proposed application obtains series of images from video taken by the iPhone's camera. These images will be processed by appropriate image processing techniques, for examples, converting RGB images to grey-scale images, noise reduction using 2D filter, edge detection, and Hough transform for lane detection. The system will determine whether the car is in lane changing status or not. If the car is indeed in lane changing status, the driver will be warned by the system in the form of sound and/or messages. The application will be implemented on iPhone, which will be installed on the windshield of the car with the camera facing to the road.

By implementing this application, the system should be able to help the drivers in avoiding accidents from lane changing. The system works well in the clear lane markers roads without any casted shadows in the scene. It is assumed that the car is driven on a clear weather. In the future, the system should be able to perform in another scenario as well.

Acknowledgements

This thesis can be successfully completed because of the kindness of advisor team. First and foremost to my advisor, Dr. Chaiwat Nuthong, who gave good advice and be guidance of this thesis until its success and my co-advisor, Dr. Isara Anantavrasilp, who gave a good guidance for software development and the special thanks for Assoc.Prof.Dr.Visit Hirankitti, for all of comments and good suggestions. Thanks also to Dr. Ukrit Watchereeruetai and Dr. Natthapong Jungteerapanich for their kindness and help.

I would like to give special thanks to Mr. Parada Naksook, for his kindness and help in good suggestions and guidance in software development.

Table of Contents

List of Tables

Page

List of Figures

Page

Chapter 1

Introduction

A project concerning reduction in road accident rate caused by lane changing is proposed. This project aims to develop a warning system application called Lane Departure Warning System (LDWS) for drivers in case of unintentional changing driving lane. This system is iOS based application developed on iPhone. The system itself contains two main parts, i.e. lane detection and tracking, and driver warning. It shall recognize the car whether it is changing lane or not. If the car changes lane, driver will be warned by the system.

1.1 Problem Descriptions

According to the studying of road accidents causes. It has been found that one of the main causes is lane changing. The following graph shown in Figure 1.1 supports this claim. This graph is reported by Thailand Accident Research Center (TARC) in 2006. It classifies the causes of accidents and gives a motivation to develop a system in an attempt to reduce this category of accidents.

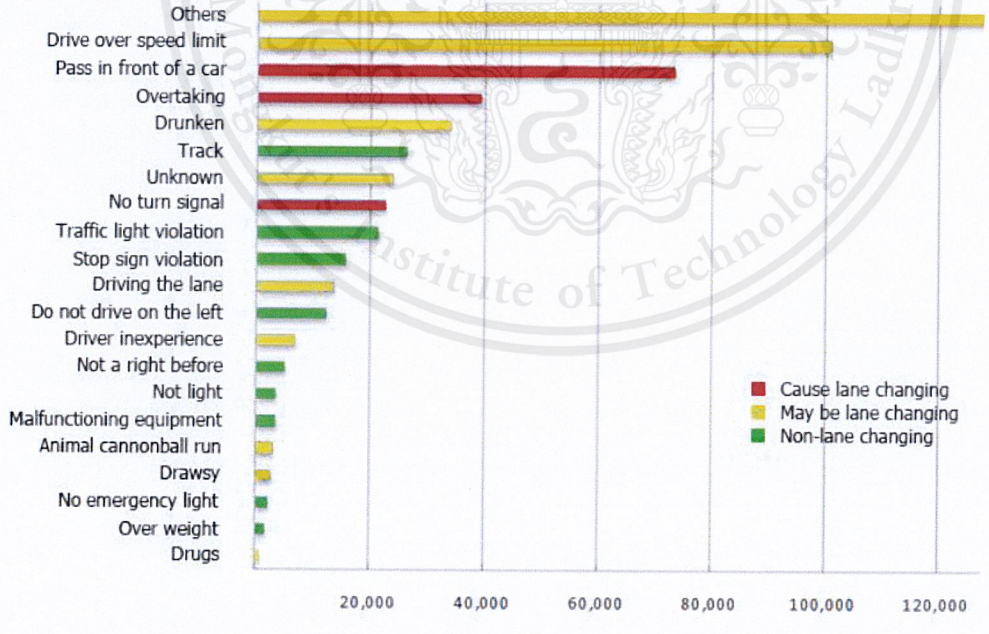


Figure 1.1 Causes of accident

1.2 Objectives

The purpose of this project is to develop a system in order to reduce the accident rate caused by lane changing. The following objectives shall be satisfied.

1. Deliver an LDWS application based on iOS platform.

2. Effectively use the sensors of the smart phone for developing corresponding applications

3. Gain knowledge and understanding in the area of image processing and smart phone applications development

1.3 Scope of Work

The proposed "Lane Departure Warning System" (LDWS) is planned to be developed on iOS platform.

The device use for development is iPhone. The application starts by obtaining a series of images from videos taken by iPhone's camera. These images will then be processed by appropriate image processing techniques including color space conversion, image segmentation, thresholding, edge detection, and Hough transform. After that, the current driving lane should be detected. The system shall warn a driver when a car tends to leave its current lane using sounds and/or messages. The iOS device shall be installed on the windshield of the car with the camera facing to the road. The system shall work well in case of the clear lane markers roads without any casted shadows in the scene. The system assumes that the car is driven on a clear weather. Once the lane changing is detected, the system shall warn the driver early enough for him/her to take the car back to normal driving condition. The development process of this system uses many tools such as Xcode, OpenCV and AVFoundation libraries.

1.4 The Proposed System

The proposed LDWS system is developed as an application on Apple devices licensed by Apple Inc. for example iPod, iPad, and iPhone. It needs to have iOS operating system which is a mobile device operating system developed by Apple Inc., of version 6.0 or higher installed on it in order to work well with the developed LDWS application as shown in Figure 1.2.

Figure 1.2 An iPhone5 with iOS 6.0 and an LDWS application

Since this application operates on road images, its installation location is of importance. For high accuracy purpose, the device should be installed at the middle of the car's windshield as shown in Figure 1.3 with landscape left orientation (back device's camera is at the right hand side of the user). Device's camera should not be obstructed by any objects and a driving vision should be clear.

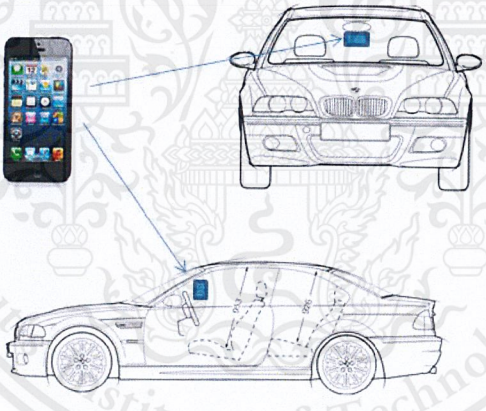


Figure 1.3 Install an iPhone on the middle of car's windshield

Once the LDWS application has been launched and the system is loaded successfully, the device screen should show driving lane as illustrated in Figure 1.4. At this stage, the device might require some user setup. It should specifically be installed such that the device's orientation is horizontal and the middle of its screen is at the center of the lane.

Figure 1.4 After LDWS application successfully launched. The driver's view should display on screen

The system will start to detect the lane. It will draw a current lane area after the lane markers are identified. This is shown in Figure 1.5.

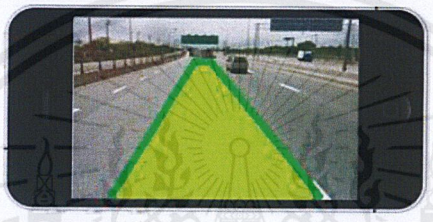


Figure 1.5 LDWS application start detects and draw the current lane

In case of lane changing, the system shall warn the driver such that he/she has enough time to take action. The system can warn by sound, message, and sound or message as shown in Figure 1.6. The user is able to set warning text, sound level and some options warning approval.

Figure 1.6 Warning systems when the lane changing is detected by showing message and/or sound

1.5 Methodology

This thesis focuses on Lane Departure Warning System (LDWS) development. The aforementioned system can be divided into two subsystems. The first subsystem is lane detection and tracking system. In this stage, lanes are identified and tracked by using image processing techniques and Hough transform. The detected lanes are then selected by selecting

algorithm. The second subsystem is driver warning. The driver shall be warned with enough time for correction when the vehicle departs from a current lane.

To achieve the design goal, the development of this project is divided into four steps; (1) gathering all of requirements, (2) analysis and design the system, (3) implementation, and (4) system testing.

First step is to gathering all of requirements including functional requirements and non-functional requirements of the system by ask for user need, collect and distinguish. Second step is analysis and designs the system. The system analysis and design are included some of system artifacts i.e. use case diagram, class diagram, and package diagram. Use case diagram shows functions that user can use on the system and functions that the system provided for the user. Class diagram and package diagram show structure of the system. Next step is implementation. In this step, the system is implemented according to system design. The system implementation should satisfy all requirements. Finally, the system applied for performance testing. This last step is tests and records results for system performance analysis.

1.6 Structure of the Thesis

This thesis is divided into five chapters; Chapter1 introduces background and problems description. It also shows the objectives, scope of work, the purposed system and methodology.

The following chapter is literature review. This chapter summarizes related works from other researchers, which will be used as basic knowledge in system development of the proposed project. Chapter 3 shows the system development platform, software tools and libraries being used. It also discuss about digital image processing techniques, lane tracking and warning techniques. The project management, system development, system testing results, and result analysis and evaluation will be shown in Chapter 4. Finally, discussion of the work and conclusion will be provided in Chapter 5.

Chapter 2

Related Work

2.1 Review of Related Work

This topic is shown related work of this proposed project i.e. SmartLDWS, Lane Detection on the iPhone, Driver Adaptive Warning Systems, and Lane change or "Side" Crash avoidance Systems (SCAS).

2.1.1 SmartLDWS

M. Lan, M. Rofouei, S. Soatto and M. Sarrafzadeh from Department of Computer Science University of California developed a system called "LDWS" [1] to use on a smartphone. An application can detect lanes and warn the driver in case the vehicle is changing lane. The system works by first getting a video from smartphone camera and then obtaining an image from video image buffer. The information in the obtained image is then extracted and processed further. The usage of their image processing technique includes color space conversion, image segmentation, thresholding, edge detection, and Hough transform. The analysis results are then fed to warn the driver. The system can detect sound of the driver and track vehicle velocity via GPS. The aforementioned system was developed on Android platform, and image processing routines part was written in C/C++. This system was developed and run on a T-Mobile G1 phone as shown in Figure 2.1. Their accuracy result is more than 80% on average in several conditions, for examples, night and clear, day and cloudy. Some lane detection results of this work are shown in Figure 2.2 and 2.3 Furthermore, in some conditions such as night and clear, the accuracy is almost 100% in real environments for more than 7 hours.

Figure 2.1 SmartLDWS system running on a T-Mobile G1 phone mounted to the windshield of a vehicle

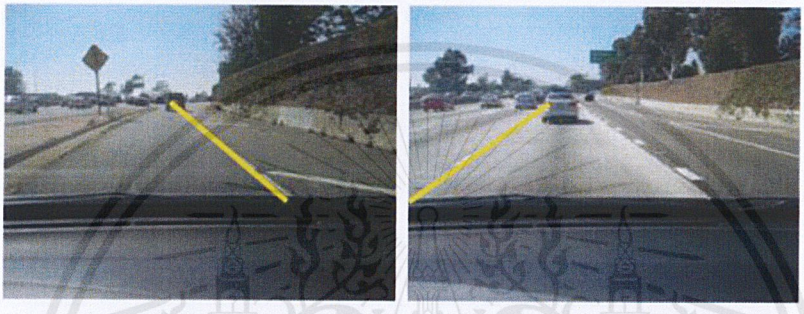


Figure 2.2 SmartLDWS system running on Highway ramp and Highway light traffic

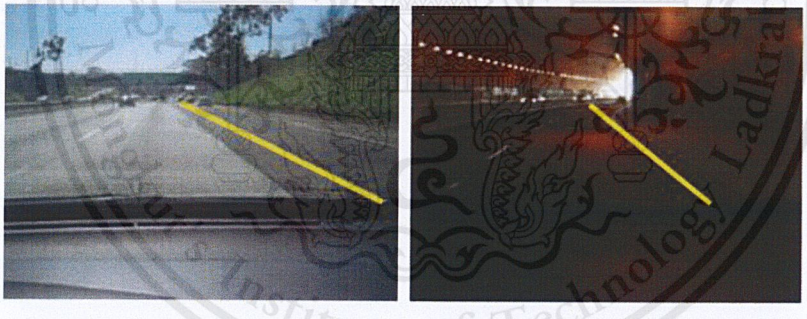


Figure 2.3 SmartLDWS system running on Highway patch work and tunnel high curvature

2.1.2 Lane Detection on iPhone

F. Ren, J. Huang, M. Terauchi, R. Jiang, and R. Klette developed a system called "Lane Detection on the iPhone" [2]. They implemented the application on an iPhone (see Figure 2.4) with restricted resources while given a fairly robust detection result. The system can detect traffic lane by installing the device at the middle of car's console. The operation of this application can be separated into two main parts; image processing and line detection. For the image processing part, the system captures the images from video frame and smoothing them to reduce noise. After that, edge detection is applied to obtain a binary edge map. The second

part is concerning line detection after the processed image is segmented into the area of interest (AOI), the Hough transformation is then applied to detected lines. Then hypotheses testing module has been implemented for these candidate lines using multiple cues such as AOI, lane width, vanishing point etc. The proposed lane detection results show that the system can operate faster and use less memory compared to other lane detection method developed in the same class of device. Furthermore, it is claimed to have accuracy more than 90% in condition of clear straight lane borders. This is illustrated in Figure 2.5.

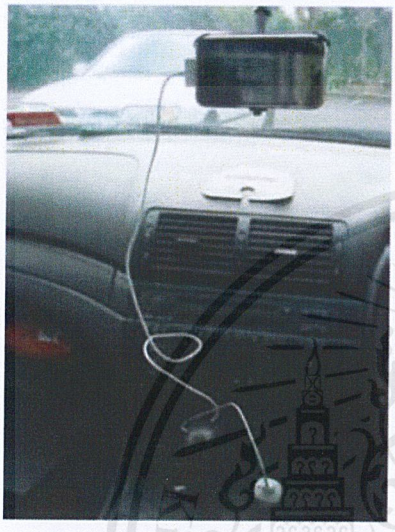


Figure 2.4 Installation of iPhone on an ordinary car

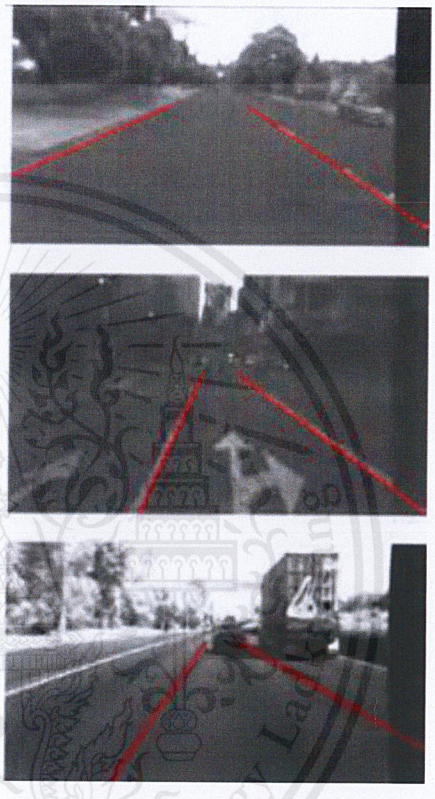


Figure 2.5 Final lane detection result

2.1.3 Driver Adaptive Warning Systems

Parag H. Batavia from The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania developed a system call "Driver Adaptive Warning Systems" [3] his objective is the project's to develop a road departure warning system that learns individual driver behavior and uses this knowledge to reduce false alarms and increase warning time. Based on the objective, he developed an adaptive lane departure and curve negotiation warning system. This system should learn individual traits of the driver -- both stationary and changing, and use this information to improve warning time and reduce false alarms. This Driver Adaptive Warning Systems was included many modules; data collection, Navtech map evaluation,

Curve behavior analysis and modeling development, Lateral control model selection, evaluation on CMRI truck driver data, Accounting for surrounding vehicles and Final user study. He used the Rapidly Adaptive Lateral Position Handler (RALPH) on Navlab 8 the late work to evaluate with his Driver Adaptive Warning Systems. Rapidly Adaptive Lateral Position Handler was a combination of a lane tracker (RALPH), lane departure warning system (RALPHWS), obstacle map generator (OPIE), and vehicle controller (PILOT). And Navlab 8 was a test vehicle, an Oldsmobile silhouette mini-van including a lot of sensors such as CCD camera, Delco radar, Laser sensor, and blind spot sensor as shown in Figure 2.6. The corresponding sensor descriptions were shown in Table 2.1. Evaluation between Rapidly Adaptive Lateral Position Handler and Driver Adaptive Warning Systems were based on system quantitative analysis and system qualitative analysis. The result showed that Driver Adaptive Lane Departure Warning system was better than Rapidly Adaptive Lateral Position Handler on the road.

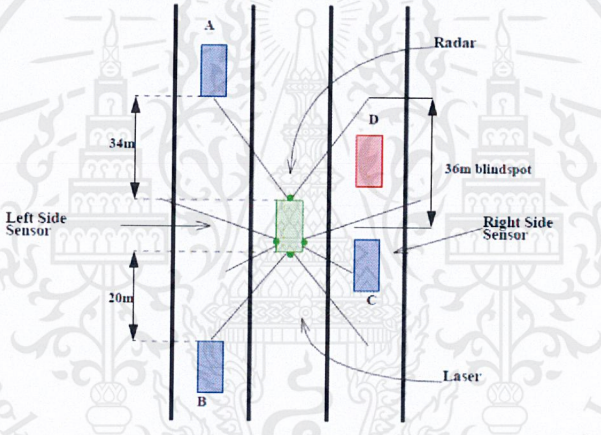


Figure 2.6 Depiction of Navlab 8 Sensor Coverage. The center vehicle is Navlab 8. Vehicles A, B, and C are sensed, vehicle D is in a blind spot

Table 2.1 Sensor Description

Sensor	Range	Field of View	Resolution
Delco Radar	120m	12 degrees	1m/range, 2 degrees
Laser	120m	20 degrees	1 cm.
Blind Spot Sensor	3m	~70 degrees	Binary only

2.1.4 Lane Change or "Side" Crash Avoidance Systems (SCAS)

S. Hetrick from the faculty of the Virginia Polytechnic Institute and State University developed a system called "Side" Crash Avoidance Systems (SCAS) [4]. The main idea of SCAS was the system should warn the driver when two conditions were met: (1) positive indication of lane change intent and (2) positive detection of a proximal vehicle in the adjacent lane of concern. He archived these by examining driver lane change behavior and evaluatingf the potential effectiveness of five warning onset rules for lane change or "side" crash avoidance system (SCAS) technologies. Five warning onset rules for SCAS were evaluated: (1) turn-signal onset (TSO), (2) minimum separation (MS), (3) line crossing (LC), (4) time-to-line crossing (TLC), and (5) tolerance limit (TL). The effectiveness of each rule was measured by the maximum response time available to avoid a crash for a particular lane change crash scenario, and by the crash outcome, crashed or crash avoided, of a particular lane change crash scenario. The results provided an understanding of the lane change crash problem and identify potential countermeasures.

Chapter 3

Background Knowledge

This chapter describes in details the background knowledge of the proposed project including, designing the concerned diagrams, the tools, and image processing techniques used in this project. The aim of this chapter is to give the necessary background knowledge in order to identify the driving lanes. The Chapter is divided into three parts, the first part discuss about the Unified Modeling Language Diagrams. This section gives background knowledge about various type of diagram use in this project. The second part mentions about the chosen developed platform as well as corresponding tools and software use for development. The last part focuses on the image processing techniques and the algorithms concerning lane detection.

3.1 Unified Modeling Language Diagrams

A Unified Modeling Language (UML) is a standard graphical notation defined by Object Management Group (OMG). It is used for designing system model. It consists of many model elements and details as follows.

3.1.1 Use Case Diagram

In UML, use case diagram is used to show all the operation of user and user's relationship with subsystem. In a large system, user is defined as an actor while a subsystem is defined as a use case. The use case diagram can be thought of as user requirements extraction. It uses a human symbol for the actor and oval for use case. The relationship between actor and use case is represented by a line.

3.1.2 Class Diagram

A class diagram is a diagram used to illustrate the classes and the relationship between them. Each relationship in this diagram is a static relationship which means is a normal relationship for the system. In each class, the presented information included class name, attribute, and operation.

3.1.3 Package Diagram

In the UML, a package diagram is used to show the collections of classes and components as well as related item. In general, the package diagram used to group a use case diagram and class diagram of the entire model into more compact relation form.

3.2 Development Platform, Tools and Software Libraries

The proposed system aims to be developed on a mobile device, a smartphone in particularly. Smartphone in the market come with various platforms for examples, iOS, Android, Bada, Blackberry, Windows Phone, and etc. Some of them are more popular among the users such as iOS, and Android. Since iOS is one of the popular smartphone. The proposed project will be developed based on this platform. In order to develop the proposed system, several tools and software libraries are used. The tool involved in project development is Xcode. The system is used AVFoundation as a media library to show driver vision view and used OpenCV as an image processing library to process an image. The details of iOS platform, tool and software libraries are given in the following sections.

3.2.1 iOS Platform

iOS platform is a mobile operating system developed and distributed by Apple Inc. This is the popular platform that has been reconstituted in an optimized form for the mobile device with the touch panel. The Software Development Kit (SDK) has been established in June 2008. It is able to add applications freely in the App Store. The platform uses as an Objective-C essential development language with Cocoa framework.

3.2.2 Xcode

Xcode is the integrated development environment (IDE) of Apple Inc. it is provided free with Mac OS X. Xcode works with Interface Builder, a legacy of NeXT, a tool for creating graphical user interfaces. Xcode includes Compiler Collection GNU (GCC) and can compile C, C++, Objective-C, Objective-C, Java and AppleScript through a wide range of programming models, including but not limited to Cocoa, Carbon and Java.

3.2.3 OpenCV

OpenCV is a software library originally developed by Intel. It released under a BSD license that means it free for academic and commercial use. It supports for many platforms e.g. Windows, Linux, Mac OS, iOS and Android. It contains many functions cover a wide range of used in computer vision areas e.g. object recognition (facial recognition), camera calibration, stereo vision and robot vision. In this project, it used as an image processing programing library. It used to process an image for lane detection and tracking lane.

3.2.4 AVFoundation

AVFoundation framework is a software library presented on iOS. It enables the system (in this system an iOS device) reproduce and create real time multimedia. It also provides an Objective-C interface that allows the system to manipulate multimedia files. In this project, it

used as software library for get a video stream buffer from iPhone's camera and capturing an image.

3.3 Image Processing

The proposed project detects and tracks a driving lane and warns a driver based on the picture extracted from video. For this reason, there are many image processing techniques involved in this project. The following section describes them in details.

3.3.1 Image Conversion

The image to be processed in this project is the gray-scaled image. However, the obtained input image is usually an RGB image. Thus the image conversion fom RGB to greyscaled image is required. The intensity value of the resulted image ranges from 0 to 255. The following equation shows the conversion from RGB to intensity value in the grey-scale image,

$$Intensity = 0.2989 * R + 0.5870 * G + 0.1140 * B \tag{3.6}$$

3.3.2 Gaussian Blur

A Gaussian blur or Gaussian smoothing is the result of blurring an image by a Gaussian function. It is a widely used effect in graphics software, typically to reduce image noise. However, the image's detail will be reduced as well. The equation of a Gaussian function is

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.7}$$

3.3.3 Edge Detection

Edge detection is a fundamental tool in image processing and computer vision, particularly in the areas of feature detection and feature extraction. If has been used for identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities.

There are many edge detectors available, however this proposed project uses the Sobel edge detector. The Sobel edge detector has some smoothing effect to the random noise of the image resulting in thick and bright edges. It is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction. Therefore, it is relatively inexpensive in terms of computations.

Mathematically, the detector approximates two derivatives - one for horizontal changes, and one for vertical. For the source image, and G_x and G_y are two matrices, which at each point contain the horizontal and vertical derivative approximations.

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, as in the following:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.8}$$

And the gradient's direction:

$$\theta = atan\left(\frac{G_y}{G_X}\right) \tag{3.9}$$

The resulted image will be a binary image with the edge feature

3.3.4 Image Morphology

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. It has two main fundamental operations are erosion and dilation. For erosion, holes and gaps between different regions become larger, and small details are eliminated. For dilation, holes enclosed by a single region and gaps between different regions become smaller, and small intrusions into boundaries of a region are filled. In this project, *opening* morphology methods is used. It is dilation of the erosion morphology technique. The *opening* method should remove small objects from the foreground of an image and placing them in the background. This technique can be used to find shapes in an image and also used to find things into which a specific structuring element can fit such as edges, and corners.

In mathematical morphology opening is the dilation of the erosion of a set A by a structuring element B:

$$A \circ B = (A \ominus B) \oplus B \tag{3.10}$$

Where $\ominus$ and $\oplus$ denote erosion and dilation, respectively.

3.3.5 Hough Transform

Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing. It is concerned with the identification of lines in the image. The purpose of this technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from

which object candidates are obtained as local maxima in a so called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

Hough transform is an alternative form of representation of a line, which is defined as follows,

$$\rho = \mathrm{x}\cos\theta + y\ \sin\theta \tag{3.11}$$

Where ρ is referred to the distance from the origin, x and y are the distance in x and y direction and θ is an angle measured from x-axis to the line perpendicular to the line passing through the concerned point. Note that there are infinitely many lines that can pass through a point in the x plane. Each line possesses value of ρ and θ. Once plotting these values in $\rho\theta$ plane, it can be seen that this results in a sinusoidal curve. Thus, a point in the xy plane becomes a corresponding sinusoidal curve in $\rho\theta$ plane.

It has been known that there is a single straight line, which passes through two points in the xy plane. The graph is shows that this line has unique parameters ρ and θ (see Figure 3.1). Thus, the line that passes through two points in the xy plane will be represented by an interception point of the two sinusoidal in $\rho\theta$ plane. This characteristic of Hough transform is used extensively to find the lines in the image.

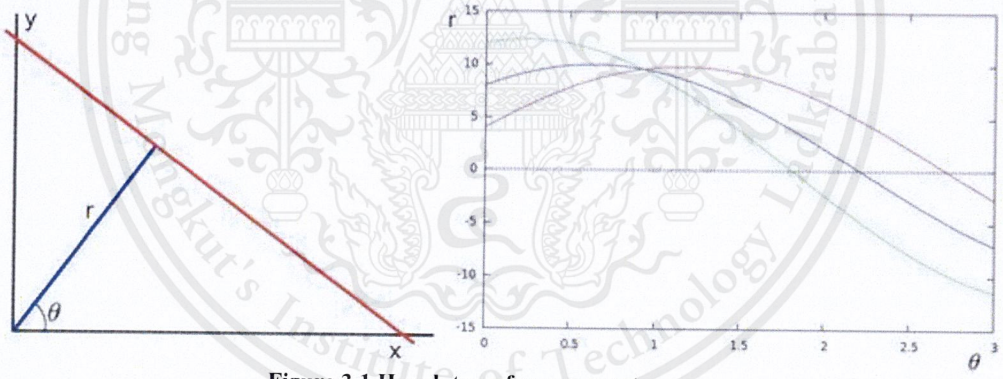


Figure 3.1 Hough transform concept

Since the line is actually a collection of points. Thus by counting a number of points for each parameter ρ and θ, it can be seen that the higher score for particular values of ρ and θ, the more possibility the lines will be detected. Note that the standard Hough transform will detect the lines which have an infinite length. In order to have the lines with finite length, additional algorithms will be required.

Since a lane is composed of two lines. It is likely that the two highest score of the parameters ρ and θ formed the detected lane.

Chapter 4

System Overview and Development

This Chapter discuss about the proposed system overview and the step for system development. It also includes project management, and experiment set up and results as well.

4.1 System Requirement

System requirement is a requirement specification for a software system. For this proposed project, system requirements are divided into two groups i.e. functional requirements and non-functional requirements

4.1.1 Functional Requirements

Functional requirement is a main requirement or role that the system should have. In this project the functional requirement are agreed to be as follows:

1. A program supports iOS 6.0 or higher
2. A program can identify the current driving lane
3. A program can identify lane markers by drawing virtual lines over the current driving lane
4. A program can detect whether there is a lane changing or not
5. A program can warn with sound and/or message when there is a lane changing
6. Driver can set the warning function to message, sound, or message and sound, and also adjust the sound level and message style
7. When drive changes the lane, a new virtual line will appear as a new lane and the old line will disappear

4.1.2 Non-function Requirements

Non-functional requirement is other requirements that are not the main requirements or roles of the system but the system should have them for security, reliably, responding time or support I/O. In this project the non-functional requirements are agreed to be as follow:

1. A program detects traffic lane accurately and reliably
2. A program warns accurately and reliably
3. A program has a friendly user interface

4.2 System Analysis and Design

System analysis and design is a process to create specifications of a software artifact. System analysis is a process of collecting data, understand the processes, identifying problems and recommending feasible suggestions for create the system functioning. Systems design is the process of defining the components, modules, interfaces, and data in the system to satisfy specified requirements. The system analysis and design for this project result in the following software model. This software model includes use case diagram, class diagram and package diagram of LDWS system. All of these diagrams are discussed in details in the following sections.

4.2.1 Use Case Diagram

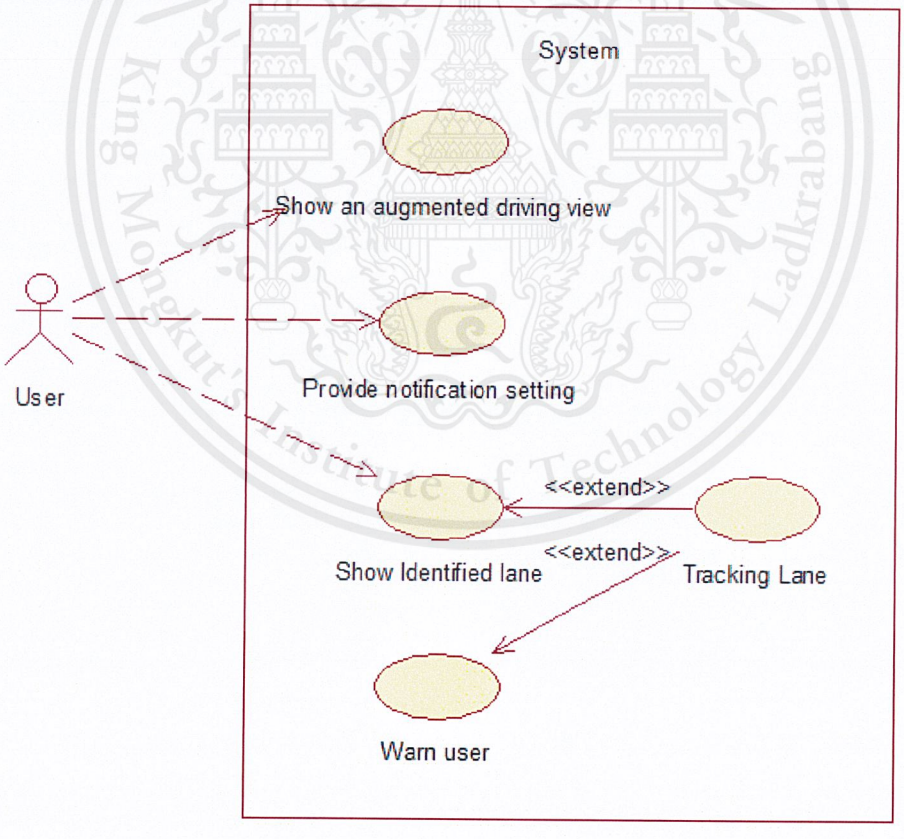


Figure 4.1 Use cases diagram for LDWS

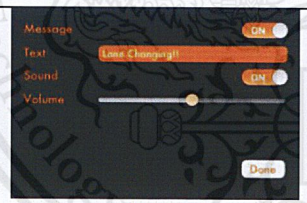
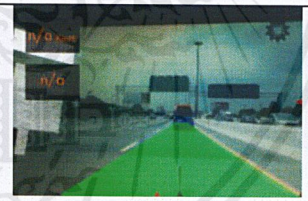
The use cases diagram had shown in Figure 4.1 shows relationship between the user (so called "Actor") and the systems features (so called "Use case").

Actor is the user who has interaction with the system while use case is the function or feature that the actor can use or the function that system has provide for the actor. From the above figure, this system has only one user (driver). A user can see an augment driving view, set notification, see a current lane by lane identifier, and get warning when a car tend to changing lane. The system provides all of these features to a user. From the system side, it can track lanes in order to identify a current lane and warn user. Table 4.1 shows actor name and actor description. Use case description, actor of use case, flow of events of use case, and user interface of use cases are show in Table 4.2.

Table 4.1 Actor in LDWS system use case diagram.

Actor	 User
Description	User is an actor, who can see an augmented drive view and set notify setting

Table 4.2 All of use cases in LDWS use case diagram

Use Case	 Show an augmented driving view	 Provide notification setting	 Show Identified lane	 Warn user	 Tracking Lane
Description	This Use Case used by a user to see an augment driving view from the device	This Use Case used by a user to set notification setting as message style, sound level or etc.	This Use Case used by the system to detect lines representing the lanes and show the result on the screen	This Use Case used by the system to warn the user when changing lane	This Use Case used by the system to tracking lane when got the image from video
Actor	User	User	User	-	-
Flow of Events	1. This Use Case begins when a user starts the system. 2. The system gets a drive vision video frame from the device and display with augment line representing the lanes on screen.	1. This Use Case begins when a user pushes the setting button. 2. The system will show all of notification setting for the user. 3. When setting is finish and push back, the system will set all new setting to the system.	1. This Use Case begins when the system gets video from camera. 2. The system gets an image from buffer. 3. The system processes an image. 4. The system detects lane lines. 5. The system draws a virtual line representing the lanes on the screen.	1. This Use Case begins when the system checks lane. 2. When lane changing happens the system will warn the user as pre-setting information	1. This Use Case begins when the image obtained from the video. 2. Image processing to identify lane 3. Show the result on screen
User Interface					

4.2.2 Class and Package Diagram

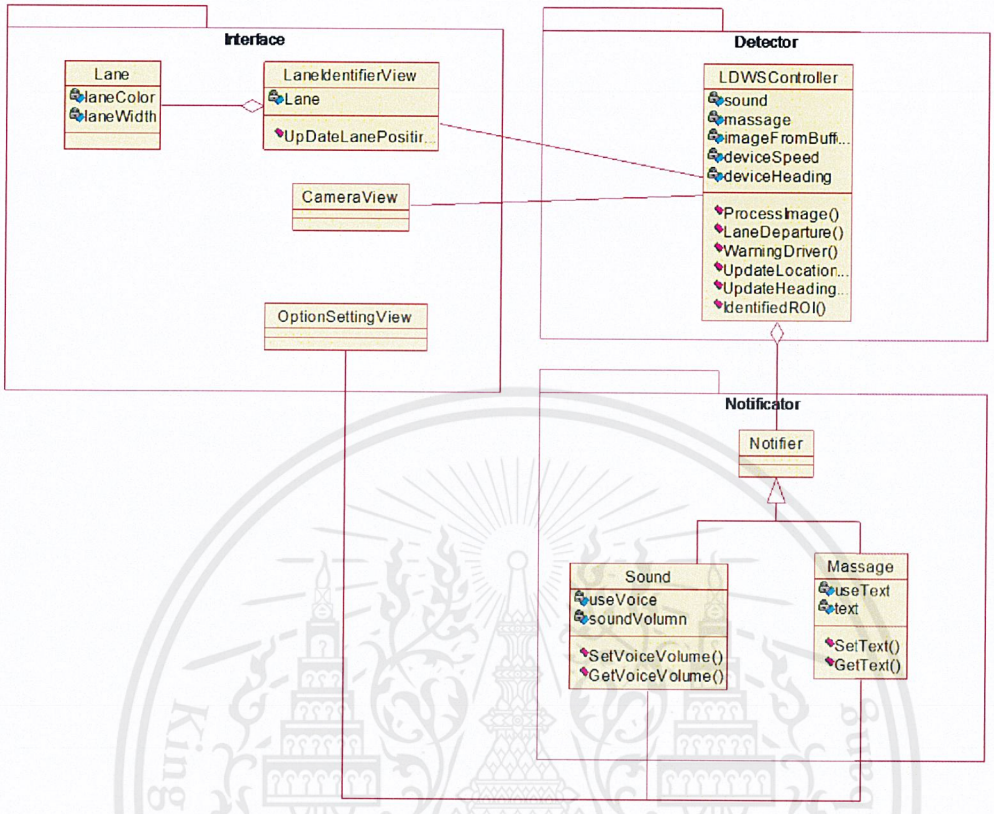


Figure 4.2 LDWS system class diagram

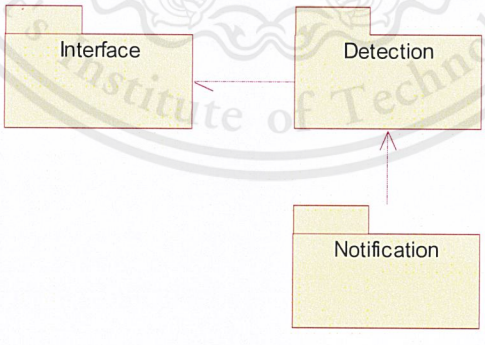


Figure 4.3 LDWS system Package diagram

The class diagram illustrates the system structure and relationship between classes in the systems. It has two main properties i.e. attribute and operation. Attribute in each class is component of class. Operation in each class is defines as function that the class can perform. The structure of class diagram for LDWS system is show in Figure 4.2. The classes in LDWS

system are grouped into three main grouped; i.e. Interface, Detector and Notificator. Each Group of classes is called "Package".

The first package is Interface package. It has four classes in this package; i.e. Lane, LaneIdentifierView, CameraView and OptionSettingView. Lane class is store lane properties and has aggregation relationship with LaneIdentifierView class. LaneIdentifierView class is use to update and display lane identifier on augmented view. It has association relation with LDWSController class and be aggregate by Lane class. So CameraView and OptionSettingView class are a system interface class. For Detector package, it has only one class in this package is LDWSController class. This class is a system control class. It used to processing images, detecting and tracking lanes and warning a driver. It has association relation with LaneIdentifierView and CameraView class. The last package is Notificator package. It has three classes in this package; i.e. Notifier, Sound and Message class. Notifier class is an abstract class (class that declared abstract and can be subclassed). It has two subclasses are Sound and Message class. Sound class is a subclass of Notifier class. It used to store warning sound volume and warning sound approval, and changing sound volume. For the last class, Message class is a subclass of Notifier class. It used to store warning text message and waring text approval and changing warning text. The details of each class are shown in Table 4.3–4.5.

Table 4.3 Details of classes in Interface Package

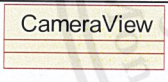
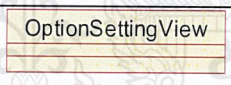
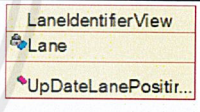
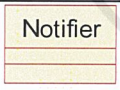
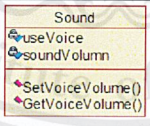
Class				
Class Name	CameraView	OptionSettingView	Lane	LaneIdentifierView
Attribute			• laneColor: store lane color. • laneWidth: store width of lane marker.	• lane: store lane and its property.
Operation				• UpDateLanePosition(): update lane position and display on augment view.

Table 4.4 Details of class in Detector Package

Class	
Class Name	LDWSController
Attribute	• message: store text and text display approval. • sound: store sound volume and sounding approval. • imageFromBuffer: get an image from stream video. • deviceSpeed: store device speed. • deviceHeading: store device's direction.
Operation	• ProcessImage(): pre-image processing as grey-scale, filtering, edge detection, thresholding, and lane tracking function. • LaneDeparture(): tracking the lane via Hough transform algorithm in combination with lane selecting algorithms. • WarningDriver(): apply warning • UpdateLocation(): update device's location and speed. • UpdateHeading(): update device's direction. • IdentifiedROI(): identify ROI from input image.

Table 4.5 Details of classes in Notificator Package

Class			
Class Name	Notifier	Sound	Message
Attribute		• useVoice: store sounding approval. • soundVolume: store sound volume level.	• useText: store message display approval. • text: store text for message displaying.
Operation		• SetVoiceVolume(): set sound volume level. • GetVoiceVolume(): get sound volume level.	• SetText(): This method used to store text. • GetText(): This method used to get text.

4.3 Process Development

System development started after system analysis and design was done. The system in this project is divided into two sub systems; i.e. lane detection and tracking system and driver warning system. The task of this process development is divided into three main tasks; (1) initialization, (2) lane detection and tracking system, and (3) driver warning systems.

Initialization discusses about hardware and software preparation prior to process development. In the stage of lane detection and tracking, the system shows the method to gathering image from device's camera, image pre-processing, and lane tracking using Hough transform. Finally, the driver warning system was discusses all of warning techniques. Figure 4.4 illustrates the flow of activities in process development. The following sections discuss each task in details.

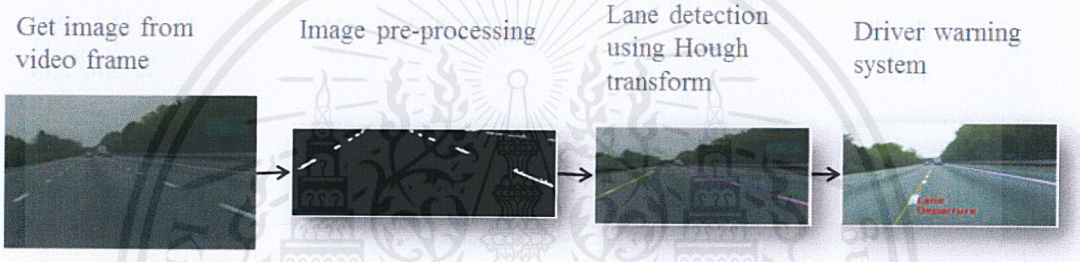


Figure 4.4 Flow of activities in development

4.3.1 Initialization

Generally, mobile devices need some initialization before developing any application on them. This also applied to iPhone, a mobile device chosen for this project. At first, developer needs to have an Apple ID and be a member of Apple Developer. Furthermore, Mac OSX should be updated to version 10.8 (Mountain Lion) to support an Xcode installation. Certificates and provisioning profile are also needed to download and installed on developing device in this project iMac was a developing device. In order to test the development application, the testing device is needed to be registered before used for test. Note that, every time when a new testing device was registered. Provisioning profiles should be updated.

4.3.2 Lane Detection and Tracking System

After the initialization has been performed on both developing and testing device an application can now be developed. At this stage the application will be developed. Xcode is the tool used for this purposed. The developed application includes AVFoundation Library for

load, set, and show a video from video stream buffer. Video stream buffer is a set of images received by device's camera. Then, the proposed Implementation method captures and processes images in real time.

After an image is obtained, it will be scaled into 4:3 ratios and resized into 120 x 90 pixels. This resizing is done for the purpose of reduction in processor resource usage. Next, the pre-processing image is performed. It starts with region of interest (ROI) identification and is followed by image conversion, image filtering, edge detection and morphology. Hough transform with lane tracking algorithm is the last step for lane detection and tracking system. This last operation is responsible for detecting lines in the image.

4.3.2.1 Image Pre-Processing

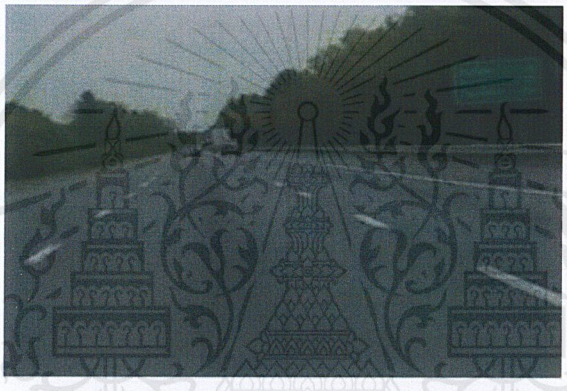


Figure 4.5 an image from video stream buffer

A captured image obtained from the device is the driver's view image including many objects which does not concern in lane detection and tracking for example the area of the sky in the image. This is shown in the upper part of figure 4.4. Setting up the ROI should discard those irrelevant areas and focus only on the area of interest, e.g. the road area. This would make the computational more effective than using the original image.

In this project, ROI are specifying by finding a vanishing point. Since input image obtained from the video stream is a perspective image. Thus every line in this image should intersect at one point so called "vanishing point".

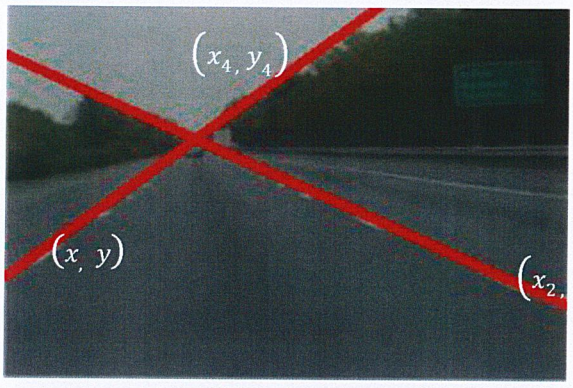


Figure 4.6 Lane tracking

In order to find a vanishing point, two lane lines will be tracked for the first time. Then, the intersection point of these two lines as shown in Figure 4.5 should be found.

Figure 4.7 an intersection point from lane line

By using this approach, strait line equation of both lane lines are needed. If their intersection point is found, the ROI can be identified.

For example, an equation of the strait line parsing though point can be express as

$$y - y_1 = m_1(x - x_1) \tag{4.1}$$

Thus, the two equations of both lanes can be express as

$$y - y_1 = m_1(x - x_1) \tag{4.2}$$

And

$$y - y_3 = m_2(x - x_3) \tag{4.2}$$

Where m_1 is the slope of the left lane and m_2 is the slope the right lane. m_1 and m_2 can be specified as

$$m_1 = \frac{y_1 - y_2}{x_1 - x_2} \tag{4.4}$$

$$m_2 = \frac{y_3 - y_2}{x_3 - x_2} \tag{4.5}$$

The vanishing point of this two lane is (x_4, y_4) as shown in Figure 4.11 y_4 can be found as

$$y_4 = \left(\frac{m_2}{m_2 - m_1}\right)(y_1 - m_1 x_1) + \left(\frac{m_1}{m_1 - m_2}\right)(y_3 - m_2 x_3) \tag{4.6}$$

Figure 4.8 Crop image start from y of vanishing point

After y value of an intersection have got, the height of an image was crop from bottom until y but the width of an image is the same as an original image This is shown in Figure 4.9

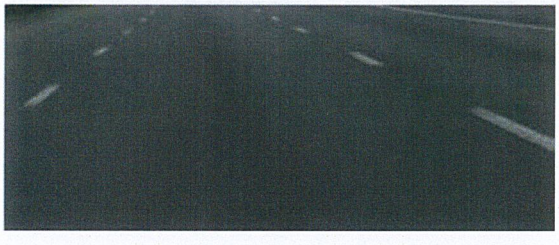


Figure 4.9 Result image

After ROI identification is defined the image (RGB) should be converted into a grey-scaled image (Figure 4.10). The intensity value of the resulted image ranges from 0 to 255. And the dimension of this imaged is reduced to 2

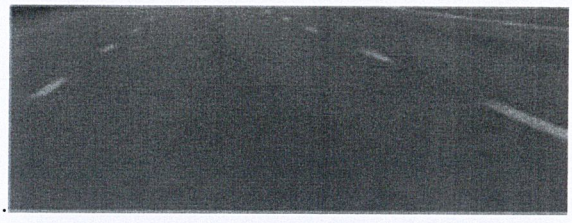


Figure 4.10 Image Conversion

After an input image is converted to a grey-scaled image, image filtering is applied. The Gaussian smoothing filter was selected for this project. This filter is used in order to filter out the noise from the original image. The resulted image will have lower noise (blur image) as shown in the Figure 4.11.

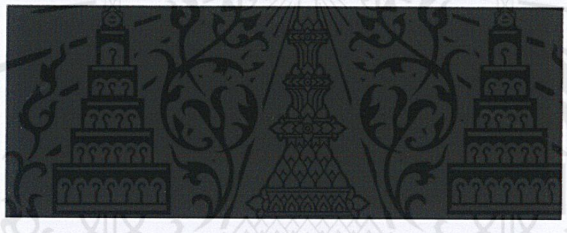


Figure 4.11 Image Filtering

At this stage, edge detection is applied. There are many edge detector proposed in the literature. However, Sobel edge detector has been chosen to use in this project. It is used to identify border of an object in the image. The results image is shown in the Figure 4.12.

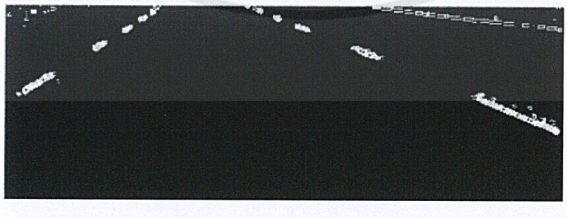


Figure 4.12 Edge Detection

Although, the edges in an image are founds. There are still some irrelevant edges included. Image morphology an Opening in particular is applies to the system. In order of fill small white areas in the image. The results image is shown in the Figure 4.13.

Figure 4.13 Image Morphology

Since, the geometry of the lane on the road is actually a line. Thus, the algorithm that can detect lines should be used for lane detection. For this reason, the Hough transform method is chosen and applied for lane detection. As it can be seen from Figure 4.14, at the end of this stage the lane can be detected.

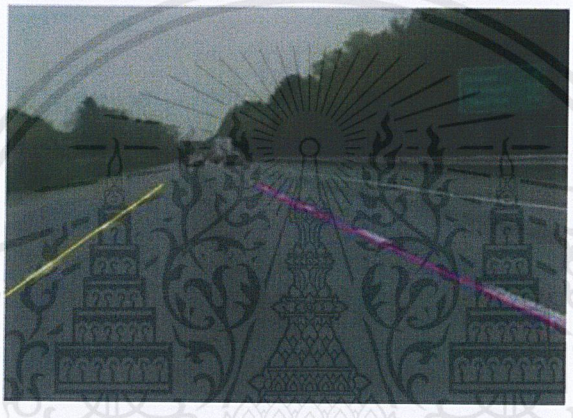


Figure 4.14 Lane Detection

4.3.2 Driver Warning System

The warning system is another subsystem of LDWS. The purpose of this subsystem is to warn the driver as the vehicle to deviate from the current lane. There are two main features used to warn the driver, i.e. sound and message. The combination of both features is also possible. Several warning techniques have been implemented in this project. The following section illustrates the details.

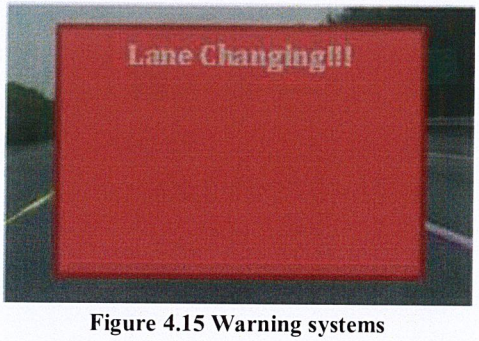


Figure 4.15 Warning systems

4.6.2.1 Warning by Zone

Since the lane is modeled straight lines, they possess two parameters, i.e. slope and interception point. These two parameters will be used in driver warning algorithm. Figure 4.16 illustrates the concerned parameter for warning system. The center line of the image indicates the current position of the vehicle while the line at the lower edge of the image is divided into two different zones, i.e. warning zone and critical zone. These zones will be used for a driver warning algorithm

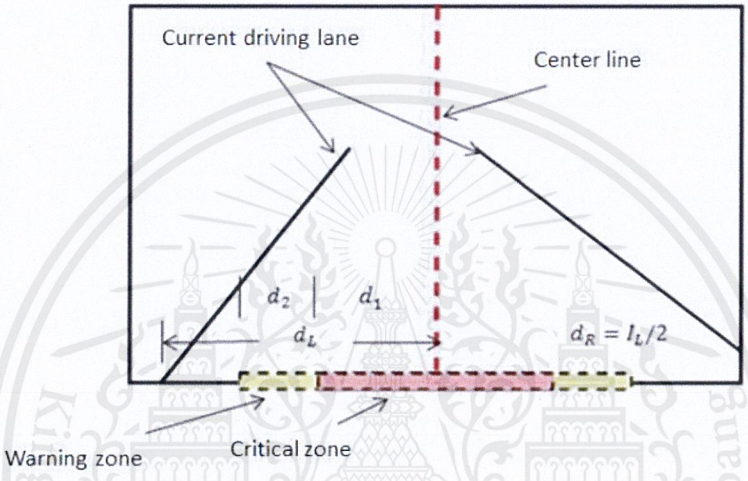


Figure 4.16 View of normal driving condition and corresponding parameters

In general, driving condition is classified into three cases

1. Normal driving condition
2. Warning driving condition
3. Critical driving condition

In the normal driving condition (see Figure 4.16) the interception point of the left (right) lane and the line at the lower edge of the image has a distance $(d_L + d_R)$ measured from the center line satisfies the following condition:

$$d_L > d_1 + d_2 \tag{4.7}$$

$$d_R > d_1 + d_2 \tag{4.8}$$

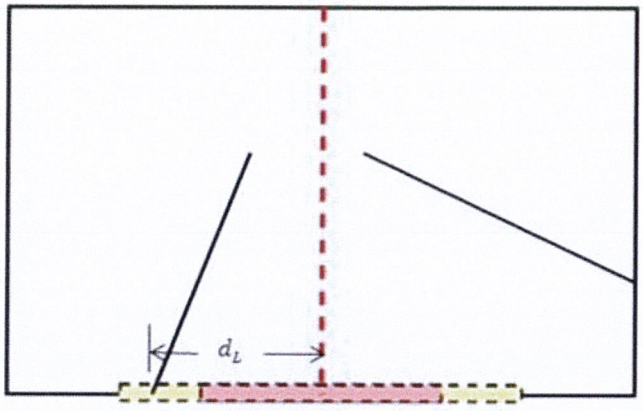


Figure 4.17 View of warning driving condition

In case of normal driving condition, no warning features should be activated. In case the interception point falls into the yellow area, this is classified as a warning driving condition and the yellow area is called the warning zone. It is define to be the zone such that the distance measured from the center line d_w satisfies the following condition:

$$d_1 < d_w < d_1 + d_2 \tag{4.9}$$

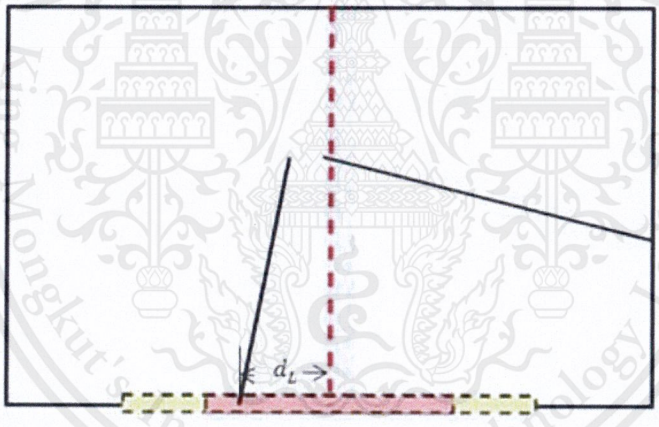


Figure 4.18 View of critical driving condition

In this case the algorithm should warn the driver as he/she tends to leave the current lane. For the last case, the interception point falls into the red area, this is classified as critical driving condition and the red area is called the critical zone.

In the similar manner, the critical zone is defined to be the zone such that the distance measured from the centerline d_c satisfies the following condition:

$$0 < d_c < d_1 \tag{4.10}$$

The algorithm should warn the driver as he/she tend to leave the current lane.

4.3.2.2 Warning by Swipe

Since the lane is modeled as two straight lines one can find the angle (θ) between these lines and the horizontal line. This angle can be used for warning algorithm as well. Figure 4.19 illustrates the view of normal driving condition and corresponding parameters. θ_L and θ_R are the angles corresponding to left lane and right lane respectively.

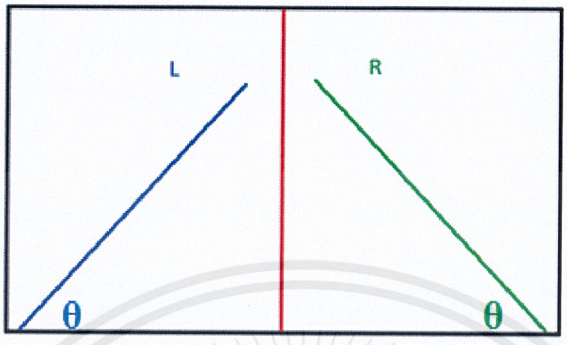


Figure 4.19 View of normal driving condition

Assume that the vehicle tend to depart the right hand side. As the vehicle continues to departure the value of θ_L will decrease and the value of θ_R will increase as shown in Figure 4.20.

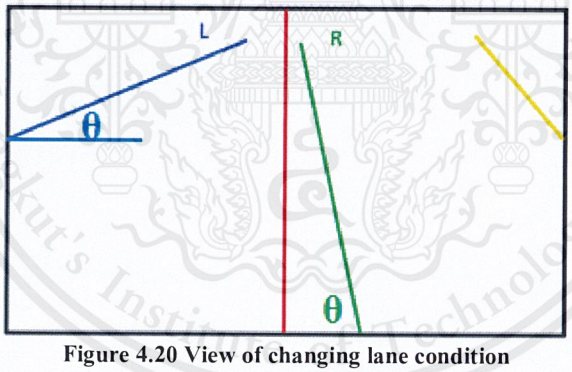


Figure 4.20 View of changing lane condition

At the instance, the system is able to detect a new right lane. The value of θ_R will drastically decrease (swipe out) as show in the Figure 4.21

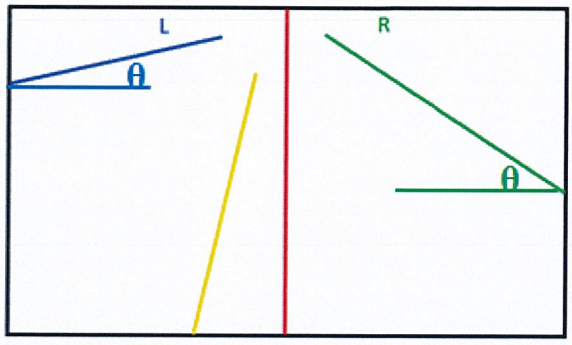


Figure 4.21 View of changing lane condition

Then, the past right lane becomes a new left lane. The value of θ_L will drastically increase (swipe in) as show in Figure 4.22.

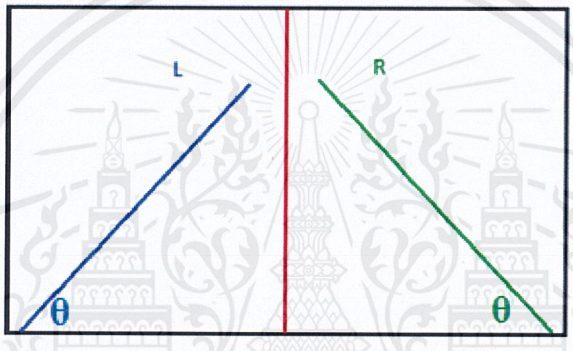


Figure 4.22 View of normal driving condition

As the rapid change of θ_L or θ_R is detected this will activate the driver warning system. Thus the system should warn the driver in accordance of occurrence of swipe in or swipe out.

4.4 Experimental Setup and Results

The graphical user interface (GUI) of the developed application is shown in Figure 4.23 and 4.24. The GUI consists of two main views i.e. augmented view and setting view. The augmented view was included driving view, current lane identifier, speed meter, world directions meter, horizontal phase shift, vertical phase shift, and setting button as shown in Figure 4.23.

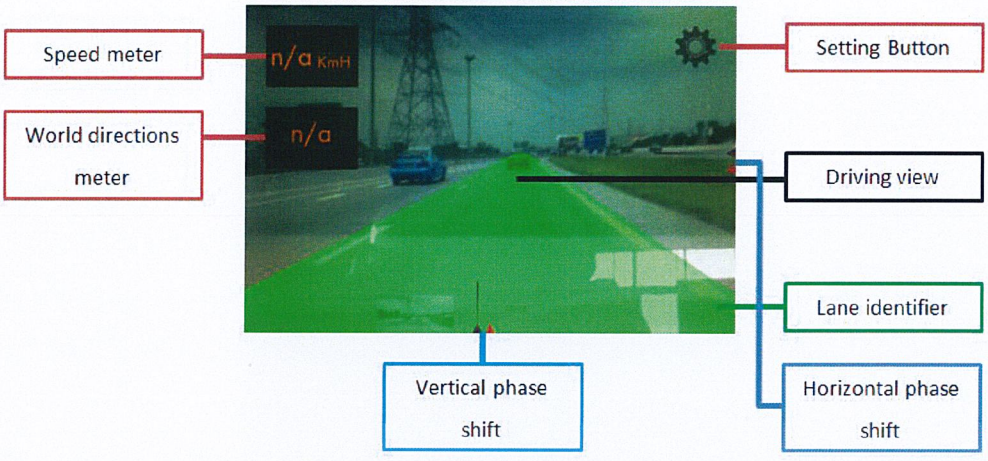


Figure 4.23 Augmented view of the developed application

In the driving view, the image of road is displayed. It is the augmented with a lane identifier as shown in Figure 4.23. Speed meter on the upper left corner and world directions meter below shows a device speed in unit of kilometers per hour and device direction compare with world directions marked on the screen. The horizontal phase shift and the vertical phase shift are a user on installation the device in the proper position. Lastly, setting button on the upper right corner is the button used to access the setting view on the system.

The second view of this application is called setting view. This view includes message warning approval, text fields for text changings, sound warning approval, sound volume slider and a push button labeled "done". This is illustrated in Figure 4.13.

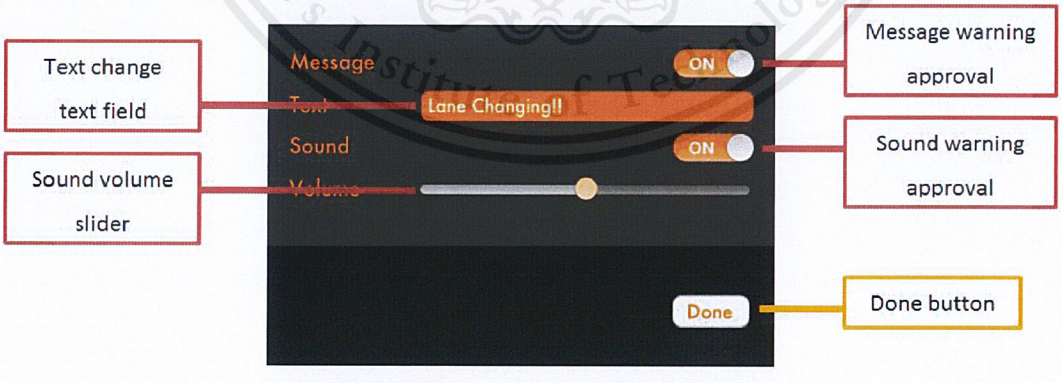


Figure 4.24 Result system a setting view

In this setting view, a user can set both message and sound warning options. He/she can also change warning text on text field and set a sound warning volume. The user can go back to the augmented view by pressed done button.

4.4.1 Experimental Setup

Before testing the system performance, tools and equipment are needed to be prepared. Firstly, a video recording of real driving scenario shall be installed in a simulator (iMac). Secondly, a test device (iPhone in this case) should be register and installed LDWS application on it.

In order to test the system, the recorded video is played on iMac. The testing device is installed in front of iMac to get a clear view of the road from the video. Note that, after installation of the testing device, the image of the road should show up on the screen clear enough as show in Figure 4.25 and Figure 4.26.

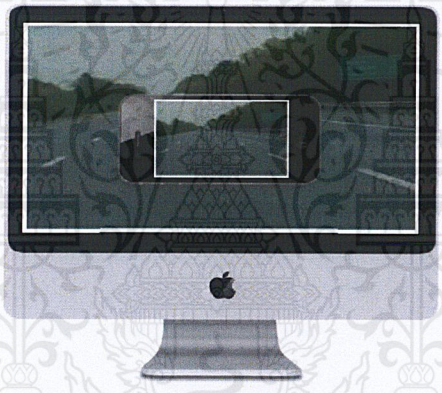


Figure 4.25 System experimentation

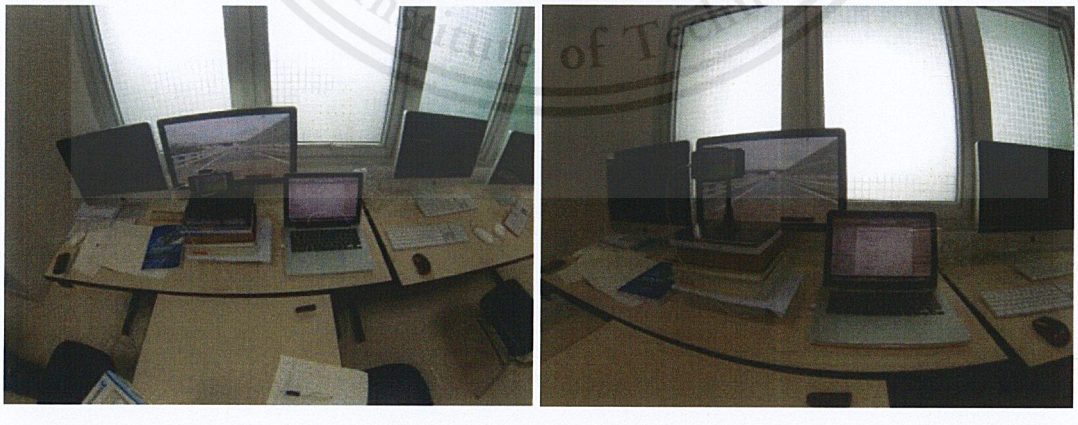


Figure 4.26 Testing device installations on virtual driving environment

The LDWS application with various algorithms can now be tested using the record video. The details of testing are discussed in the following section.

4.4.2 Experimented Results and Analysis

After setup and run the LDWS application, the results will be recorded. They are collected base on three main following techniques.

1. Warning by swipe "swipe out"
2. Warning by swipe "swipe in"
3. Warning by zone

The performance of the tests are measureed as warning accuracy. This is done by checking system alarm. The system warning can be categorized as the following Table.

Table 4.6 system warning cases

	Should alarm	Should not alarm
Alarm on	✓	×
Alarm off	×	✓

The case when the system should alarm and the alarm is on and the case when the system should not alarm and the alarm is off are grouped together as a correct action. The other two cases are classified as an incorrect action.

In order to analyze system performance, there are two sets of video recording. The first set is checked and searched manually for the instances the driver should be warned and the instances the driver should not be warned. For example, at the instances 0-10 s, the vehicle tends to leave the lane so the driver should be warned. However at the time in stances 10-15 s, the vehicle stays in the current lane. In this case the alarm should be off. Similar results are collected and put the table as illustrated on the Table 4.7 and 4.8.

Table 4.7 Performance test of first recorded video

	swipe out		swipe in		warning by zone	
Time	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
0.00-0.08	●		●		●	
0.08-0.12	●		●		●	
0.12-0.39	●		●		●	
0.39-0.42		●	●			●
0.42-0.50	●			●	●	
0.50-0.59		●	●		●	
0.59-1.09	●			●		●
1.09-1.13	●		●		●	
1.13-1.14	●		●		●	
1.14-1.18		●	●		●	
1.18-1.25		●	●		●	
1.25-1.27		●		●	●	
1.27-1.28	●		●		●	
1.28-1.30		●		●	●	
1.30-1.31	●		●			●
1.31-1.35	●			●	●	
1.35-1.41		●		●		●
1.41-1.56	●		●		●	
1.56-1.59	●		●		●	
1.59-2.18	●		●		●	
2.18-2.25	●		●		●	
2.25-2.29	●		●		●	
2.29-2.31		●	●		●	
	15	8	17	6	19	4
	65.21%	34.79%	73.91%	26.09%	82.6%	17.4%

Table 4.8 Performance test of second recorded video

	swipe out		swipe in		warning by zone	
Time	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
0.00-0.25	●			●	●	
0.25-0.29	●		●		●	
0.29-0.36		●		●	●	
0.36-0.38	●		●		●	
0.38-0.41		●		●	●	
0.41-0.45	●		●		●	
0.45-0.48	●		●		●	
0.48-0.50	●		●		●	
0.50-0.55	●		●		●	
0.55-1.43	●		●			●
1.43-1.47		●	●		●	
1.47-2.05	●			●		●
2.05-2.07	●		●		●	

	swipe out		swipe in		warning by zone	
Time	Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
2.07-2.12	●		●		●	
2.12-2.13	●		●		●	
2.13-2.16	●		●		●	
2.16-2.54	●			●		●
2.54-2.57	●		●		●	
2.57-2.58	●		●		●	
2.58-3.00		●	●		●	
3.00-3.11	●		●		●	
	17	4	16	5	18	3
	80.95%	19.05%	76.19%	13.71%	85.71%	14.29%

The system perfomance test has been set and theaccuracy of each warning algorithm is analyzed. The results show that warning by area technique has highest accuracy and following by swipe out technique and swipe in technique respectively. This can be seen more clearly in Figure 4.27. Note that the system testing results are obtained from simulation test with specific parameter values setting.

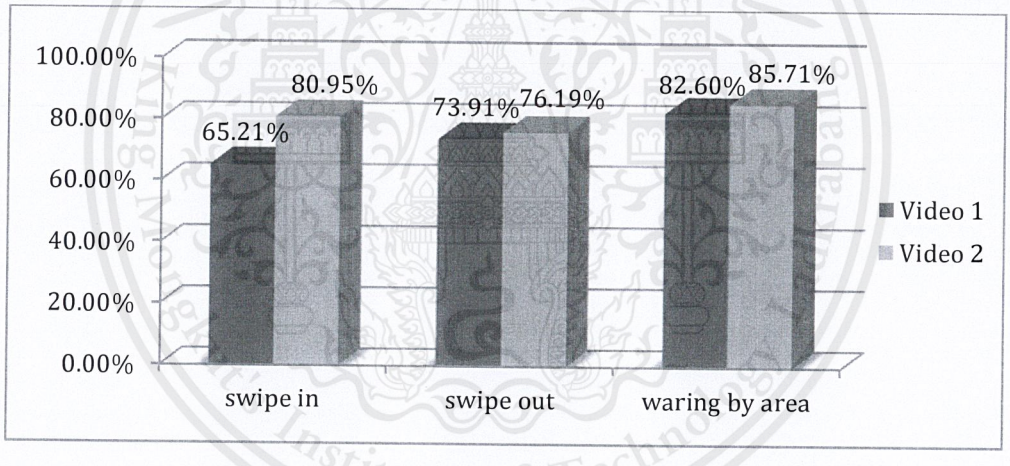


Figure 4.27 Testing summary results

Chapter 5

Discussion and Conclusion

This Chapter provides discussion about the problem and future work. The conclusion of this thesis is also summarized as well.

5.1 Discussion

There are many skills and knowledge included in project development; i.e. digital image processing, application development on iOS platform, Objective-C programing language, OpenCV libraries and teamwork.

In addition, many problems had been during development process i.e. in the beginning, the problem of capturing an image from video arises. This is because the result image size is too large. Large size images affect the system such that they slow down the system response. This problem is solved by acquiring image from video buffer directly. The acquired images then have suitable size for processing. They can be process faster with higher frame rate at 30 frames per second.

A Memory leak problem is found afterward. The code then is modified such that it can handle memory leak problem. In particular, the memory is release manually. After lane detection and tracking system implemented, the testing device (iPod touch 4th Gen) is too slow to perform the tasks. For this reason, testing device is changed into iphone5. However, the user interface of iPhone5 becomes a problem because iPhone5's screen has different aspect ratio. Hence, the user interface is needed some modifications.

Another problem has been according to founded. Because by default, the Xcode is set to customize a user interface automatically when an application is initialize. The user interface, drawing algorithm, and warning algorithm then needed some modifications to fix this problem. Finally, the system testing with the real environment is shifted. This is because of unsuitable weather. As a result, the system testing delayed.

The system testing results of the proposed warning techniques show that they work well with high accuracy of lane detection and tracking, and driver warning. However, in the scenario of imperfect lane or bad weather condition these proposed warning techniques might fail.

At this stage, the current system uses only feature-based design. In the future the LDWS should be designed in combination of both model-based and feature-based approach. This would result in a system of higher accuracy. It has been found that the size of the road lane affects the driver warning algorithms. Thus, the future system might consider facing a lane size to be a parameter for driver warning for better warning results. In general, the system should warn only unintentionally lane changing. Thus, in the future additional signal such as turning sound or turning signal from the vehicle can be integrated into the system in order to identify this scenario.

Finally, the warning zone and critical zone are separated for future works. The velocity of the vehicle might be taken into account in order to improve warning algorithm. This can be seen in warning by area. If the vehicle travels slower than the preset threshold speed V_{th} , and the driving condition is of warning driving condition, the warning system might be set not to warn the driver. However, if the vehicle travels with the speed greater than V_{th} this will activate the warning system. For the critical driving condition, the warning system should be activated independent of the vehicle's traveling speed.

5.2 Conclusion

The aim of this project is to reduce the road accident rate caused by lane changing. This is done by developing a driver's warning system as a mobile application. This application will warn the driver as they unintentionally change driving lane. The proposed application will be developed on iOS platform and implemented on iOS mobile device. In this project, iPhone5 has been used

The system starts by obtaining a set of images from video taken by the iPhone's camera. These images are processed by appropriate image pre-processing techniques i.e. ROI, image conversion, image filtering, edge detection, and Hough transform for lane detection. The system will determine whether the car is changing lane or not. If the car is changing lane, the system will warn the driver in form of sound and/or messages. For accuracy purposes, the mobile device must be installed on the windshield of the car with the camera facing to the road.

The experiments are set up for testing the system performance. Lane tracking and warning area technique is a warning technique that used in this project. It warning accuracy is higher more than 80% in simulation test.

Currently, the system works well in the clear lane markers roads without any casted shadows in the scene. It is assumed that the car is driven on a clear weather. In the future, the system should be able to perform in another scenario as well.

Bibliography

[1] Mars Lan, Mahas Rofouei, Stefano Soatto, and Majid Sarrafzadeh; "**SmartLDWS: A robust and scalable lane departure waning system for the smartphones**", *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, 2009,* pp.108-113.

[2] Feixiang Ren, Jinsheng Huang, Mutsuhiro Terauchi, Ruyi Jiang, and Reinhard Klette; "**Lane Detection on the iPhone**", *In Proceedings "ArtsIT 2009", LNICST 30, 2009*, pp.198-205.

[3] Parag H. Batavia "**Driver Adaptive Warning Systems**", *The Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania 15213, March, 1998*, pp.15-16.

[4] Shannon Hetrick "**EXAMINATION OF DRIVER LANE CHANGE BEHAVIOR AND THE POTENTIAL EFFECTIVENESS OF WARNING ONSET RULES FOR LANE CHANGE OR "SIDE" CRASH AVOIDANCE SYSTEMS**", *Thesis submitted to the Faculty of the Virginia Polytechnic Institute & State University, March 27, 1997*, pp.43-45.

Appendix A

Source Code

```
//
//  Lane.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import <Foundation/Foundation.h>

@interface Lane : NSObject

@property(nonatomic, assign) float laneWidth;

@property(nonatomic, retain) UIColor *laneColor;

@property(nonatomic, assign) CGPoint startPoint;

@property(nonatomic, assign) CGPoint endPoint;

@end
```

```
//
//  Lane.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "Lane.h"

@implementation Lane

@end
```

```
//
//  LineView.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/21/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import <UIKit/UIKit.h>
#import "Lane.h"

@interface LineView : UIView

@property(nonatomic, strong) Lane *laneRight;

@property(nonatomic, strong) Lane *laneLeft;

@property(nonatomic, strong) Lane *laneCenter;

@property(nonatomic, assign) int horizontalLine;

- (void)UpdateLane:(CGPoint) rightLaneStart With:(CGPoint)
rightLaneEnd And:(CGPoint) leftLaneStart With:(CGPoint) leftLaneEnd
On:(int)center;

@end
```

```
//
//  LineView.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/21/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "LineView.h"

@implementation LineView

- (id)initWithFrame:(CGRect)frame
{
    self.laneRight = [[Lane alloc]init];
    self.laneLeft = [[Lane alloc]init];
    self.laneCenter = [[Lane alloc]init];

    self.laneRight.laneColor = [UIColor colorWithRed:0 green:1.0
blue:0 alpha:0.5];
    self.laneLeft.laneColor = [UIColor colorWithRed:0.2 green:0.2
blue:0.2 alpha:1.0];
    self.laneCenter.laneColor = [UIColor colorWithRed:0.7 green:0.2
blue:0.2 alpha:1.0];

    CGPoint pt1,pt2,pt3,pt4;

    pt1.x = 0;pt1.y = 0;
    pt2.x = 0;pt2.y = 0;
    pt3.x = 0;pt3.y = 0;
    pt4.x = 0;pt4.y = 0;

    self.laneRight.startPoint = pt1;
    self.laneRight.endPoint = pt2;
    self.laneLeft.startPoint = pt3;
    self.laneLeft.endPoint = pt4;

    self = [super initWithFrame:frame];
    self.backgroundColor = [UIColor clearColor];
    if (self) {
        // Initialization code
    }
    return self;
}


// Only override drawRect: if you perform custom drawing.
// An empty implementation adversely affects performance during
animation.


//Draw line
- (void)drawRect:(CGRect)rect {
    [super drawRect:rect];

    CGPoint pt1,pt2;

    float rightLaneM = 0.0, leftLaneM = 0.0;
    float rightLanePoint = 0.0, leftLanePoint = 0.0;
```

```
    //Road area
    CGContextRef context = UIGraphicsGetCurrentContext();

    CGContextSetFillColorWithColor(context ,
 self.laneRight.laneColor.CGColor);

    if (self.laneRight.startPoint.x < self.laneLeft.startPoint.x) {
        if (self.laneLeft.startPoint.x - self.laneRight.startPoint.x
 > 200) {
            pt1.x = 290.0; pt1.y = self.laneLeft.startPoint.y;
            self.laneLeft.startPoint = pt1;
            pt2.x = 190.0; pt2.y = self.laneRight.startPoint.y;
            self.laneRight.startPoint = pt2;
        }

        rightLaneM = (self.laneLeft.startPoint.y -
self.laneRight.endPoint.y)/(self.laneLeft.startPoint.x -
self.laneRight.endPoint.x);
        leftLaneM = (self.laneRight.startPoint.y -
self.laneLeft.endPoint.y)/(self.laneRight.startPoint.x -
self.laneLeft.endPoint.x);

        CGContextMoveToPoint(context, self.laneRight.endPoint.x+44,
self.laneRight.endPoint.y);

        CGContextAddLineToPoint(context,
self.laneLeft.startPoint.x+44, self.laneLeft.startPoint.y);

        CGContextAddLineToPoint(context,
self.laneRight.startPoint.x+44, self.laneRight.startPoint.y);

        CGContextAddLineToPoint(context, self.laneLeft.endPoint.x+44,
self.laneLeft.endPoint.y);

        CGContextAddLineToPoint(context,
self.laneRight.endPoint.x+44, self.laneRight.endPoint.y);
    } else {

        rightLaneM = (self.laneRight.startPoint.y -
self.laneRight.endPoint.y)/(self.laneRight.startPoint.x -
self.laneRight.endPoint.x);
        leftLaneM = (self.laneLeft.startPoint.y -
self.laneLeft.endPoint.y)/(self.laneLeft.startPoint.x -
self.laneLeft.endPoint.x);

        CGContextMoveToPoint(context, self.laneRight.endPoint.x+44,
self.laneRight.endPoint.y);

        CGContextAddLineToPoint(context,
self.laneRight.startPoint.x+44, self.laneRight.startPoint.y);

        CGContextAddLineToPoint(context,
self.laneLeft.startPoint.x+44, self.laneLeft.startPoint.y);

        CGContextAddLineToPoint(context, self.laneLeft.endPoint.x+44,
self.laneLeft.endPoint.y);

        CGContextAddLineToPoint(context,
self.laneRight.endPoint.x+44, self.laneRight.endPoint.y);
```

```
    }

    CGContextFillPath(context);

    rightLanePoint = ((320-
self.laneRight.endPoint.y)/rightLaneM)+self.laneRight.endPoint.x;
    leftLanePoint = ((320-
self.laneLeft.endPoint.y)/leftLaneM)+self.laneLeft.endPoint.x;


//Horizontal////////////////////////////////////////////////////////
/////////////////////////

    //Right triangle
    CGContextSetFillColorWithColor(context ,
self.laneLeft.laneColor.CGColor);

    CGContextMoveToPoint(context, rightLanePoint+44, 310);

    CGContextAddLineToPoint(context, rightLanePoint-5+44, 320);

    CGContextAddLineToPoint(context, rightLanePoint+5+44, 320);

    CGContextAddLineToPoint(context, rightLanePoint+44, 310);

    CGContextFillPath(context);

    //Left triangle
    CGContextSetFillColorWithColor(context ,
self.laneLeft.laneColor.CGColor);

    CGContextMoveToPoint(context, leftLanePoint+44, 310);

    CGContextAddLineToPoint(context, leftLanePoint+5+44, 320);

    CGContextAddLineToPoint(context, leftLanePoint-5+44, 320);

    CGContextAddLineToPoint(context, leftLanePoint+44, 310);

    CGContextFillPath(context);

    //Scale triangle
    CGContextSetFillColorWithColor(context ,
self.laneLeft.laneColor.CGColor);

    CGContextMoveToPoint(context,
(rightLanePoint+leftLanePoint)/2+44, 310);

    CGContextAddLineToPoint(context,
((rightLanePoint+leftLanePoint)/2)-5+44, 320);

    CGContextAddLineToPoint(context,
((rightLanePoint+leftLanePoint)/2)+5+44, 320);

    CGContextAddLineToPoint(context,
(rightLanePoint+leftLanePoint)/2+44, 310);

    CGContextAddLineToPoint(context,
((rightLanePoint+leftLanePoint)/2)-1+44, 270);
```

```
    CGContextAddLineToPoint(context,
((rightLanePoint+leftLanePoint)/2)+1+44, 270);

    CGContextAddLineToPoint(context,
(rightLanePoint+leftLanePoint)/2+44, 310);

    CGContextFillPath(context);

    //Center triangle
    CGContextSetFillColorWithColor(context ,
self.laneCenter.laneColor.CGColor);

    CGContextMoveToPoint(context, 240+44, 310);

    CGContextAddLineToPoint(context, 235+44, 320);

    CGContextAddLineToPoint(context, 245+44, 320);

    CGContextAddLineToPoint(context, 240+44, 310);

    CGContextFillPath(context);


//Horizontal//////////////////////////////////////////////////////
//////////////////////////

    //Scale triangle
    CGContextSetFillColorWithColor(context ,
self.laneLeft.laneColor.CGColor);

    CGContextMoveToPoint(context, 470+44, self.horizontalLine);

    CGContextAddLineToPoint(context, 480+44, self.horizontalLine-5);

    CGContextAddLineToPoint(context, 480+44, self.horizontalLine+5);

    CGContextAddLineToPoint(context, 470+44, self.horizontalLine);

    CGContextAddLineToPoint(context, 400+44, self.horizontalLine-1);

    CGContextAddLineToPoint(context, 400+44, self.horizontalLine+1);

    CGContextAddLineToPoint(context, 470+44, self.horizontalLine);

    CGContextFillPath(context);

    //Center triangle
    CGContextSetFillColorWithColor(context ,
self.laneCenter.laneColor.CGColor);

    CGContextMoveToPoint(context, 470+44, 160);

    CGContextAddLineToPoint(context, 480+44, 155);

    CGContextAddLineToPoint(context, 480+44, 165);

    CGContextAddLineToPoint(context, 470+44, 160);

    CGContextFillPath(context);
```

```
    UIColor *Test = [UIColor blueColor];

    //Test triangle
    CGContextSetFillColorWithColor(context , Test.CGColor);

    CGContextMoveToPoint(context, 80+44, 310);

    CGContextAddLineToPoint(context, 75+44, 320);

    CGContextAddLineToPoint(context, 85+44, 320);

    CGContextAddLineToPoint(context, 80+44, 310);

    CGContextFillPath(context);

    //Test triangle
    CGContextSetFillColorWithColor(context , Test.CGColor);

    CGContextMoveToPoint(context, 400+44, 310);

    CGContextAddLineToPoint(context, 395+44, 320);

    CGContextAddLineToPoint(context, 405+44, 320);

    CGContextAddLineToPoint(context, 400+44 , 310);

    CGContextFillPath(context);
}

- (void)UpdateLane:(CGPoint) rightLaneStart With:(CGPoint)
rightLaneEnd And:(CGPoint) leftLaneStart With:(CGPoint) leftLaneEnd
On:(int)center{

    center += (320-center)/3;

    float rightLaneM = 0.0, leftLaneM = 0.0;
    float rightLanePoint = 0.0, leftLanePoint = 0.0;

    CGPoint pt1,pt2;

    rightLaneM = (rightLaneStart.y -
rightLaneEnd.y)/(rightLaneStart.x - rightLaneEnd.x);
    leftLaneM = (leftLaneStart.y - leftLaneEnd.y)/(leftLaneStart.x -
leftLaneEnd.x);

    rightLanePoint = ((center-
rightLaneEnd.y)/rightLaneM)+rightLaneEnd.x;
    leftLanePoint = ((center-leftLaneEnd.y)/leftLaneM)+leftLaneEnd.x;

    pt1.x = rightLanePoint;pt1.y = center;
    pt2.x = leftLanePoint;pt2.y = center;
    self.horizontalLine = center;

    //Right lane
    self.laneRight.startPoint = pt1;
    self.laneRight.endPoint = rightLaneEnd;

    //Left lane
    self.laneLeft.startPoint = pt2;
    self.laneLeft.endPoint = leftLaneEnd;
```

```
}

@end
```

```
//
//  Message.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import <Foundation/Foundation.h>
#import "Notifier.h"

@interface Message : NSObject

@property (nonatomic, assign) BOOL useMessage;

@property (nonatomic, retain) NSString *text;

@end
```

```
//
//  Message.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "Message.h"

@implementation Message

@end
```

```
//
//  Notifier.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import <Foundation/Foundation.h>

@interface Notifier : NSObject

@end
```

```
//
//  SettingValue.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 4/8/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import <Foundation/Foundation.h>

@interface SettingValue : NSObject

@property (nonatomic, assign) BOOL useMessage;

@property (nonatomic, retain) NSString *text;

@property (nonatomic, assign) BOOL useSound;

@property (nonatomic, assign) float volumn;

@end
```

```
//
//  SettingValue.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 4/8/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "SettingValue.h"

@implementation SettingValue

@end
```

```
//
//  SettingViewController.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/4/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//
@class ViewController;

#import <UIKit/UIKit.h>
#import "Message.h"
#import "Sound.h"
#import "ViewController.h"

@interface SettingViewController : UIViewController
<UITextFieldDelegate>

@property (nonatomic, strong) ViewController *mainViewController;

@property (nonatomic, strong) Message *message;

@property (nonatomic, strong) Sound *sound;

@property (nonatomic, strong) IBOutlet UITextField *textField;

@property (nonatomic, strong) IBOutlet UISlider *soundSlider;

@property (nonatomic, strong) IBOutlet UISwitch *textSwitch;

@property (nonatomic, strong) IBOutlet UISwitch *soundSwitch;

@property (nonatomic, strong) IBOutlet UIButton *done;

- (IBAction)textFieldChanged:(id)sender;

- (IBAction)soundSliderChanged:(id)sender;

- (IBAction)textSwitchChanged:(id)sender;

- (IBAction)soundSwitchChanged:(id)sender;

- (IBAction)doneButtonPressed:(id)sender;

@end
```

```
//
//  SettingViewController.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/4/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "SettingViewController.h"
#import <MediaPlayer/MPMusicPlayerController.h>

@interface SettingViewController ()

@end

@implementation SettingViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle
*)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil
bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    self.message = [[Message alloc]init];
    self.sound = [[Sound alloc]init];

    self.textField.delegate = self;

    self.message.text = self.mainViewController.settingValue.text;
    self.sound.volumn = self.mainViewController.settingValue.volumn;
    self.message.useMessage =
self.mainViewController.settingValue.useMessage;
    self.sound.useSound =
self.mainViewController.settingValue.useSound;

    self.textField.text = self.message.text;
    self.soundSlider.value = self.sound.volumn;
    self.textSwitch.on = self.message.useMessage;
    self.soundSwitch.on = self.sound.useSound;
}

- (BOOL)shouldAutorotate
{
    return NO;
}

-
(BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)
interfaceOrientation {
```

```
    return (interfaceOrientation ==
UIInterfaceOrientationLandscapeLeft);
}

-(NSUInteger)supportedInterfaceOrientations {
    return UIInterfaceOrientationMaskLandscapeLeft;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)textFieldChanged:(id)sender{
    self.message.text = self.textField.text;
    self.mainViewController.settingValue.text = self.textField.text;
}

- (void)soundSliderChanged:(id)sender{
    self.sound.volumn = self.soundSlider.value;
    [[MPMusicPlayerController
applicationMusicPlayer]setVolume:self.sound.volumn];
    self.mainViewController.settingValue.volumn =
self.soundSlider.value;
}

- (void)textSwitchChanged:(id)sender{
    self.message.useMessage = self.textSwitch.isOn;
    self.mainViewController.settingValue.useMessage =
self.textSwitch.isOn;
}

- (void)soundSwitchChanged:(id)sender{
    self.sound.useSound = self.soundSwitch.isOn;
    self.mainViewController.settingValue.useSound =
self.soundSwitch.isOn;
}

- (void)doneButtonPressed:(id)sender{
    [self dismissViewControllerAnimated:YES completion:nil];
}

-(BOOL) textFieldShouldReturn:(UITextField *)textField {
    [self.textField resignFirstResponder];
    return YES;
}

@end
```

```
//
//  Sound.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import <Foundation/Foundation.h>
#import "Notifier.h"

@interface Sound : NSObject

@property (nonatomic, assign) BOOL useSound;

@property (nonatomic, assign) float volumn;

@end
```

```
//
//  Sound.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 2/1/13.
//  Copyright (c) 2013 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "Sound.h"

@implementation Sound

@end
```

```
//
//  UIImage+OpenCV.h
//  OpenCVClient
//
//  Created by Robin Summerhill on 02/09/2011.
//  Copyright 2011 Aptogo Limited. All rights reserved.
//
//  Permission is given to use this source code file without charge
in any
//  project, commercial or otherwise, entirely at your risk, with the
condition
//  that any redistribution (in part or whole) of source code must
retain
//  this copyright and permission notice. Attribution in compiled
projects is
//  appreciated but not required.
//

#import <UIKit/UIKit.h>

@interface UIImage (UIImage_OpenCV)

+(UIImage *)imageWithCVMat:(const cv::Mat&)cvMat;

-(id)initWithCVMat:(const cv::Mat&)cvMat;

@property(nonatomic, readonly) cv::Mat CVMat;

@end
```

```
//
//  UIImage+OpenCV.mm
//  OpenCVClient
//
//  Created by Robin Summerhill on 02/09/2011.
//  Copyright 2011 Aptogo Limited. All rights reserved.
//
//  Permission is given to use this source code file without charge
in any
//  project, commercial or otherwise, entirely at your risk, with the
condition
//  that any redistribution (in part or whole) of source code must
retain
//  this copyright and permission notice. Attribution in compiled
projects is
//  appreciated but not required.
//

#import "UIImage+OpenCV.h"

static void ProviderReleaseDataNOP(void *info, const void *data,
size_t size)
{
    // Do not release memory
    return;
}

@implementation UIImage (UIImage_OpenCV)

-(cv::Mat)CVMat
{

    CGColorSpaceRef colorSpace = CGImageGetColorSpace(self.CGImage);
    CGFloat cols = self.size.width;
    CGFloat rows = self.size.height;

    cv::Mat cvMat(rows, cols, CV_8UC4); // 8 bits per component, 4
channels

    CGContextRef contextRef = CGBitmapContextCreate(cvMat.data,
// Pointer to backing data
                                                    cols,
// Width of bitmap
                                                    rows,
// Height of bitmap
                                                    8,
// Bits per component
                                                    cvMat.step[0],
// Bytes per row
                                                    colorSpace,
// Colorspace

kCGImageAlphaNoneSkipLast |

kCGBitmapByteOrderDefault); // Bitmap info flags

    CGContextDrawImage(contextRef, CGRectMake(0, 0, cols, rows),
self.CGImage);
    CGContextRelease(contextRef);
    return cvMat;
```

```
}

+ (UIImage *)imageWithCVMat:(const cv::Mat&)cvMat
{
    return [[[UIImage alloc] initWithCVMat:cvMat] autorelease];
}

- (id)initWithCVMat:(const cv::Mat&)cvMat
{
    NSData *data = [NSData dataWithBytes:cvMat.data
length:cvMat.elemSize() * cvMat.total()];

    CGColorSpaceRef colorSpace;

    if (cvMat.elemSize() == 1)
    {
        colorSpace = CGColorSpaceCreateDeviceGray();
    }
    else
    {
        colorSpace = CGColorSpaceCreateDeviceRGB();
    }

    CGDataProviderRef provider =
CGDataProviderCreateWithCFData((CFDataRef)data);

    CGImageRef imageRef = CGImageCreate(cvMat.cols,
// Width
                                        cvMat.rows,
// Height
                                        8,
// Bits per component
                                        8 * cvMat.elemSize(),
// Bits per pixel
                                        cvMat.step[0],
// Bytes per row
                                        colorSpace,
// Colorspace
                                        kCGImageAlphaNone |
kCGBitmapByteOrderDefault,  // Bitmap info flags
                                        provider,
// CGDataProviderRef
                                        NULL,
// Decode
                                        false,
// Should interpolate
                                        kCGRenderingIntentDefault);
// Intent

    self = [self initWithCGImage:imageRef];
    CGImageRelease(imageRef);
    CGDataProviderRelease(provider);
    CGColorSpaceRelease(colorSpace);

    return self;
}

@end
```

```
//
//  ViewController.h
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 11/8/12.
//  Copyright (c) 2012 Manutsanant Subtechitmanee. All rights
reserved.
//

@class AVCaptureStillImageOutput;
@class AVCaptureVideoDataOutput;
@class SettingViewController;
@class SettingValue;

#import <UIKit/UIKit.h>
#import <AVFoundation/AVFoundation.h>
#import <CoreLocation/CoreLocation.h>
#import "LineView.h"
#import "SettingViewController.h"
#import "SettingValue.h"

@interface ViewController : UIViewController
<AVCaptureVideoDataOutputSampleBufferDelegate,CLLocationManagerDelega
te>

@property(nonatomic, retain) IBOutlet UIImageView *vImage;

@property(nonatomic, retain) IBOutlet UIView *vImagePreview;

@property(nonatomic, retain) IBOutlet UIView *warningView;

@property(nonatomic, retain) IBOutlet UIView *lineViewBG;

@property(nonatomic, retain) IBOutlet UIButton *settingButton;

@property(nonatomic, retain) IBOutlet UILabel *warningLabel;

@property(nonatomic, retain) IBOutlet UILabel *speedLabel;

@property(nonatomic, retain) IBOutlet UILabel *compass;

@property(nonatomic, retain) IBOutlet UILabel *compassDegree;

@property(nonatomic, retain) AVCaptureStillImageOutput
*stillImageOutput;

@property(nonatomic, retain) AVCaptureVideoDataOutput *imageOutput;

@property(nonatomic, retain) CLLocationManager *locationManager;

@property(nonatomic, retain) LineView *lineView;

@property(nonatomic, strong) SettingValue *settingValue;

@property(nonatomic, assign) CGPoint laneRightStart;

@property(nonatomic, assign) CGPoint laneRightEnd;

@property(nonatomic, assign) CGPoint laneLeftStart;
```

```
@property(nonatomic, assign) CGPoint laneLeftEnd;
@property(nonatomic, assign) int speed;
@property(nonatomic, assign) int roiline;
@property(nonatomic, assign) int frameCounter;
@property(nonatomic, assign) BOOL warningSwitch;
- (IBAction)settingButtonPressed:(id)sender;
@end
```

```
//
//  ViewController.m
//  LDWS
//
//  Created by Manutsanant Subtechitmanee on 11/8/12.
//  Copyright (c) 2012 Manutsanant Subtechitmanee. All rights
reserved.
//

#import "ViewController.h"
#import "UIImage+OpenCV.h"
#import "LineView.h"
#import <AudioToolbox/AudioToolbox.h>


@interface ViewController ()
@property(nonatomic, strong) AVCaptureSession *session;
@property(nonatomic, strong) dispatch_queue_t queue;
@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from
a nib.

    // Line view add
    self.lineView = [[LineView alloc] initWithFrame:CGRectMake(0, 0,
self.view.bounds.size.height, self.view.bounds.size.width)];
    [self.lineViewBG addSubview:self.lineView];
    [self.lineView setUserInteractionEnabled:NO];

    // CLLocation init
    self.locationManager = [[CLLocationManager alloc]init];
    self.locationManager.delegate = self;
    self.locationManager.desiredAccuracy= kCLLocationAccuracyBest;
    [self.locationManager startUpdatingLocation];
    [self.locationManager startUpdatingHeading];

    // Set Transform
    self.vImagePreview.transform = CGAffineTransformMakeRotation
(M PI/2);

    // SettingValue init
    self.settingValue = [[SettingValue alloc]init];
    self.settingValue.text = @"Lane Changing!!";
    self.settingValue.volumn = 0.5;
    self.settingValue.useMessage = true;
    self.settingValue.useSound = true;
    self.warningView.hidden = true;

    //set device speed
    self.speed = 0;
    self.speedLabel.text = @"n/a";

    //Set warning a times
    self.warningSwitch = true;
```

```
    //Init start lane departure
    CGPoint p1,p2,p3,p4;
    p1.x = 280; p1.y = 180;
    p2.x = 380; p2.y = 320;
    p3.x = 200; p3.y = 180;
    p4.x = 100; p4.y = 320;
    self.laneRightStart = p1;
    self.laneRightEnd = p2;
    self.laneLeftStart = p3;
    self.laneLeftEnd = p4;

    //Set area of interest line
    self.roiline = 180;
    self.frameCounter = 30;

    //For Testing
    self.vImage.hidden = true;
}

-(void) viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];

    self.session = [[AVCaptureSession alloc] init];
    self.session.sessionPreset = AVCaptureSessionPresetMedium;

    AVCaptureVideoPreviewLayer *captureVideoPreviewLayer =
[[AVCaptureVideoPreviewLayer alloc] initWithSession:self.session];

    captureVideoPreviewLayer.frame = CGRectMake(0, 0,
self.view.bounds.size.height+40, self.view.bounds.size.width);

    [self.vImagePreview.layer addSublayer:captureVideoPreviewLayer];

    //Create and Configure the Device and Device Input

    AVCaptureDevice *device = [AVCaptureDevice
defaultDeviceWithMediaType:AVMediaTypeVideo];

    NSError *error = nil;

    AVCaptureDeviceInput *input = [AVCaptureDeviceInput
deviceInputWithDevice:device error:&error];

    [self.session addInput:input];

    [self.session startRunning];

    //Create and Configure the Data Output

    // Create a VideoDataOutput and add it to the session
    AVCaptureVideoDataOutput *imageOutput =
[[AVCaptureVideoDataOutput alloc] init];
    [self.session addOutput:imageOutput];

    // Configure output
    self.queue = dispatch queue create("myQueue", NULL);

    [imageOutput setSampleBufferDelegate:self
                                   queue:self.queue];
```

```
        // Specify the pixel format
        imageOutput.videoSettings =
        [NSDictionary dictionaryWithObject:
         [NSNumber numberWithInt:kCVPixelFormatType_32BGRA]

  forKey:(id)kCVPixelBufferPixelFormatTypeKey];

        // Frame rate
        AVCaptureConnection *conn = [imageOutput
  connectionWithMediaType:AVMediaTypeVideo];
        if (conn.isVideoMinFrameDurationSupported)
            conn.videoMinFrameDuration = CMTimeMake(1, 30);
        //CMTimeShow(conn.videoMinFrameDuration);
  }

  - (void)didReceiveMemoryWarning
  {
        [super didReceiveMemoryWarning];
        // Dispose of any resources that can be recreated.
  }

  // Setting button pressed
  - (void)settingButtonPressed:(id)sender
  {
        UIStoryboard *storyboard = [UIStoryboard
  storyboardWithName:@"MainStoryboard"
                                                        bundle:nil];
        SettingViewController *settingView = [storyboard
  instantiateViewControllerWithIdentifier:@"SettingViewController"];
        settingView.mainViewController = self;
        [self presentViewController:settingView animated:YES
  completion:nil];
  }

  // Delegate routine that is called when a location change
  - (void)locationManager:(CLLocationManager *)manager
        didUpdateToLocation:(CLLocation *)newLocation
               fromLocation:(CLLocation *)oldLocation
  {
        //simply get the speed provided by the phone from newLocation

        if (newLocation.speed <= 0) {
            self.speed = 0;
        } else {
            self.speed = newLocation.speed*3.6;//transform speed into KmH
        }
        dispatch_async(dispatch_get_main_queue(), ^{
            self.speedLabel.text = [NSString stringWithFormat:@"%d",
  self.speed];
        });
  }

  //  Delegate routine that is called when a Heading change
  - (void)locationManager:(CLLocationManager *)manager
  didUpdateHeading:(CLHeading *)newHeading
  {
        float heading = newHeading.magneticHeading;
        if(heading < 22.5){
            self.compass.text = @"N";
```

```
    }else if (heading < 67.5){
        self.compass.text = @"NE";
    }else if (heading < 112.5){
        self.compass.text = @"E";
    }else if (heading < 157.5){
        self.compass.text = @"SE";
    }else if (heading < 202.5){
        self.compass.text = @"S";
    }else if (heading < 247.5){
        self.compass.text = @"SW";
    }else if (heading < 292.5){
        self.compass.text = @"W";
    }else if (heading < 332.5){
        self.compass.text = @"NW";
    }else{
        self.compass.text = @"N";
    }
    self.compassDegree.text = [NSString
stringWithFormat:@"%d°",(int)heading];;
}

// Delegate routine that is called when a sample buffer was written
- (void)captureOutput:(AVCaptureOutput *)captureOutput
didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
       fromConnection:(AVCaptureConnection *)connection
{
    @autoreleasepool {
        // Create a UIImage from the sample buffer data
        if(!self.view.isHidden){

            UIImage* image = [self
imageFromSampleBuffer:sampleBuffer];//get UIImage from image buffer
            image = [self scaleAndRotateImage:image];//Rotate image
            image = [self getSubImageFrom:image
WithRect:CGRectMake(0,180,480,360)];//Crop image
            image = [self imageWithImage:image
scaledToSize:CGSizeMake(120, 45)];//Scale image

            image = [self processImage:image];

            //Show image
            dispatch_async(dispatch_get_main_queue(), ^{
                self.vImage.image = image;
                [self.lineView setNeedsDisplay];
            });

        }
    }
}

// Create a UIImage from sample buffer data
- (UIImage *) imageFromSampleBuffer:(CMSampleBufferRef) sampleBuffer
{
    // Get a CMSampleBuffer's Core Video image buffer for the media
data
    CVImageBufferRef imageBuffer =
CMSampleBufferGetImageBuffer(sampleBuffer);
    // Lock the base address of the pixel buffer
    CVPixelBufferLockBaseAddress(imageBuffer, 0);
```

```
    // Get the number of bytes per row for the pixel buffer
    void *baseAddress = CVPixelBufferGetBaseAddress(imageBuffer);

    // Get the number of bytes per row for the pixel buffer
    size_t bytesPerRow = CVPixelBufferGetBytesPerRow(imageBuffer);
    // Get the pixel buffer width and height
    size_t width = CVPixelBufferGetWidth(imageBuffer);
    size_t height = CVPixelBufferGetHeight(imageBuffer);

    // Create a device-dependent RGB color space
    CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();

    // Create a bitmap graphics context with the sample buffer data
    CGContextRef context = CGBitmapContextCreate(baseAddress, width,
height, 8,
                                                 bytesPerRow,
colorSpace, kCGBitmapByteOrder32Little |
kCGImageAlphaPremultipliedFirst);
    // Create a Quartz image from the pixel data in the bitmap
graphics context
    CGImageRef quartzImage = CGBitmapContextCreateImage(context);
    // Unlock the pixel buffer
    CVPixelBufferUnlockBaseAddress(imageBuffer,0);

    // Free up the context and color space
    CGContextRelease(context);
    CGColorSpaceRelease(colorSpace);

    // Create an image object from the Quartz image
    UIImage *image = [UIImage imageWithCGImage:quartzImage];

    // Release the Quartz image
    CGImageRelease(quartzImage);

    return (image);
}

//Manage an
Image////////////////////////////////////////////////////////////////
//////////////////////

// Resize an image
- (UIImage *)imageWithImage:(UIImage *)image
scaledToSize:(CGSize)newSize
{
    //UIGraphicsBeginImageContext(newSize);
    UIGraphicsBeginImageContextWithOptions(newSize, NO, 0.0);
    [image drawInRect:CGRectMake(0, 0, newSize.width,
newSize.height)];
    UIImage *newImage = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();
    return newImage;
}

// Rotate a UIImage
- (UIImage *)scaleAndRotateImage:(UIImage *)image
{
    CGImageRef imgRef = image.CGImage;

    CGFloat width = CGImageGetWidth(imgRef);
```

```
    CGFloat height = CGImageGetHeight(imgRef);

    CGAffineTransform transform = CGAffineTransformIdentity;

    CGRect bounds = CGRectMake(0, -height/2, width, height*2);

    transform = CGAffineTransformMakeTranslation(width, 0.0);
    transform = CGAffineTransformScale(transform, -1.0, 1.0);

    UIGraphicsBeginImageContext(bounds.size);
    CGContextRef context = UIGraphicsGetCurrentContext();
    CGContextConcatCTM(context, transform);
    CGContextDrawImage(context, CGRectMake(0, -height/2, width,
height*2), imgRef);
    UIImage *imageCopy = UIGraphicsGetImageFromCurrentImageContext();
    UIGraphicsEndImageContext();

    return imageCopy;
}

// Crop image
- (UIImage *)getSubImageFrom: (UIImage*) img WithRect: (CGRect) rect
{

    UIGraphicsBeginImageContext(rect.size);
    CGContextRef context = UIGraphicsGetCurrentContext();

    // translated rectangle for drawing sub image
    CGRect drawRect = CGRectMake(-rect.origin.x, -rect.origin.y,
img.size.width, img.size.height);

    // clip to the bounds of the image context
    // not strictly necessary as it will get clipped anyway?
    CGContextClipToRect(context, CGRectMake(0, 0, rect.size.width,
rect.size.height));

    // draw image
    [img drawInRect:drawRect];

    // grab image
    UIImage* subImage = UIGraphicsGetImageFromCurrentImageContext();

    UIGraphicsEndImageContext();

    return subImage;
}

// Find ROI using vanishing point
- (int)findLine:(cv::vector<cv::vector<int>>)linesP
{
    float m1 = 0, m2 = 0, y = 0;

    CGPoint p1,p2,p3,p4;

    p1.x = linesP[0][0]; p1.y = linesP[0][1];
    p2.x = linesP[0][2]; p2.y = linesP[0][3];

    p3.x = linesP[1][0]; p3.y = linesP[1][1];
    p4.x = linesP[1][2]; p4.y = linesP[1][3];
```

```
    m1 = (p2.y - p1.y)/(p2.x - p1.x);
    m2 = (p4.y - p3.y)/(p4.x - p3.x);

    y = (m2/(m2-m1))*(p1.y-(m1*p1.x))+(m1/(m1-m2))*(p3.y-(m2*p3.x));

    int result = y;
    result = (result*4)+180;

    //NSLog(@"aoi %d",result);

    return result;
}

//Image
Processing////////////////////////////////////////////////////////////
////////////////////////////

//Lane Departure//////////////////////////////////////

//OpenCV
- (UIImage *)processImage:(UIImage *)image
{

    cv::Mat cvMat = [image CVMat];

    //2Dfilter
    cv::blur(cvMat, cvMat, cv::Size(3,3));

    //Grayscale
    cv::cvtColor(cvMat, cvMat, cv::COLOR_RGB2GRAY);

    //Edge detection
    cv::Sobel(cvMat, cvMat, cvMat.depth(), 1, 1);

    //Thresholding
    cv::threshold(cvMat, cvMat, 127, 255,
CV THRESH BINARY|CV THRESH OTSU);

    //Morphology
    cv::morphologyEx(cvMat, cvMat, CV MOP OPEN, cv::KERNEL GENERAL,
cv::Point(-1,-1),100);

    //Hough Tranfroms
    cvMat = [self findLane:cvMat];

    image = [UIImage imageWithCVMat:cvMat];

    return image;
}

//Hough Tranforms
- (cv::Mat)findLane:(cv::Mat)cvMat
{
    BOOL b1 = false,b2 = false;

    cv::Point pt1 = cv::Point(0,0);
    cv::Point pt2 = cv::Point(0,0);
    cv::Point pt3 = cv::Point(0,0);
    cv::Point pt4 = cv::Point(0,0);
```

```
    cv::vector<cv::Vec2f> lines;

    int rightLaneMark = -1;
    int leftLaneMark = -1;

    for (int i = 40; i >= 20; i -= 2) {
        cv::HoughLines(cvMat, lines, 1, CV_PI/180, i);
        if(lines.size() > 0){
            //sort line
            [self sortLine:lines];
            //right
            if (!b1){
                rightLaneMark = [self findRightLane:lines];
                if (rightLaneMark >= 0) {
                    float rho = lines[rightLaneMark][0],theta =
lines[rightLaneMark][1];
                    double a = cos(theta), b = sin(theta);
                    double x0 = a*rho, y0 = b*rho;
                    pt1 = cv::Point(x0 + 200*(-b),y0 + 200*(a));
                    pt2 = cv::Point(x0 - 200*(-b),y0 - 200*(a));
                    //draw right lane
                    if (theta <= 168*(CV_PI/180) and theta >=
110*(CV_PI/180)) {
                        cv::line(cvMat, pt1, pt2, cvScalar(125));
                        b1 = true;
                    }
                }
            }
            //left
            if (!b2) {
                leftLaneMark = [self findLeftLane:lines];
                if (leftLaneMark >= 0) {
                    float rho = lines[leftLaneMark][0],theta =
lines[leftLaneMark][1];
                    double a = cos(theta), b = sin(theta);
                    double x0 = a*rho, y0 = b*rho;
                    pt3 = cv::Point(x0 + 200*(-b),y0 + 200*(a));
                    pt4 = cv::Point(x0 - 200*(-b),y0 - 200*(a));
                    //draw left lane
                    if (theta >= 12*(CV PI/180) and theta <=
70*(CV PI/180)) {
                        cv::line(cvMat, pt3, pt4, cvScalar(125));
                        b2 = true;
                    }
                }
            }
            //break
            if (b1 and b2){
                break;
            }
        }
    }


    cv::vector<cv::vector<int>> linesP;
    cv::vector<int> pos1, pos2;

    if (self.frameCounter > 5 and b1 and b2) {

        pos1.push_back(pt1.x);
```

```
        pos1.push_back(pt1.y);
        pos1.push_back(pt2.x);
        pos1.push_back(pt2.y);
        linesP.push_back(pos1);

        pos2.push_back(pt3.x);
        pos2.push_back(pt3.y);
        pos2.push_back(pt4.x);
        pos2.push_back(pt4.y);
        linesP.push_back(pos2);

        pos1.clear();
        pos2.clear();

        self.roiline = [self findLine:linesP];
        self.frameCounter = 0;
    }
    self.frameCounter++;

    //Warning Part
    CGPoint p1,p2,p3,p4;

    if (b1 or b2) {
        p1.x = pt1.x; p1.y = pt1.y;
        p2.x = pt2.x; p2.y = pt2.y;
        p3.x = pt3.x; p3.y = pt3.y;
        p4.x = pt4.x; p4.y = pt4.y;
        [self Warn:b1 with:p1 And:p2 Warn:b2 with:p3 And:p4
withCarSpeed:self.speed];

    }else{
        dispatch_async(dispatch_get_main_queue(), ^{
            [self UnShowWarnMessage];
        });
    }

    //draw line on uiview
    if(b1 and b2){
        //right
        p1.x = pt1.x * 4; p1.y = (pt1.y * 4) + 180;
        p2.x = pt2.x * 4; p2.y = (pt2.y * 4) + 180;

        //left
        p3.x = pt3.x * 4; p3.y = (pt3.y * 4) + 180;
        p4.x = pt4.x * 4; p4.y = (pt4.y * 4) + 180;

        [self.lineView UpdateLane:p1 With:p2 And:p4 With:p3
On:self.roiline];
    }
    else if (b1){
        //right
        p1.x = pt1.x * 4; p1.y = (pt1.y * 4) + 180;
        p2.x = pt2.x * 4; p2.y = (pt2.y * 4) + 180;

        //left
        p3 = self.laneLeftStart;
        p4 = self.laneLeftEnd;

        [self.lineView UpdateLane:p1 With:p2 And:p4 With:p3
On:self.roiline];
```

```
    }
    else if (b2){
        //right
        p1 = self.laneRightStart;
        p2 = self.laneRightEnd;

        //left
        p3.x = pt3.x * 4; p3.y = (pt3.y * 4) + 180;
        p4.x = pt4.x * 4; p4.y = (pt4.y * 4) + 180;

        [self.lineView UpdateLane:p1 With:p2 And:p4 With:p3
On:self.roiline];
    }

    if(b1){
        self.laneRightStart = p1;
        self.laneRightEnd = p2;
    }
    if(b2){
        self.laneLeftStart = p3;
        self.laneLeftEnd = p4;
    }

    //draw warning line
    cv::line(cvMat, cv::Point(30,0), cv::Point(30,45),
cvScalar(255));
    cv::line(cvMat, cv::Point(90,0), cv::Point(90,45),
cvScalar(255));

    return cvMat;
}

- (int)findRightLane:(cv::vector<cv::Vec2f>) lines
{
    int index = -1;
    for( size_t i = lines.size(); i>0; i--)
    {
        if (lines[i][1] <= 168*(CV_PI/180) and lines[i][1] >=
110*(CV_PI/180)) {
            index = i;
            break;
        }
    }
    return index;
}

- (int)findLeftLane:(cv::vector<cv::Vec2f>) lines
{
    int index = -1;
    for( size_t i = 0; i < lines.size(); i++)
    {
        if (lines[i][1] >= 12*(CV_PI/180) and lines[i][1] <=
70*(CV_PI/180)) {
            index = i;
            break;
        }
    }
    return index;
}
```

```
- (void)sortLine:(cv::vector<cv::Vec2f>) lines
{
    cv::Vec2f temp;
    int i,j;
    int min;

    for( j = 0; j < lines.size()-1; j++)
    {
        min = j;
        for( i = j+1; i < lines.size(); i++)
        {
            if (lines[i][1] < lines[min][1]) {
                min = i;
            }
        }
        if (min != j) {
            temp = lines[j];
            lines[j] = lines[min];
            lines[min] = temp;
        }
    }
}

//Warning system//////////////////////////////////////

//warn
- (void)Warn:(BOOL) b1 with:(CGPoint) rightA And:(CGPoint) rightB
Warn:(BOOL) b2 with:(CGPoint) leftA And:(CGPoint) leftB
withCarSpeed:(int)speed
{
    float rightM = 0.0,leftM = 0.0;
    float rightLanePoint= 0.0,leftLanePoint = 0.0;

    rightM = (rightA.y - rightB.y)/(rightA.x - rightB.x);
    leftM = (leftA.y - leftB.y)/(leftA.x - leftB.x);

    rightLanePoint = ((45-rightB.y)/rightM)+rightB.x;
    leftLanePoint = ((45-leftB.y)/leftM)+leftB.x;

    //right
    if (b1 and ((rightLanePoint <= 105 and speed < 60) or
(rightLanePoint <= 105 and speed >= 60))){
        if (self.warningSwitch) {
            if (self.settingValue.useMessage) {
                dispatch async(dispatch get main queue(), ^{
                    [self ShowWarnMessage];
                });
            }
            if (self.settingValue.useSound) {
                [self PlayWarnSound];
            }
            sleep(1);
        }
    }
    //left
    else if (b2 and ((leftLanePoint >= 15 and speed < 60) or
(leftLanePoint >= 15 and speed >= 60))){
        if (self.warningSwitch) {
            if (self.settingValue.useMessage) {
                dispatch_async(dispatch_get_main_queue(), ^{
```

```
                    [self ShowWarnMessage];
                });
            }
            if (self.settingValue.useSound) {
                [self PlayWarnSound];
            }
            sleep(1);
        }
    }
    else{
        dispatch_async(dispatch_get_main_queue(), ^{
            [self UnShowWarnMessage];
        });
    }
}
//Show Msg
-(void)ShowWarnMessage
{
    self.warningView.hidden = false;
    self.lineView.hidden = true;
    self.warningLabel.text = self.settingValue.text;
    self.warningSwitch = false;
}

//Hidden Msg
-(void)UnShowWarnMessage
{
    self.warningView.hidden = true;
    self.lineView.hidden = false;
    self.warningSwitch = true;
}

//Play sound
-(void)PlayWarnSound
{
    NSURL* musicFile = [NSURL fileURLWithPath:[[NSBundle mainBundle]

pathForResource:@"beep"
                                                ofType:@"mp3"]];
    SystemSoundID musicFileSound;

AudioServicesCreateSystemSoundID((CFURLRef)CFBridgingRetain(musicFile
), &musicFileSound);
    AudioServicesPlaySystemSound(musicFileSound);
}

@end
```

Appendix B

Project Management

Project management is an essential tool to manage a project. It is a tool used to achieve the goals by planning, organizing, and controlling resources. This project follows such an approach. The activities of the project have already been planned at the beginning period. The plan is given as to identify objectives, procedures, and schedules to achieve the goal. It has been done to eliminate or reduce uncertainty, improve performance, and prepare for follow up and controlling tasks. The main tasks to be achieved are identify goals and objectives, gathering system requirements, analysis and design the system, and implement and test the system.

In the first semester, goals and objectives was discussed and specified. All of the requirements were gathering to scope the project. Mock-up was designed to show the systems interface and flow of activities. This task was done on late of Aug 2012.

The team members then studied basic knowledge and improved concerning skills. This included digital Image processing and basic iOS development. Next step was hardware preparation. Apple account was register for a developer team member and testing device was registered for implementation and test. Example video was recorded. All of these works were done on late of Sep 2012.

Next, system analysis and design had been started. During this period, use case diagram, Class diagram and Package diagram was documented. After that, the system implementation was started. A subsystem "lane tracking" was implemented and finish on late of Dec 2012 as shown on Figure 1.

In the second semester, Implementation process proceeded until after New Year. A subsystem called warning system was implementing. Afterward the system was then improve and finished on late of Apr 2013

Finally, system was tested. The results were recorded and summarized for 10 days. System artifact was specified and final presentation was prepared. All of work in this semester was finished on 25 May 2013 as shown in Figure 2.

ID	Task Name	Duration	Start	Finish
1	**Scope**	**15 days**	**Mon 8/6/12**	**Fri 8/24/12**
2	Project Background	5 days	Mon 8/6/12	Fri 8/10/12
3	Gathering Requirement	5 days	Mon 8/13/12	Fri 8/17/12
4	Mock-up Design	5 days	Mon 8/20/12	Fri 8/24/12
5	**Learning Image Processing**	**35 days**	**Wed 8/15/12**	**Tue 10/2/12**
6	Spatial and Frequency domain method	20 days	Wed 8/15/12	Tue 9/11/12
7	Image Segmentation and Morphology	15 days	Wed 9/12/12	Tue 10/2/12
8	**Tools**	**4 days**	**Mon 8/27/12**	**Thu 8/30/12**
9	Register Testing Devices	1 day	Mon 8/27/12	Mon 8/27/12
10	Record Testing Video	3 days	Tue 8/28/12	Thu 8/30/12
11	Preparing to Presentation	11 days	Mon 10/1/12	Mon 10/15/12
12	**Learning iOS**	**25 days**	**Mon 8/27/12**	**Fri 9/28/12**
13	**Practice Implementation**	**10 days**	**Mon 8/27/12**	**Fri 9/7/12**
14	Table View Application	5 days	Mon 8/27/12	Fri 8/31/12
15	To Do List Application	5 days	Mon 9/3/12	Fri 9/7/12
16	Include Image Processing Library	10 days	Mon 9/10/12	Fri 9/21/12
17	Introduction to iOS	5 days	Mon 9/24/12	Fri 9/28/12
18	Study iOS Library and Framework	5 days	Mon 9/24/12	Fri 9/28/12
19	**Analysis & Design**	**4 days**	**Mon 10/22/12**	**Thu 10/25/12**
20	Use Case Diagram	1 day	Mon 10/22/12	Mon 10/22/12
21	Activity Diagram	1 day	Mon 10/22/12	Mon 10/22/12
22	Class Diagram	1 day	Tue 10/23/12	Tue 10/23/12
23	Package Diagram	1 day	Thu 10/25/12	Thu 10/25/12
24	Implementation	35 days	Mon 10/29/12	Fri 12/14/12
25	Testing	17 days	Mon 11/26/12	Tue 12/18/12
26	Documentation	18 days	Mon 11/26/12	Wed 12/19/12
27	Preparing to Presentation	6 days	Wed 12/12/12	Wed 12/19/12
28	Final Presentation	1 day	Thu 12/20/12	Thu 12/20/12

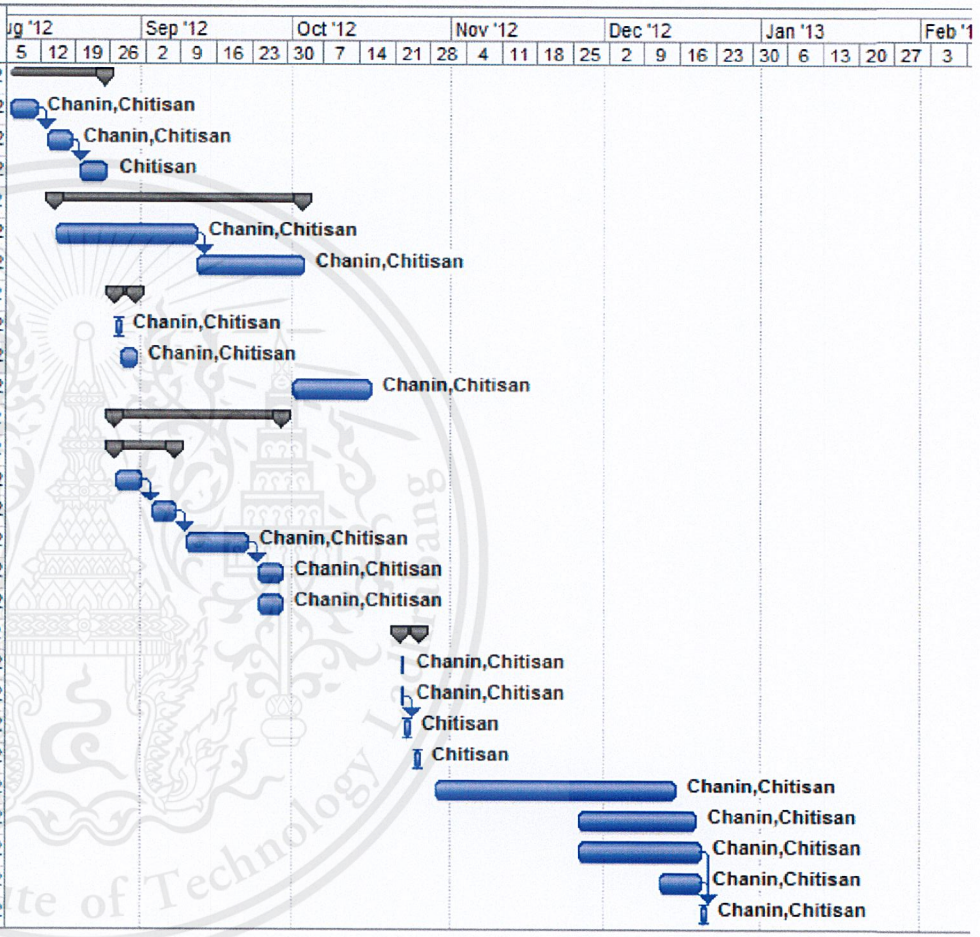


Figure B.1 Project plan 1st semester

ID	Task Name	Duration	Start	Finish	Resource Names
1	Study research	30 days	Mon 1/7/13	Fri 2/15/13	Chanin,Chitisan
2	**Continue Implementation**	**79 days**	**Mon 1/7/13**	**Wed 4/24/13**	
3	**Warning System**	**30 days**	**Mon 1/7/13**	**Fri 2/15/13**	**Chanin,Chitisan**
4	Sound warning system	30 days	Mon 1/7/13	Fri 2/15/13	
5	Message warning system	30 days	Mon 1/7/13	Fri 2/15/13	
6	**Improve system**	**30 days**	**Mon 2/18/13**	**Fri 3/29/13**	**Chanin,Chitisan**
7	Improve lane detection accuracy	30 days	Mon 2/18/13	Fri 3/29/13	
8	Lane detection algorithms	30 days	Mon 2/18/13	Fri 3/29/13	
9	Improve system interface	14 days	Mon 4/1/13	Wed 4/17/13	Chanin,Chitisan
10	Region of interest identifier system	14 days	Mon 4/8/13	Wed 4/24/13	
11	**Testing**	**10 days**	**Mon 4/22/13**	**Fri 5/3/13**	**Chitisan**
12	Integrated test	5 days	Mon 4/22/13	Fri 4/26/13	
13	System test	5 days	Mon 4/29/13	Fri 5/3/13	
14	Documentation	20 days	Tue 4/30/13	Sat 5/25/13	Chanin
15	Prepairing to Presentation	15 days	Tue 5/7/13	Sat 5/25/13	Chanin,Chitisan

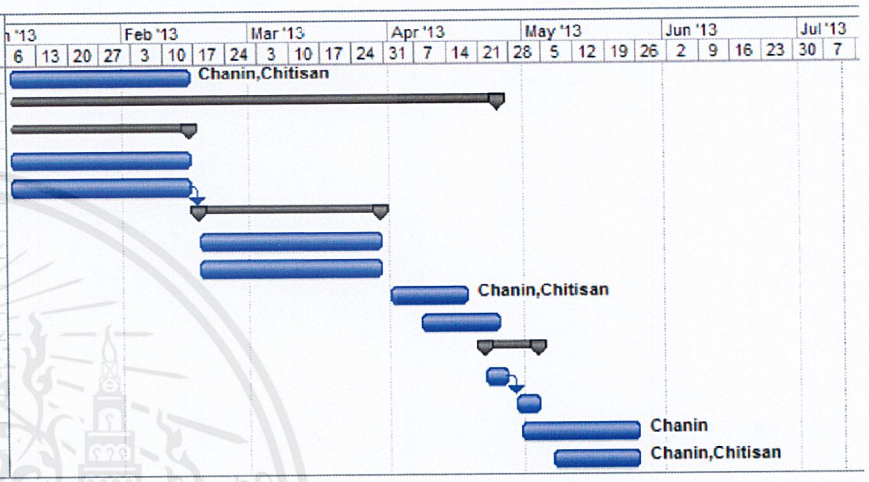


Figure B.2 Project plan 2nd semester