University of South Florida
Scholar Commons

Graduate Theses and Dissertations | Graduate School

January 2013

A Study on Lane-Change Recognition Using Support Vector Machine

Weiping Deng
University of South Florida, wdeng@mail.usf.edu

Follow this and additional works at: http://scholarcommons.usf.edu/etd

Part of the Civil Engineering Commons

A Study on Lane-Change Recognition Using Support Vector Machine

by

Weiping Deng

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Civil and Environmental Engineering
College of Engineering
University of South Florida

Major Professor: Jian Lu, Ph.D.
Chanyoung Lee, Ph.D.
Abdul Pinjari, Ph.D.
Lakshminarayan Rajaram, Ph.D.
Zhenyu Wang, Ph.D.

Date of Approval:
March 22, 2013

Keywords: GPS, IMU, PID Controller,
Trajectory Tracking, Simulation

Copyright © 2013, Weiping Deng

Acknowledgments

I would like to express my sincere appreciation to my advisor, Dr. Jian John Lu, for his support, guidance and encouragement during my Ph.D. program. Thanks to all my committee members – Dr. Chanyoung Lee, Dr. Abdul Pinjari, Dr. Lakshminarayan Rajaram, and Dr. Zhenyu Wang – for their time and valuable comment. Thanks also to thank Dr. Achilleas Kourtellis for serving as the chairperson of my dissertation defense. I have benefited greatly from my studies at University of South Florida.

Special thanks to Bin Cao, Qian Chen, Shengdi Chen, Tao Chen, Rui Guo, Bing Huang, Linjun Lu, Yongneng Xu, and all my other colleagues and friends for their suggestions, support, and help on this research.

Finally, my family deserves my deepest appreciation. It is their love that makes everything great and glorious in my life.

Table of Contents

i

List of Tables

iii

List of Figures

iv

v

Abstract

This research focuses primary on recognition of lane-change behaviors using support vector machines (SVMs). Previous research and statistical results show that the vast majority of motor vehicle accidents are caused by driver behavior and errors. Therefore, the interpretation and evaluation of driver behavior is important for road safety analysis and improvement. The main limit to understanding driver behavior is the data availability. In particular, a full-scale lane-change data set is difficult to collect in a real traffic environment because of the safety and cost issues. Considering the data demands of the recognition model development and the obstacles of field data collection, data were collected from two aspects: simulation data and the field data. To obtain field data, an in-vehicle data recorder (IVDR) that integrates a Global Positioning System (GPS) and Inertial Measurement Unit (IMU) are developed to collect data on speed, position, attitude, acceleration, etc. To obtain simulation data, a lane-change simulation with a speed controller and a trajectory tracking controller with preview ability were developed, and sufficient lane-change data were generated. Proportional-Integral-Derivative (PID) control is applied to the speed controller and trajectory tracking controller.

Simulation data were divided into two classes: dual lane-change data and single lane-change data; field data were further divided as single lane-change and non-lane-change data. Two-class and three-class classification SVM model are trained by simulation data and field data, and the model parameters were optimized by Genetic

Algorithm (GA). A radial basis function and polynomial kernel functions were found that suitable for this recognition task. The recognition results indicate that, the SVM model trained by simulation data and non-lane-change data can correctly classify up to 85 percent of single lane-change field data.

Chapter 1 Introduction

1.1 Background

Road accidents are increasingly costly in the global economy and are a growing societal concern. In 2009, there were around 10.8 million motor vehicle accidents in United States, including 30,862 fatal crashes, with total fatalities of 33,883. According to an estimate by the Nation Highway Traffic Safety Administration (NHTSA), the cost of a single vehicle fatality is $6 million, and the cost of an injury is $126,000. Since the number of motor vehicle accidents remains at a high level and the cost is rapidly increasing, road safety is being pushed to the forefront of the debate related to transportation development priorities.

As indicated by previous scientific research and statistical results, the vast majority of motor vehicle accidents are caused by driver behavior and errors. Therefore, the interpretation and evaluation of driver behavior is crucial for road safety analysis. Data availability is primarily limited to understanding driver behavior. Unlike other traffic data that can be collected for a specific site and period, driver behavior data have the characteristics of suddenness, short life span, uncertainty, etc. Thus, a desired data set to analyze driver behavior should cover long cycles and include vehicle operation parameters, roadway conditions and driver intent. Otherwise, an assessment of driver behavior may be conducted only at a very small scope and scale. The demands for driver behavior evaluation has led to the emergence of IVDR, which are on-board devices that track many functions such as data acquisition and recording of the movement, control and

1

performance of the vehicle. The technology was first used in event data recorders, which store only small fragments of crash events information for crash investigation and fault allocation.

Recently IVDR has been playing an important role in promoting safe driving and widely used. It has been reported that the use of IVDR significantly reduces traffic accident rates. Many industries have shown great interest in IVDR and have applied these devices to improve services and solution. For example, auto insurance companies encourage their customers to install data recorders to monitor driving behavior, and they offer insurance discounts to those drivers who drive well.

Evaluating driver behavior usually is based on the assessments of risk consulting experts. Most consulting methods depend on the experience of these experts, and the consulting process is completed manually. So, this approach is time-consuming and costly, and the assessment results could be subjective. Therefore, there is clearly a need for an automatic and efficient driver behavior evaluation method.

As a simulator of human intelligence in learning and decision making, artificial intelligence (AI) has been evolving to support complex recognition in particular areas. This technology has already been applied to transportation areas as a core method of intelligent transportation systems (ITS). One of the successes of AI in transportation is license plate recognition systems, which are of great significance for traffic management. To date, some studies have been conducted on applying different AI algorithms to solve driver behavior recognition problems, such as rapid acceleration, frequency of rough braking, lane changing, etc. Better understanding the characteristics of driving data and AI modeling will help to improve driver behavior recognition accuracy.

2

1.2 Problem Statement

As the foundation of driver behavior evaluation, all driving behaviors have to be detected and distinguished, and the related parameters of each driving maneuver also have to be calculated. Unfortunately, this process is obstructed not only by data availability, but also by driving maneuver classification methods, and it relies greatly on manual operation. Driving is a complex task that depends on driver habits, vehicle performance, and traffic environments, all of which determine the diversity and complexity of driving data, especially for lane-change maneuvers. Without certain judgment conditions, a traditional discriminate method is not a good solution for driving behavior recognition. Without an effective approach acquiring driving data easily and classifying driving maneuver quickly, the evaluation process will continue to be inefficient.

1.3 Research Objectives

This research intends to achieve three major objectives:

(1) To develop an IVDR for collecting data on driving behavior. Based on a GPS and IMU, the system will have the capability of collecting a variety of operation data from a probe vehicle, such as longitudinal acceleration, lateral acceleration, vertical acceleration, distance, velocity, heading angle, pitch angle, roll angle, etc. The field data set is an indispensable part of modeling and verification for lane-change recognition.

(2) To build a driving model for simulating a preview of the behavior of human drivers and obtaining full-scale lane-change data that are required by the recognition modeling. The driving model will be capable of seeing the lane-change target path directly ahead of the vehicle with a constant sight distance and determining how much to

3

turn the steering wheel. The driving model will also be able to keep a steady speed when performing lane-change movements.

(3) To develop a lane-change recognition model using SVMs. This model will have the capability of discriminating lane-change behavior from other driving behaviors.

1.4 Research Approach

Previous studies were reviewed, and the methodologies for data collection, lane-change models, and lane-change recognition were selected. Considering various vehicle operation parameters impacted by lane-change behavior, IVDR should be able to collect as many vehicle dynamic features as needed to exactly describe a lane-change maneuver. Therefore, the selection of a measurement unit is key to implement this functionality. To avoid the safety issues of data collection in real traffic environment, a lane-change model is needed to simulate lane-change maneuvers and provide a full-scale data set. The simulation method remedies the shortage of data in lane-change recognition modeling. The quality of the recognition model is under the influence of data input, data form, and the data processing method. A set of optimized model parameters is important to lane-change recognition accuracy, so an optimization method needs to be applied in the modeling process. Detailed methodologies are discussed in Chapter 3.

To achieve the objectives of this dissertation, adequate data of driver behavior was collected by an appropriate IVDR. With the advantage of cost-effective and diversity of data, a GPS and IMU were integrated into the IVDR proposed by this dissertation. The data collected by the GPS included speed over ground, course over ground (heading), latitude and longitude, the IMU collected vehicle roll angle, pitch angle, yaw angle, roll angular velocity, pitch angular velocity, yaw angular velocity, longitudinal acceleration,

4

lateral acceleration, and vertical acceleration. Since the sample rates of the GPS and IMU were distinctly different, a synchronization method based on linear interpolation was applied to combine these two sets of data sampled at different rates and provided by two independent hardware systems. To make the data query and verification more intuitive and convenient, Google Maps was embedded into the IVDR to improve the effectiveness of the manual lane-change data screening.

To obtain sufficient lane-change data that was difficult to collect on a real roadway, a lane-change simulation model was developed to imitate drivers with diverse driving habits trying to follow different given lane-change paths at different speeds. To achieve high similarity between the simulation results and the real data, a complete lane-change model should include as many characteristics as both human drivers and vehicles have in actual traffic environments. An existing full-vehicle dynamics model was imported into the lane-change simulation model and was controlled by a driving controller that had the capability of keeping a steady speed and adjusting the steering wheel to follow given paths. The driving controller simulated the preview behavior of human drivers to read the information ahead of the vehicle with a constant sight distance. A group of driving controllers with different preview times and corresponding driving parameters was found, and a full-scale lane-change dataset was generated based on the preview times, speeds, and target paths.

SVM, one of the supervised machine learning models functioning as a classifier, was chosen to analyze driving data and recognize lane-change patterns from other driving behavior patterns. Thus, the recognition task was simplified as a classification task. With the associated learning algorithms, a well-trained SVM model can accurately classify a

5

given data set into two categories or multiple categories, for this dissertation, the categories are lane-change and non-lane-change.

The first step to build an SVM classification model was data reduction. Driving data were divided into two datasets, a training data set and a validation dataset, formed as the input datasets of SVM classification model training. To increase recognition precision and reduce model calculation time, a time window that decided how many driving data samples to input into the SVM model in one computation cycle was determined, even if the driver behaviors did not have fixed time lengths. Generally, the major characteristics were covered by that fixed time window. It is important to choose only the most effective set of features to train the SVM model. Inputting all the possible features into the training process not only failed to improve the output result, but it also degraded the discrimination performance of the SVM. The features that show apparent differences during lane-change behavior against others are critical ones. Lateral acceleration, vehicle heading and vehicle angle were discussed in data reduction. For extracting exclusive feature patterns from various kinds of driving behaviors, a simple data representation that represents the features of lane-change behavior also was selected.

The second step, to build a SVM classification model was to select a kernel function, which is a class of algorithms for data processing. However, in the theory of SVM, a general process for determining an appropriate kernel function is not provided. Usually, only a few types of kernels are available for classification tasks, and they have different performance for the same classification problem. Thus, to find the correct kernel function, the available driving dataset was tested against different types of kernel functions before knowing the performance of each kernel function.

6

A closed loop feedback training approach was applied to optimize the kernel function parameters. During the training process, a genetic algorithm (GA), was used to generate a new set of kernel parameters in every SVM model training cycle to decrease the estimate errors on validation data set. The parameters sets of the final training cycle were extracted and used as the parameters of the lane-change recognition model.

Figure 1 shows the approach of this research.
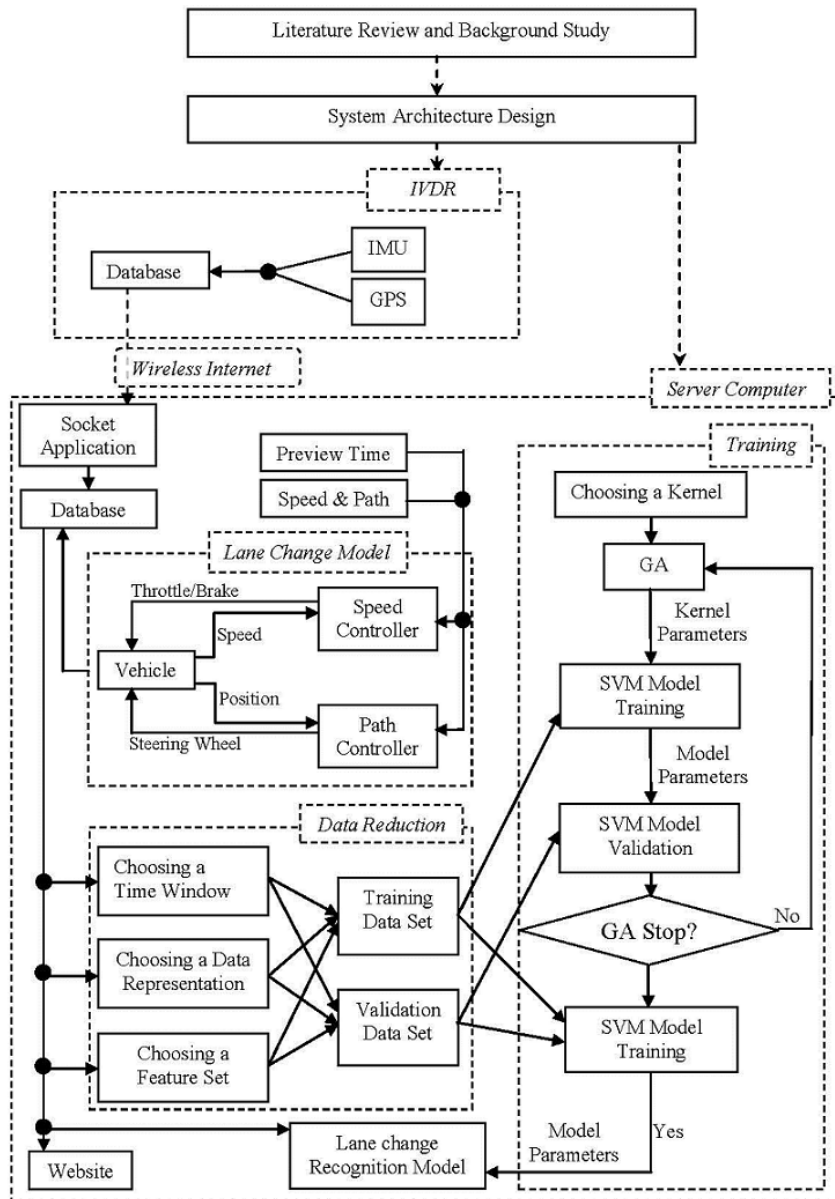
Figure 1 Research Approach

Chapter 2 Literature Review

Previous studies and findings related to driving simulation and driving behavior recognition are reviewed and summarized in this chapter.

Data availability is the primary limitation to better understanding driver behavior, collecting field data related to driving behavior is restricted by several factors, such as safety issues, cost, etc. The use of simulation can avoid these obstacles and provide sufficient data for lane-change recognition study.

With the wide use of ITS, the demand for driver behavior detection or assistance technology has increased rapidly, and can help to evaluate and improve transportation safety quickly and easily. AI technologies have shown outstanding performance for driver behavior detection and recognition. SVM, a concept in statistics and computer science used for classification and regression analysis, has been widely studied and applied to solve practical problems.

2.1 Driving Simulation

Lee and Park (1994) built a driver model as a preview PID controller to adjust the steering wheel of a full-car model in a closed-loop control system. For efficiency in obtaining driver control gains, a bicycle model with a driver was first used to get the proper control parameters for lane-change maneuvers. The bicycle model originated from a full-car model, and the control parameters depended on the bicycle speed and preview distance. In the driving simulation, the speed was set to 100 kilometer per hour, and the preview distance was 15 meters. After simulation with several combinations of control

8

parameters, the researchers found the dominant parameters in the control system performance and applied these parameters to a full-car model. Due to the different characteristics between the bicycle model and a full-car model, the control parameters were adjusted to achieve the desired lane-change performance. Finally, a good combination of the control parameters was obtained for the full-car model.

Guo (1981) found that there were no criteria for evaluating vehicle handling characteristics. To solve this problem, he proposed a driver-vehicle closed loop model to simulate driving behavior using a preselected optimal curvature method. He first developed a single point preview driver-vehicle model in which it was assumed that the driver could pay attention only at one point with a constant sight distance. The driver-vehicle model was required to make lane-change movements along given paths. Based on what the driver saw in the target path, the driver accepted the lateral position feedback information and adjusted the steering wheel of the vehicle model. Different parameters with combinations of both driver and vehicle were loaded into the model to generate a group of results. A driver-vehicle model with a preview time window was also built and was compared to the single preview time model. To verify the driver-vehicle closed loop models, simulations of a heavy truck making a double lane-change were conducted. The simulation results of both models were good. This method was developed based on the author's prior work on simulation on a preselected follower method using Taylor's expansive formulate, and the author notes that both methods had similar results.

2.2 Diver Behavior Detection

Toledo, Musicant, and Lotan (2008) used IVDR systems to monitor and provide feedback to drivers on their driving behavior. The system described in the research

recorded vehicle operation data and used this information to identify and classify various driving maneuvers. These maneuvers were then used to calculate various driving risk indices. The results from the research showed that these indices were correlated with the drivers' crash records. This result suggested that risk indices might be used as indicators to the drivers' rates of involvement in car crashes. A study of crash rates before and after system installation showed statistically significant reductions in the crash rates. It should be noted that these results were based on a relatively short period of time after using the system and, therefore, need to be further evaluated. An evaluation of the temporal changes in the IVDR risk indices showed that the initial exposure to the IVDR feedback caused a substantial reduction in risk indices, which could be further enhanced if drivers continued to access the IVDR feedback. Even without additional feedback, the initial impact was sustained for several months. Therefore, further research is needed to better understand the temporal and long-term impacts of the installation and to develop feedback management schemes to maintain driver interest in the feedback and maximize its impact. Finally, the researchers noted that whereas their analysis demonstrated that IVDR might be useful to impact driver behavior, they did not study the psychological and social mechanisms that underlie this change. Understanding these mechanisms is very important in making effective use of IVDR systems.

Liang, Reyes, and Lee (2007) applied SVMs to build a real-time approach for recognizing cognitive distraction using driving performance data and driver eye movements. Because the use of in-vehicle information systems (IVIS) such as cell phones and navigation systems has increased, driver distraction has become a significant and growing safety concern. The researchers believe that an effective way to solve this

10

problem was to detect driver distraction and use in-vehicle systems to reduce such distractions. Data were collected in a driving simulation experiment in which 10 drivers interacted with an IVIS while driving. The simulation data were used to build both SVM and logistic regression models, and three different model characteristics were discussed: definition of distraction, input data feature, and input data representation. The results showed that the SVM models could detect driver distraction with an average accuracy of 81.1 percent, which was better than traditional logistic regression models. The best recognition accuracy was 96.1 percent, which resulted when driving distraction was defined by experimental conditions (i.e., IVIS drive or baseline drive); the input data included both eye movement and driving measures, and these data were represented over a 40-second window with a 95 percent overlap of windows. These recognition results indicated that eye movements and simple measures of driving performance could be used to detect driving distraction in real time. This study provided knowledge for the design of adaptive in-vehicle systems and the estimate of driver distraction.

Mandalia and Salvucci (2005) used SVM to detect vehicle lane-change movements. They performed an evaluation study by applying the SVM technique to lane-change recognition and, at the same time, tested a subset of the space of possible data representations and feature sets. In the study, driving data were collected by four drivers, who were asked to drive on a Japanese multi-lane highway environment for one hour each through dense and smooth traffic without specific goals or instructions. The results for the various combinations of window size, feature sets (shown in Table 1), and non-overlapping vs. overlapping representations are shown in Tables 2 and 3.

11

Table 1 Feature Sets

Set	Features
1	Acceleration, Lane position 0, Lane position 30, Heading
2	Acceleration, Lane position 0, Lane position 30, Heading, Lead Car distance
3	Acceleration, Lane position 0, Lane Position 20, Lane position 30, Heading, Longitudinal acceleration, Lateral acceleration
4	Acceleration, Lane position 0, Lane position 30, Heading, Steering Angle
5	Lane position 0, Lane position 10, Lane position 20, Lane position 30

Table 2 Accuracy of Window Size (Non-overlapping)

Window	Set 1	Set 2	Set 3	Set 4	Set 5
5 s	83.5	90.0	91.2	90.0	91.1
4 s	88.1	91.3	92.5	91.5	92.2
2 s	89.3	93.0	97.7	94.0	97.4
1.5 s	85.2	93.7	96.3	93.2	97.7
1.2 s	96.8	96.0	96.0	96.0	96.7
0.8 s	86.3	91.8	90.0	86.6	94.6

Table 3 Accuracy of Window Size (Overlapping)

Window	Set 1	Set 2	Set 3	Set 4	Set 5
5 s	87.1	86.9	89.0	88.1	87.8
4 s	89.9	90.7	91.5	89.5	91.1
2 s	96.2	96.2	97.8	95.8	97.3
1.5 s	94.5	94.6	97.6	93.3	97.5
1.2 s	93.8	93.7	95.0	93.2	**97.9**
0.8 s	97.0	95.5	96.0	96.4	96.7

In all feature sets, the average accuracy of the SVM models on detection lane-change movements was 95 percent, and results improved with decrease in window size. The overlapping representation with all the lane position features (Set 5) generated the

12

best recognition result, with 97.9 percent accuracy. The system generated a continuous recognition, which means it marked each sample with a positive (lane-change) or negative (lane-keeping) label. Thus, out of every 100 actual lane-changing samples, about 98 were detected correctly, whereas out of every 100 lane-keeping samples, only 5 were incorrectly recognized. The recognition system was also analyzed for time accuracy to calculate how much time had elapsed from the start of a lane-change until the point of detection. The system was able to detect about 87 percent of all true positives within the first 0.3 seconds from the start of the maneuver. A richer data set with features such as lead-car velocity and eye movements should lead to even better accuracy. The authors compared their results to previous studies and found that Pentland and Liu (1999) achieved accuracy of 95 percent, but only after 1.5 seconds into the maneuver; Kuge et al. (2000) achieved 98 percent accuracy of recognizing entire lane-changes; and Salvucci's (2004) mind-tracking algorithm achieved approximately 87 percent accuracy. The SVM approach outperformed previous approaches in lane-change detection.

13

Chapter 3 Methodology

This chapter describes the methodologies selected for driving simulation, data processing and lane-change recognition. In driving simulation model, a speed controller is used to control the throttle and brake, while a path follower is used to control the steering wheel. Both of these two controllers use a PID control strategy. Because of the noise included in the data collected by the IMU, these methodologies are used to solve some SVM modeling issues, such as using a SVM library, kernel selection and data reduction for SVM training.

3.1 PID Controller

In this dissertation, the vehicle model is extracted from a multi-body dynamics software that has the capability of simulating real world physics. The characteristics of this vehicle model are very close to those of a real vehicle. Therefore, a driver model is needed to drive the vehicle as a real driver, who is capable of controlling the steering wheel, throttle and brake to accomplish a driving task. Two PID controllers are applied to control the vehicle speed and vehicle position. The speed controller is used to adjust the throttle and brake to keep a steady speed when making a lane-change maneuver. The position controller is used to adjust the steering wheel for the attempt of following a given lane-change path, at any prescribed speed.

A PID controller is a generic closed loop feedback controller widely used in control system. It aims to minimize the error between the controlled object output and a desired setpoint by adjusting the controlled object input. The PID controller algorithm

14

involves three terms: proportional (P), integral (I) and derivative (D). For each term, there is only one constant parameter. Usually, these three terms are interpreted in term of error type: P is used on the present error, I on the accumulative error and D is the current rate of error change. The weighted sum of these three terms is used to adjust the controlled object via a control element. The form of the PID is shown in Equation 1.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt}e(t) \qquad (1)$$

where,

$u(t)$ = controller output

K_p = proportional gain

K_i = integral gain

K_d = derivative gain

e(t) = s(t) − y(t), where s(t) is the setpoint, y(t) is the output of the controlled object

τ = variable of integration.

Figure 2 shows a block diagram of a PID controller in a closed feedback loop.
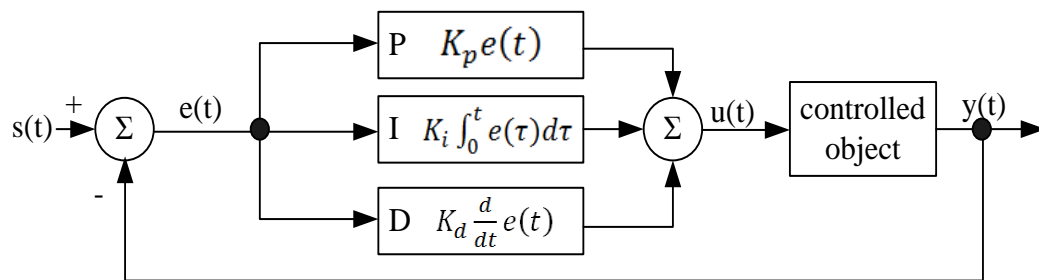
Figure 2 PID Controller in Closed Feedback Loop

A PID controller can be classified as analog or digital type, and both can be used in a closed control loop. Equation 1 shows an analog PID controller. A digital PID is discrete and generally is used in a computer control system. In the lane-change model of

15

this dissertation, the vehicle model was built in an analog development environment but can be simulated only as a discrete model on computer. As one kind of digital PID, an incremental PID (Equation 2) algorithm is embedded into both the speed controller and the position controller in the lane-change simulation.

$$u(k) = u(k-1) + K_p[e(k) - e(k-1)] + K_i e(k) + K_d[e(k) - 2e(k-1) + e(k-2)] \quad (2)$$

where,

$u(k)$ = controller output

K_p = proportional gain

K_i = integral gain

K_d = derivative gain

e(k) = s(k) – y(k), where s(k) is the setpoint, y(k) is the output of the controlled object.

Equation 2 indicates that the output of the incremental PID algorithm uses the last only three errors to generate a control increment in each computation step. A feedback loop control using incremental PID algorithm is different from that shown in Figure 2, and is shown in Figure 3. For controlling the speed of the vehicle model, both the setpoint
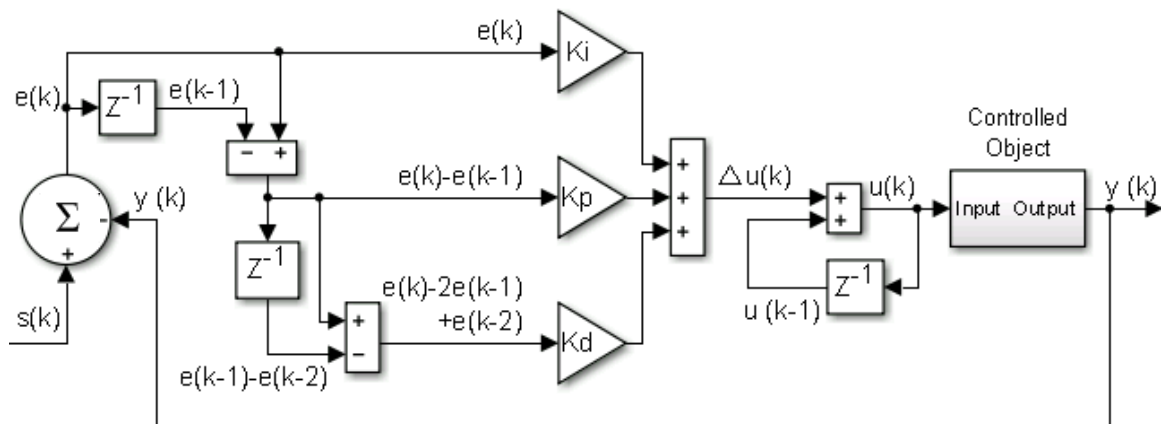
Figure 3 Incremental PID Controller in a Feedback Loop

16

s(k) and the controlled object output y(k) are speed value, the u(k) is throttle and brake inputs. Similarly, in position control, the s(k) and y(k) are in a lateral position, and the u(k) is the steering wheel radian.

A combination of optimum parameters is the key factor for the PID controller to achieve the desired control response. The incremental PID only has three parameters and is described in a simple equation 1; however, it is a difficult to determine three parameters because different controlled objects have different system properties and behaviors. If a precise mathematical model of the controlled object is given, the task of finding the PID parameters could be simple. However, a mathematical model is difficult to obtain, and an empirical value is set to the PID controller and adjusted according to the response results. An optimization method could help to improve the control effect, but only will be available when the numeric ranges of the PID parameters are given.

3.2 Genetic Algorithm

GA, as one part of AI, is a search algorithm using techniques inspired by natural evolution. This algorithm is usually used to solve optimization and search problems. GA generates a series of populations of possible solutions to an optimization problem and improves these solutions toward a better one. Each solution has a set of properties that can be updated in the optimization process. This dissertation uses GA to optimize the parameters of the speed controller and the trajectory tracking controller in the lane-change simulation model and other parameters used in lane-change recognition model.

The numerical computing software MATLAB provides functions using GA to determine the minimum of an objective function. The GA function used in this research is shown as follows.

17

[x, fval, exitflag, output] = ga(fitnessfcn, nvars, [ ], [ ], [ ], [ ], LB, UB, [ ], options)

where,

fitnessfcn = handle to the fitness function. The fitness function should accept a

row vector of length nvars and return a scalar value which is aimed to minimize by GA

nvars = positive integer representing the number of variables in the problem

LB = vector of lower bounds of the fitnessfcn input

LB = vector of lower bounds of the fitnessfcn input

options = structure containing optimization options

x = best point that GA located during its iterations

fval = fitness function evaluated at x

exitflag = integer giving the reason GA stopped iterating

output = structure containing output from each generation and other information

about algorithm performance.

The implementation of an optimization process using GA is shown in Figure 4.
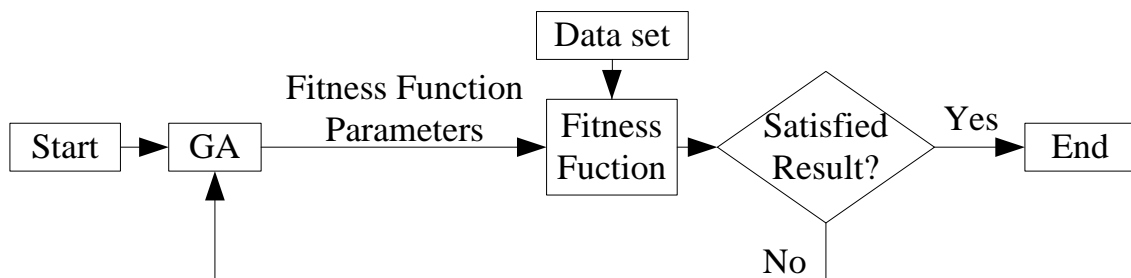
Figure 4 Optimization Process using GA

3.3 Support Vector Machine

3.3.1 Brief Overview of SVM

SVM is a set of supervised models with related learning algorithms that analyze

data and identify patterns, and is used for solving classification and regression problems.

18

It is capable of maximizing recognition accuracy and automatically avoiding over-fit to the data. SVM uses a given data set to construct a set of hyperplanes in a high-dimensional space, which can be used for classification, pattern recognition, distribution estimation, etc. SVM has been an active part of AI research and has been increasingly used in data analysis and image recognition areas.

In supervised learning, a machine is trained instead of preset related parameters by using a set of training instances of input-output pairs. The machine training aims to learn a combination of parameters that are used in the machine and can best describe the relationship between the inputs and the outputs of the training data. The SVM training in this dissertation can be considered as an approach of determining appropriate kernels and a set of parameters in Equation 3, which is used for classification.

$$f(x) = \sum_{i=1}^{l} c_i * K(X, X_i) \tag{3}$$

where,

f = category labels

X_i = input instances, *i = 1,2,...,l*

K = kernel, a certain definite function

c_i = a set of parameters to be determined from the training data set

Kernel functions map the instance data into higher dimensional spaces to make the instance data more easily separated and classified. Different kernel function have different for performance on reconstructing the instance data. Generally, there are two types of kernel functions, linear and nonlinear. The kernel function and parameters are discussed in the next section.

19

3.3.2 LIBSVM - A Library for SVM

LIBSVM is a library for SVMs and has been developed and upgraded by Chih-Chung Chang and Chih-Jen Lin at National Taiwan University since year 2000. It supports regression, distribution estimation, and multi-class classification. As a tool, it provides a simple interface and can be easily used in users' programs. The LIBSVM has been widely used in many areas. Some representative works that successfully used the LIBSVM are listed in Table 4.

Table 4 Successes in LIBSVM

Domain	Representative works
Computer vision	LIBPMK (Grauman and Darrell, 2005)
Natural language processing	Maltparser (Nivre et al., 2007)
Neuroimaging	PyMVPA (Hanke et al., 2009)
Bioinformatics	BDVal (Dorff et al., 2010)

LIBSVM provides a MATLAB interface with only four functions, two of which, svmtrain and svmpredict, are used for training and estimate, respectively.

model = svmtrain(training_label_vector, training_instance_matrix [, 'libsvm_options'])

where,

model = the output of 'svmtrain'

training_label_vector = an m by 1 vector of training labels with m instances

training_instance_matrix = an m by n matrix of m training instances with n

features

libsvm_options = a string of training options using the format as follows.

-s svm_type = select type of SVM (default value= 0)

0 -- c-support vector classification (SVC)

20

1 -- nu-SVC

2 -- one-class SVM

3 -- epsilon- support vector regression (SVR)

4 -- nu-SVR

-t kernel_type = select kernel function (default value = 2)

0 -- linear: $u'*v$

1 -- polynomial: $(gamma*u'*v + coef0)^{degree}$

2 -- radial basis function: $exp(-gamma*|u-v|^2)$

3 -- sigmoid: $tanh(gamma*u'*v + coef0)$

-d degree = set degree in polynomial kernel function (default value = 3)

-g gamma = set gamma in non-linear kernel function (default value = 1/k,

k is the number of attributes in one input instance)

-r coef0 = set coef0 in polynomial and sigmoid kernel function (default

value = 0)

-c cost = set the parameter C of C-SVC, epsilon-SVR, and nu-SVR

(default value = 1)

-n nu = set the parameter nu of nu-SVC, one-class SVM, and nu-SVR

(default value = 0.5)

-p epsilon = set the epsilon in loss function of epsilon-SVR (default value

= 0.1)

-m cache size = set computation cache memory size, in MB (default value

= 100)

-e epsilon = set tolerance of termination criterion (default value = 0.001)

21

-h shrinking = whether to use the shrinking heuristics, 0 or 1 (default value

= 1)

-b probability_estimates = whether to train a SVC or SVR model for

probability estimates, 0 or 1 (default value = 0)

-wi weight = set the parameter C of class i to weight*C, for C-SVC

(default value = 1)

[predicted_label, accuracy, decision_values/prob_estimates]= svmpredict(testing_

label_vector, testing_instance_matrix, model [, 'libsvm_options']);

where,

predictd_label = a vector of predicted labels

accuracy = a vector including classification accuracy, mean squared error, and

squared correlation coefficient for regression

decision_values/prob_estimates = a matrix containing decision values or

probability estimates

testing_label_vector = an m by 1 vector of prediction labels

testing_instance_matrix = an m by n matrix of m testing instances with n features

model = the output of 'svmtrain'

libsvm_options = same as the one in 'svmtrain'

Chapter 4 Field Data Collection

This chapter focuses on field data collection, including IVDR, observing site selection, and data description.

4.1 Data Acquisition System - IVDR

4.1.1 GPS and IMU

Vehicle speed, heading, and position are the basic information that should be collected by the IVDR. These data fields can be obtained easily by a low-cost GPS receiver with good precision. One of the most important technical indices of a GPS receiver is its update rate, which decides the frequency of acquiring data from the satellites and the cost of the device. Another factor of concern during selecting a GPS receiver is the data interface, which affects the application development period and reliability. As a low-cost GPS receiver board equipped with two serial ports, the RCB-4H GPS receiver provides up to a 4 Hz data sampling rate and is integrated into the IVDR. Its features are listed as follows.

- Position accuracy: 2.5m CEP (Circular Error Probability)
- Accuracy of timepulse signal: 99% <100ns
- Operating temperature range: -40 to 85 ℃
- 4 Hz position update rate
- Two serial ports
- 16 channel positioning engine
- Ultra-low power consumption

23

- Power brown-out protection

- Wide supply voltage range: 3.15 to 5.25V

An antenna is needed for the GPS to receive a signal from the positioning satellites. The output messages of RCB-4H GPS can report the condition of the antenna supplies. The RCB-4H supports NMEA protocol, whose structure is shown in Figure5.

		Checksum range		
$	<Address>	{,<value>}	*<checksum>	<CR><LF>
Start character	Address field	Data field	Checksum field	End sequence

Includes 2 fields

<XX>	<XXX>
Talker Identifier	Sentence Formatter

Delimited by a ','. Length can vary.

Consists of 2 characters representing a hex number.

Example:

$	GP	ZDA	,141644.00,22,03,2002,00,00	*67	<CR><LF>

Figure 5 NMEA Protocol Frame

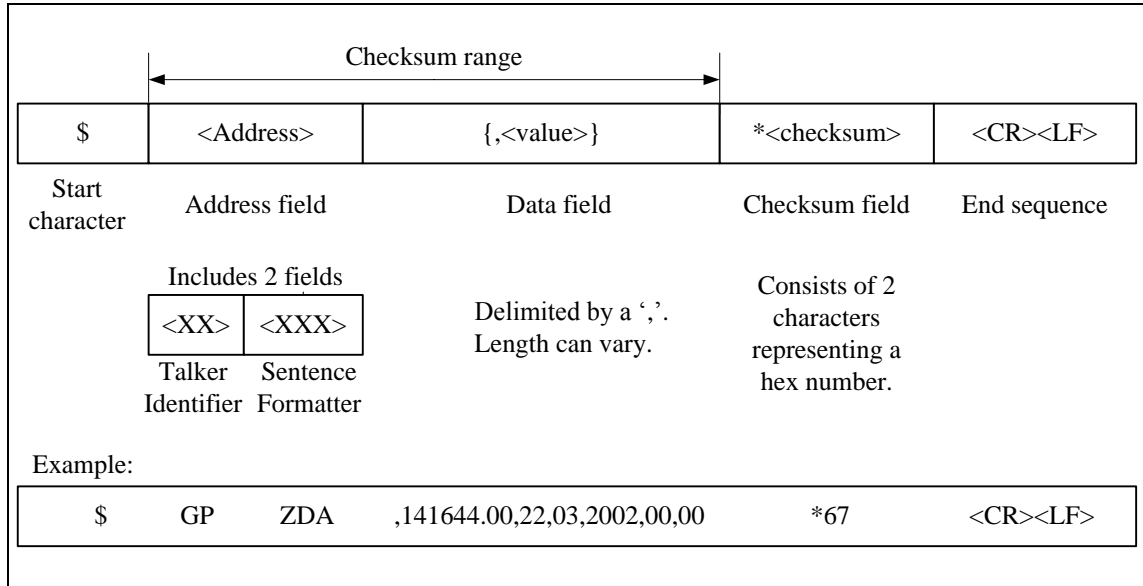NMEA protocol supports many NMEA messages, one of which, recommended minimum data (RMC) message, issued in the IVDR. The RMC message provides all the data fields needed by this research, such as speed, heading, latitude, and longitude. The RMC message is defined as follow.

$GPRMC,hhmmss,status,latitude,N,longitude,E,spd,cog,ddmmyy,mv,mvE,mode*cs<CR><LF>

Table 5 shows the detail of RMC message.

24

Table 5 Recommended Minimum Data (RMC)

Name	ASCII String		Units	Description
	Format	Example		
$GPRMC	string	$GPRMC		RMC protocol header
hhmmss	hhmmss.sss	153559.25		Time of position fix
status	character	A		Status
				V = navigation receiver warning
				A = data valid. See Position Fix Flags in NMEA Mode
latitude	ddmm.mmmm	2817.1144		User datum latitude degrees, minutes, decimal minutes format
N		N		N=north or S=south
longitude	ddmm.mmmm	-8233.912		User datum latitude degrees, minutes, decimal minutes format
E	character	E		E=east or W=west
Spd	numeric	30.5	knots	Speed Over Ground
cog	numeric	77.52	degrees	Course Over Ground
ddmmyy	ddmmyy	022813		Current Date in Day, Month Year format
mv	numeric		degrees	Not being output by receiver
mvE	character			Not being output by receiver
mode				See Position Fix Flags in NMEA Mode
cs	hexadecimal	*53		
<CR><LF>				End of message

IMU, which measures roll angle, roll angular, pitch angle, pitch angular, yaw angle, yaw angular, longitudinal acceleration, lateral acceleration, and vertical acceleration of the vehicle, is another essential part of IVDR.VG320, an IMU from Crossbow Technology, is integrated into the IVDR to fulfill this data acquisition task.

VG320 is a commercial MEMS-based IMU with a 6-DOF MEMS inertial sensor core, which includes three axes of MEMS angular rate sensing and three axes of MEMS linear acceleration sensing. This IMU is capable of acquiring various data such as attitude,

25

angular rates, and accelerations at a high sampling rate, up to 100 Hz. It uses a serial port

for data communication and each data transmission. Table 6 shows the performance of

VG320.

Table 6 Performance of VG320

Heading	
Range (°)	±180
Accuracy (°)	relative, freely integrated
Resolution (°)	< 0.1
Attitude	
Range: Roll, Pitch (°)	±180, ±90
Dynamic Accuracy(°)	< 2.0
Resolution (°)	< 0.1
Angular Rate	
Range: Roll, Pitch, Yaw (°/sec)	±150
Scale Factor Accuracy (%)	< 1
Resolution (°/sec)	< 0.02
Acceleration	
Input Range: X/Y/Z (g)	±4
Scale Factor Accuracy (%)	< 1
Resolution (mg)	< 0.5
Environment	
Operating Temperature (°C)	-40 to +85
Electrical	
Input Voltage (VDC)	9 to 42
Power Consumption (W)	< 3

In VG320, Angle Data Packet 2 (A2) is the default output data that will be

generated automatically after the IMU is powered up. Its format is shown in Table 7.

Table 7 Angle Data Packet 2

Preamble	Packet type	Length	Payload	Termination
0x5555	0x4132	0x1E	<A2 payload>	<CRC>

Table 8 shows the A2 payload contents.

26

Table 8 Payload Contents

Byte Offset	Name	Scaling	Units	Description
0	rollAngle	$2*pi/2^{16}$	radians	Roll angle
2	pitchAngle	$2*pi/2^{16}$	radians	Pitch angle
4	yawAngle	$2*pi/2^{16}$	radians	Yaw angle
6	xRate	$7*pi/2^{16}$	rad/s	Roll angular rate
8	yRate	$7*pi/2^{16}$	rad/s	Pitch angular rate
10	zRate	$7*pi/2^{16}$	rad/s	Yaw angular rate
12	xAccel	$20/2^{16}$	g	X accelerometer
14	yAccel	$20/2^{16}$	g	Y accelerometer
16	zAccel	$20/2^{16}$	g	Z accelerometer
18	xRateTemp	$200/2^{16}$	deg C	X rate sensor temperature
20	xRateTemp	$200/2^{16}$	deg C	Y rate sensor temperature
22	ZRateTemp	$200/2^{16}$	deg C	Z rate sensor temperature
24	timeITOW	1	ms	DMU ITOW
28	BITstatus	-	-	Master BIT and Status

The coordinate system of the VG320 is shown in Figure 6. It is an orthogonal right-handed coordinate system. Accelerations are positive when they are oriented towards the positive side of the coordinate axes. The angular rate sensors are aligned with these same axes and measure angular rotation rate around a given axis. The direction of a positive rotation is defined by the right-hand rule.
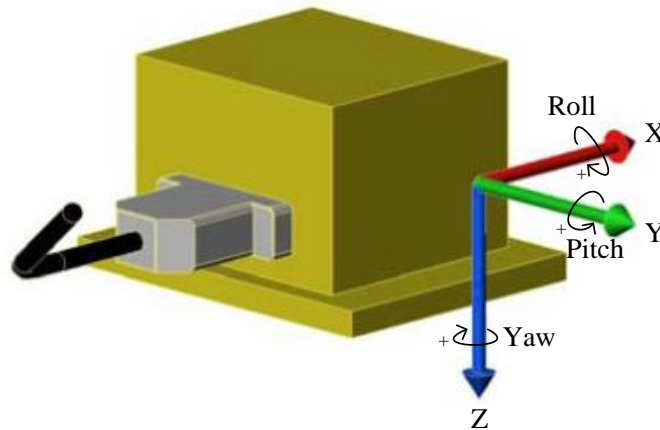

Figure 6 Coordinate System of VG320

27

The measurement data are provided by the GPS and IMU at different update rates and in different formats. These two sets of data have to be integrated inside the IVDR to attain uniform data that are acceptable for the database. Data synchronization and data linear interpolation were applied to merge the data. Figure7 shows the simplified working procedure of the GPS/IMU.
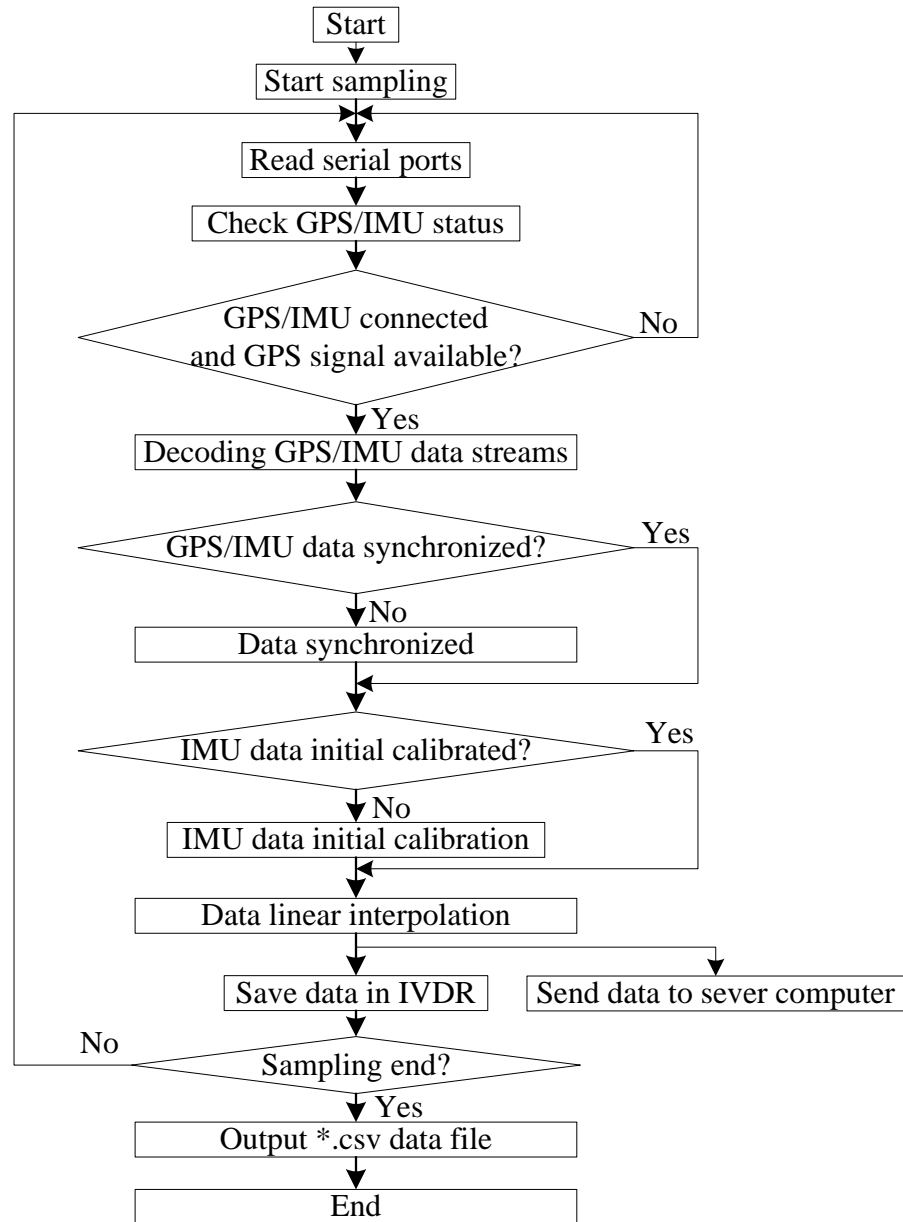

Figure 7 Working Procedure of GPS/IMU

28

4.1.2 Network Socket Application

A User Datagram Protocol (UDP) socket was deployed for the communication between the IVDR and server computer. The socket application links the IVDR to the database installed in the server computer and realizes real-time data transmission from the IVDR to the server computer. The in-vehicle computer can focus on data collection while the SVM model computation and the website demonstration are distributed on the server computer.

As a core member of the Internet protocol suites, UDP enables the computer application to send datagrams to an objective computer on the Internet without prior communication to set up a connection. These two computers are known as the UDP client and the UDP server.UDP is a simple and transaction-oriented transmission model that can be used for purposes where checking datagram loss and reporting error is either not necessary or a user-defined data format is needed in the application. Although there is no guarantee of data delivery over unreliable Internet media, the lack of retransmission delays makes UDP suitable for real-time application as the IVDR in this research. For a data transmission task, the IP address and port number of the UDP sever have to be set to the UDP client, and a user-defined data coding method is used for coding and decoding the datagram transmitted between the UDP client and server. In this project, the diagram format includes 18 data fields delimited by a comma. The first data field is a "#" as a beginning flag; the last one is a "$" as an ending flag; and the other 16 data fields include 15 collected by the IVDR and 1 time index, which is used for sorting data according to collected time in the UDP server. Figure 8 presents how the UDP socket works between the IVDR and the server computer.
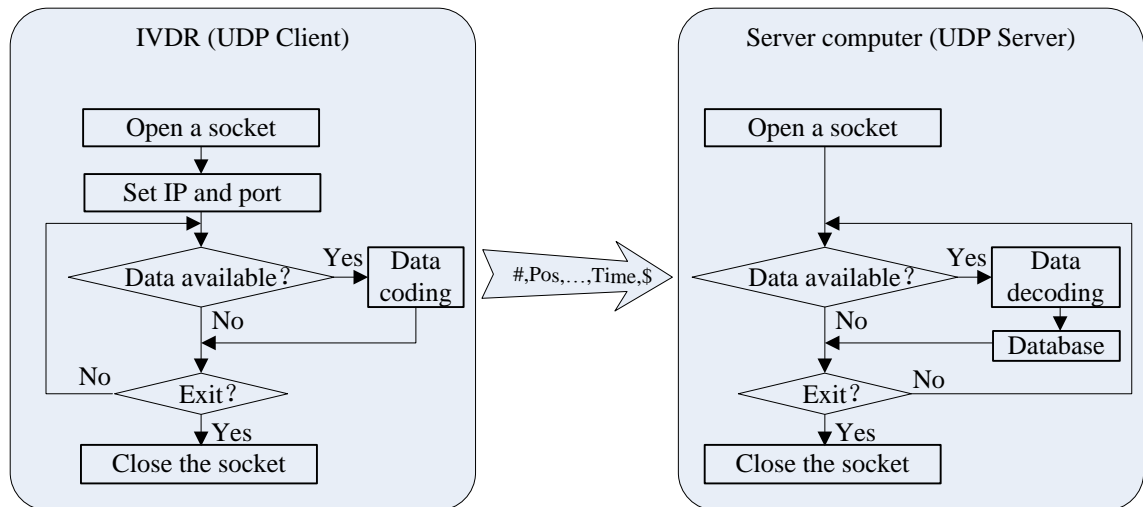
Figure 8 UDP Socket Flow Diagram

4.1.3 Website and Map

To dynamically output the recognition result and track the instrumented vehicle, a Google Map API-based website driven by Asynchronous JavaScript and XML (AJAX) was deployed on the server computer.

AJAX is a group of Web development technologies used on the client side for exchanging data asynchronously with a website server and updating parts of a Web page without reloading the entire page. It provides an XMLHttpRequest object to send the request to a website server for retrieving data. The Web page developed in this project retrieves data from the server computer once per second. The server responds to every request with a data string, whose format is the same as the one used in the UDP. The data string is decoded on the client browser and updated to a corresponding field of the Web page.

Google Maps provides rich functional APIs that allow users to embed the various functionality and usefulness of Google Maps into their own website and applications and overlay their own data on top of them. The Google Maps API is a free service, available

30

for any website that is free to consumers. The latitude and longitude data acquired by GPS can locate the probe vehicle on Google Maps or show its history location. With AJAX, the location of the probe vehicle is updated dynamically on the map. Compared to presenting the raw data or simple charts, introduction of a dynamic electronic map increases the intuition of data and improves user experience of IVDR significantly

4.1.4 IVDR

The structure of the system is shown in Figure 9.The IVDR consists of a GPS, a GPS antenna, an IMU, a DC/AC converter, a computer, data acquisition application software, a two-serial PCMCIA adapter (if necessary), and a wireless network card.
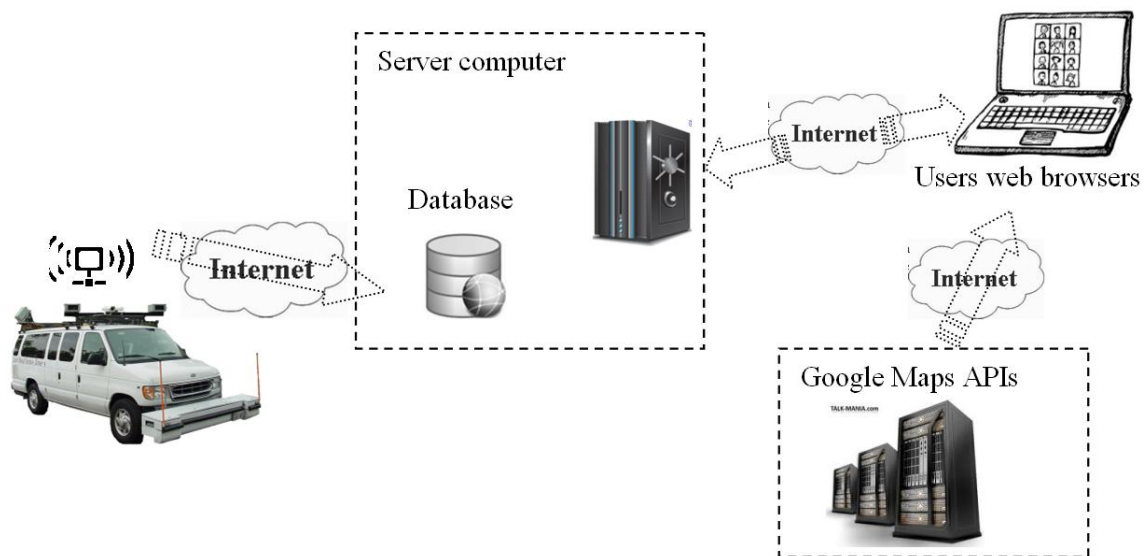

Figure 9 IVDR and Information System

According to the GPS manual, a GPS antenna is required to be placed on top of the vehicles to ensure exposure to as many satellites in the sky as possible. A 5V DC car power supply was converted into a 110V AC by the DC/AC converter to power the GPS/IMU and onboard computer.

Figure 10 IVDR Installation

Data acquisition application software was developed to provide a set of friendly user interfaces that facilitate system parameter setting, data calibration, data sampling, and analysis. The sampling interface that integrates the calibration function and the real-time data curve display is shown in Figure 11.
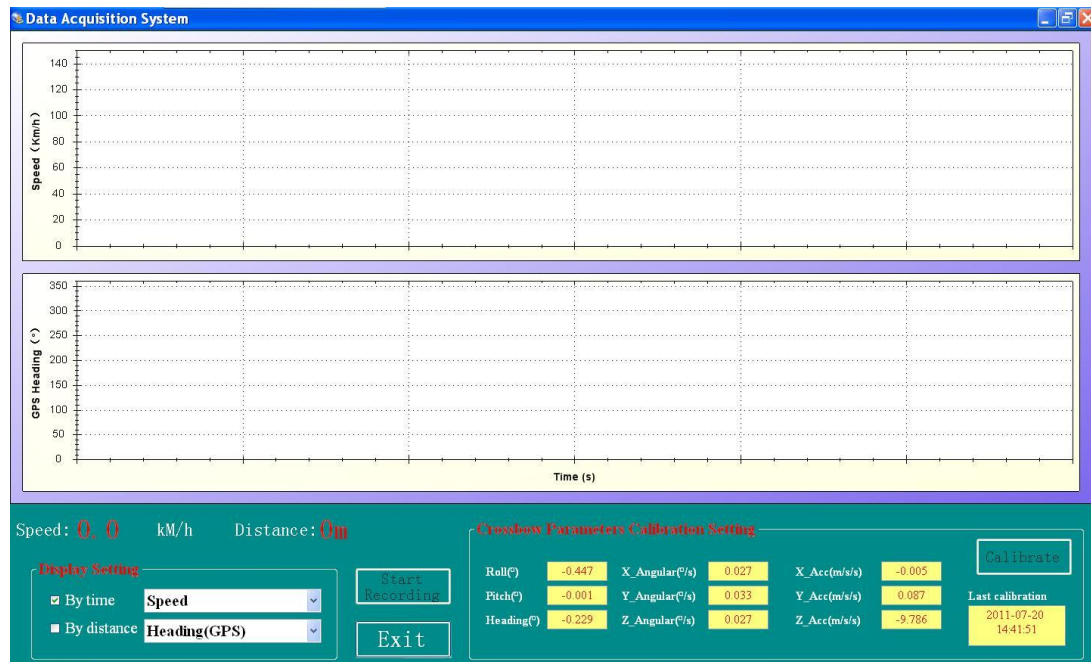

Figure 11 Sampling Interface

32

Figure 12 shows the socket application and database deployed on the server computer. The socket application is able to decompose the datagram retrieved from the IVDR and store the data to the database in a time sequence.
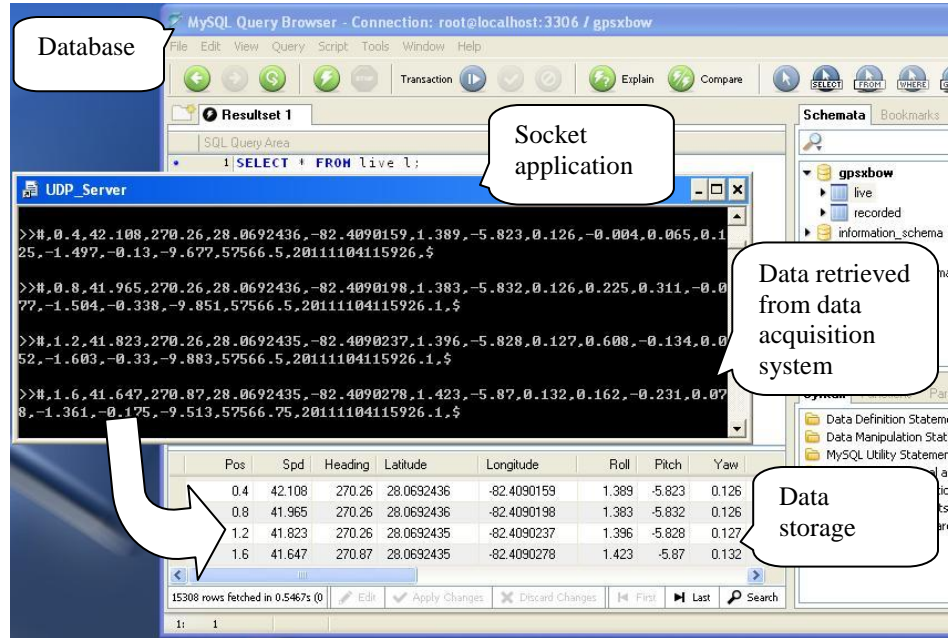

Figure 12 Socket Application and Database on Server Computer

On the server computer, a website server was set up to manage the deployment and publishing of Google Maps API-based website. Figure 13 shows a launched website server that continuously sends GPS/IMU data to a connected user browser.
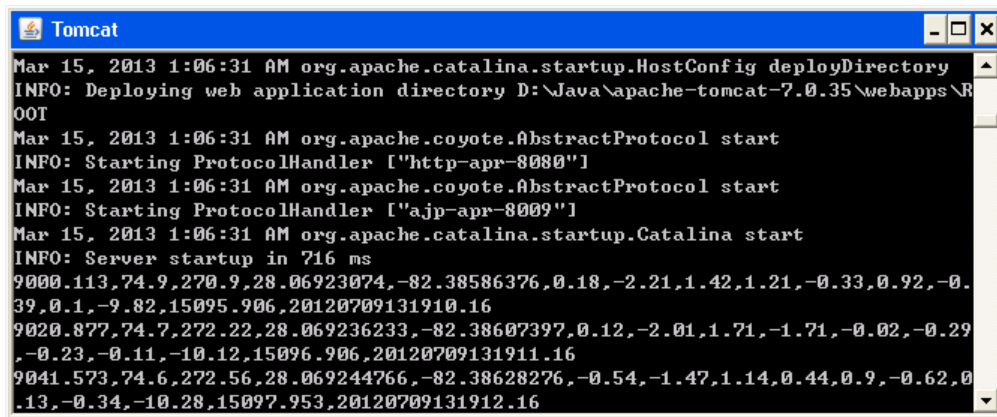

Figure 13 Website Server

33

In the users' browsers, a dynamic website shows both an animated bubble and constantly updated data, which demonstrates the data acquisition process in real time.
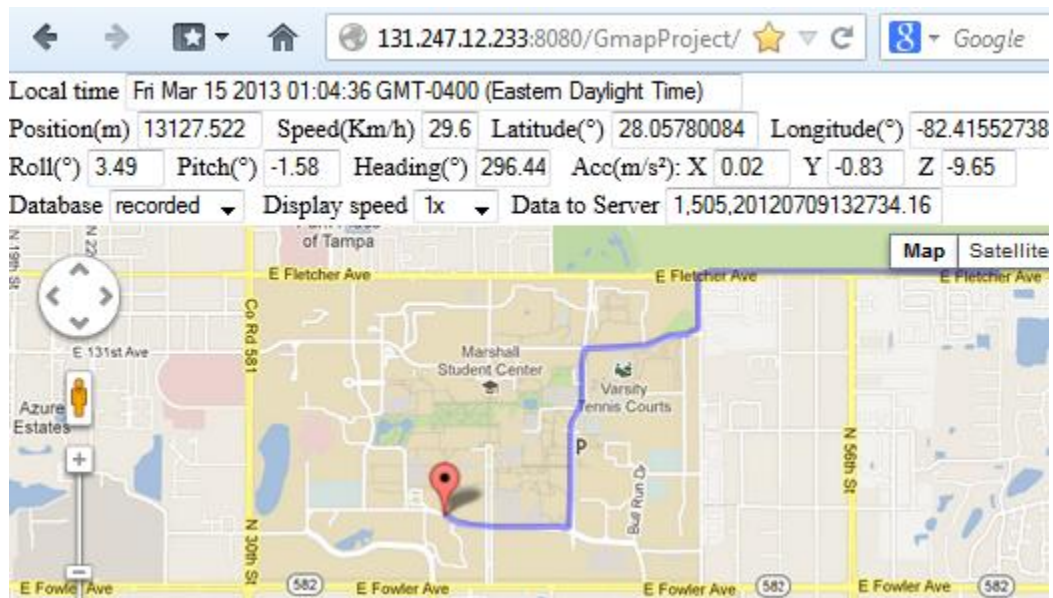
Figure 14 Website Based on Google Maps API

4.2 Site Selection and Data

4.2.1 Site Selection

The exact number of instances that are needed to train a lane-change recognition model is still in question. A large amount of lane-change data is not easily collected in real traffic, especially at high speeds. Inappropriate and frequent lane-change movements can significantly affect the traffic environment and cause safety issues. Multiple-lane roads with low traffic volumes are ideal data collection sites.

Therefore, roadway sections on I-75, E. Fowler Ave., E. Fletcher Ave., and 56th St. in Tampa (in Figure 15) were selected for data collection. The operation speeds are from 60 km/h to 100 km/h on these roadway sections. Some roadway sections with lower speed limits on the USF campus were also selected for lane-change data collection. Lane-

34

change data should be collected by experienced drivers during the day with good weather conditions and low traffic volumes.
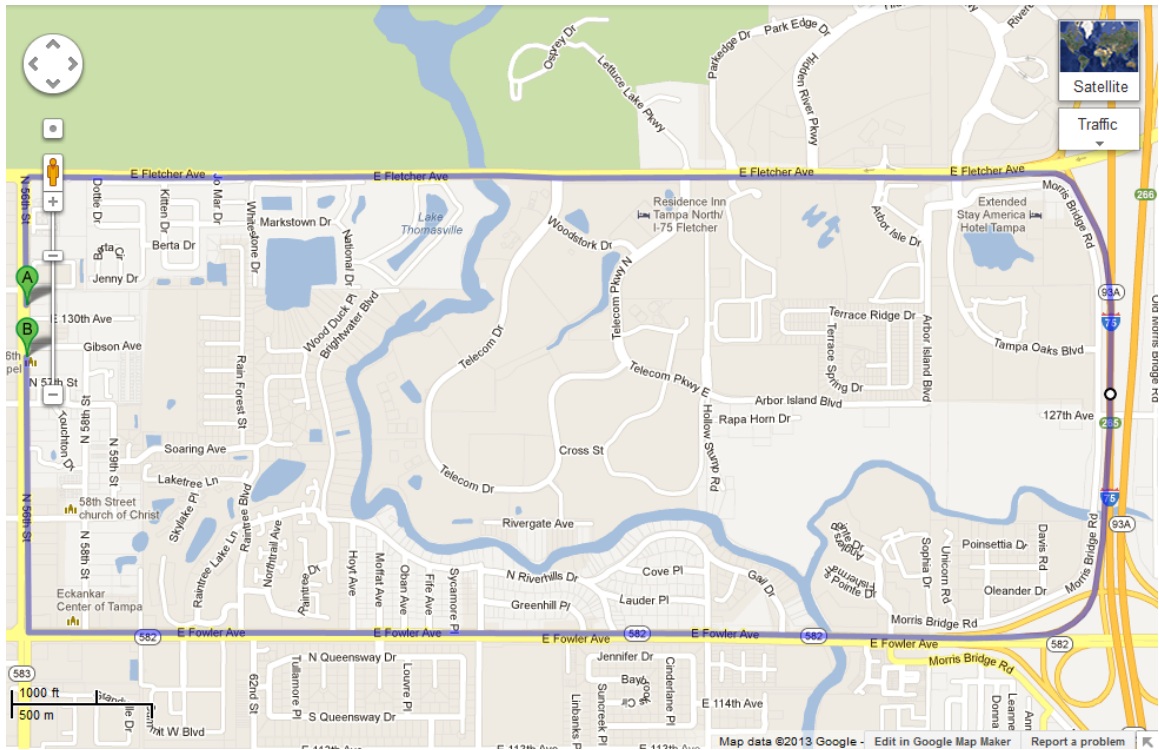
Figure 15 Selected Roadway Sections

4.2.2 Driving Data

Vehicle movement data collected by the GPS/IMU reflect driver behaviors. Multiple instances of a specific type of driver behavior are of similar patterns. Some featured data are as following.

A set of deceleration data for a vehicle is shown in Figure 16. The vehicle slowed down until it stopped for a red light on a straight roadway. In the data fragment, the vehicle traveled 236 meters in 19 seconds. Harder braking was applied in the last 5 seconds than the previous 10 seconds. The changes in vehicle speed and pitch, except for pitch angular velocity, are strongly correlated with longitudinal acceleration.

35
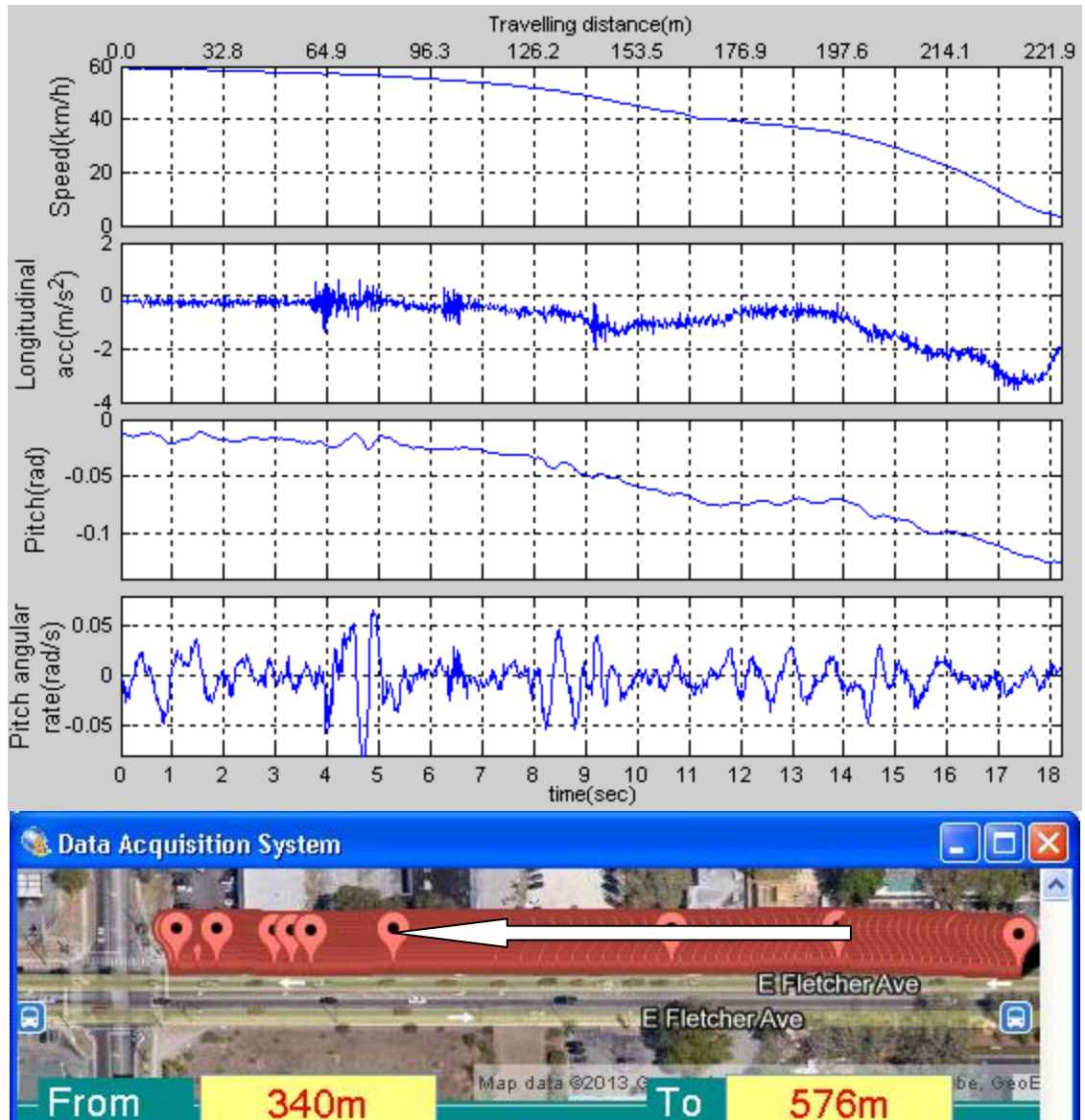
Figure 16 Deceleration

Figure 17 shows a vehicle accelerating from 0 to 60 km/h after waiting for a green light at the stop line of an intersection. During acceleration, the vehicle traveled 174 meters forward on a straight roadway in 17 seconds. Acceleration increased to peak value and lasted for 1.5 seconds at the beginning, and then decreased gradually to 0. As a result of longitudinal acceleration variance, the pitch angle of the vehicle rose significantly in

36

the first 8 seconds and then declined slowly. There is no obvious relevance between pitch

angular velocity and longitudinal acceleration.
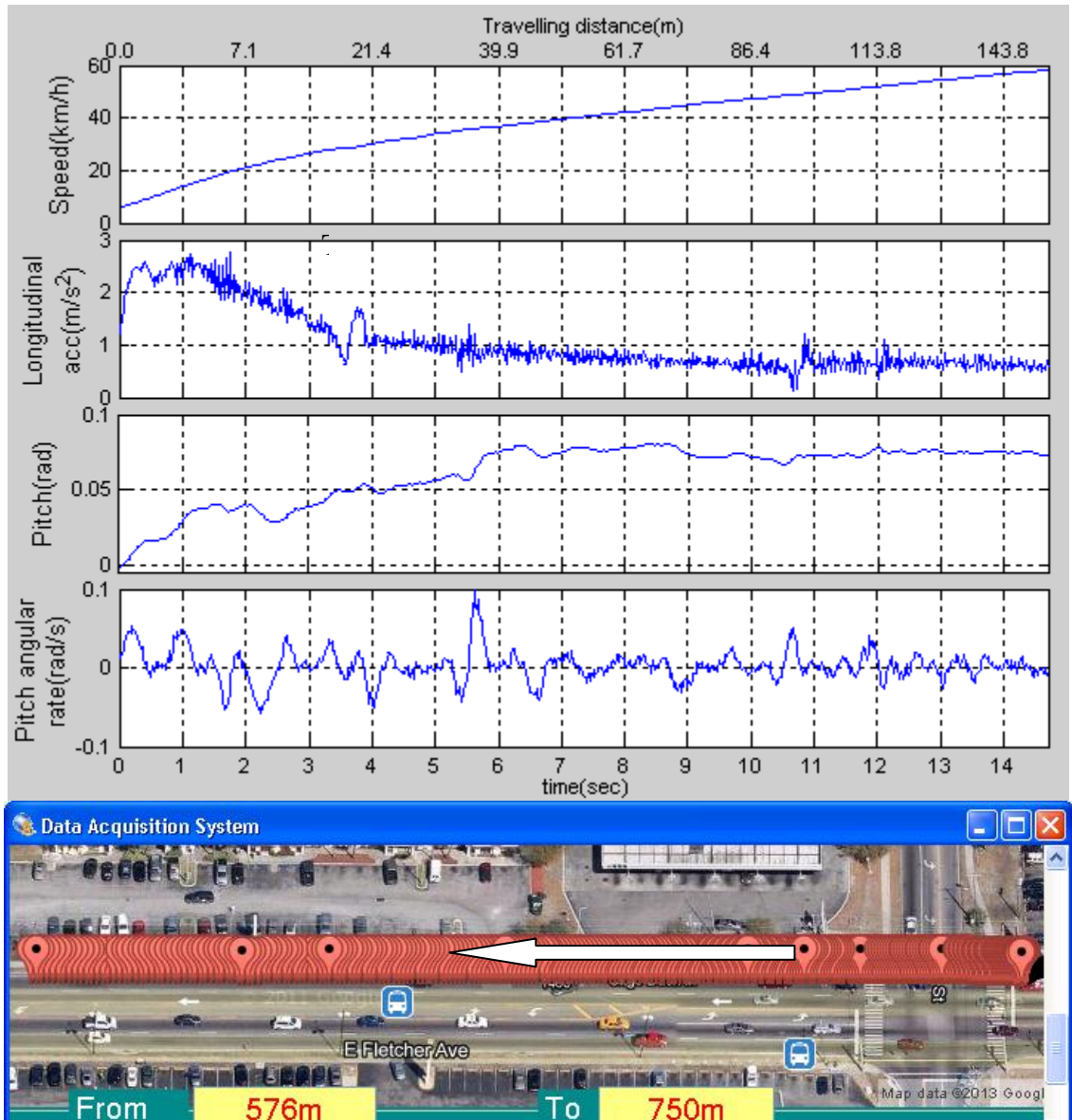
Figure 17 Acceleration

Figure 18 shows the result of a vehicle passing over a hump. The lateral

acceleration pattern in this instance is completely different from the ones of other

movements. The hump obviously influenced the pitch angle and pitch angular velocity of

37

the vehicle. The size of the hump and the speed of vehicle are the major factors for generating such patterns.
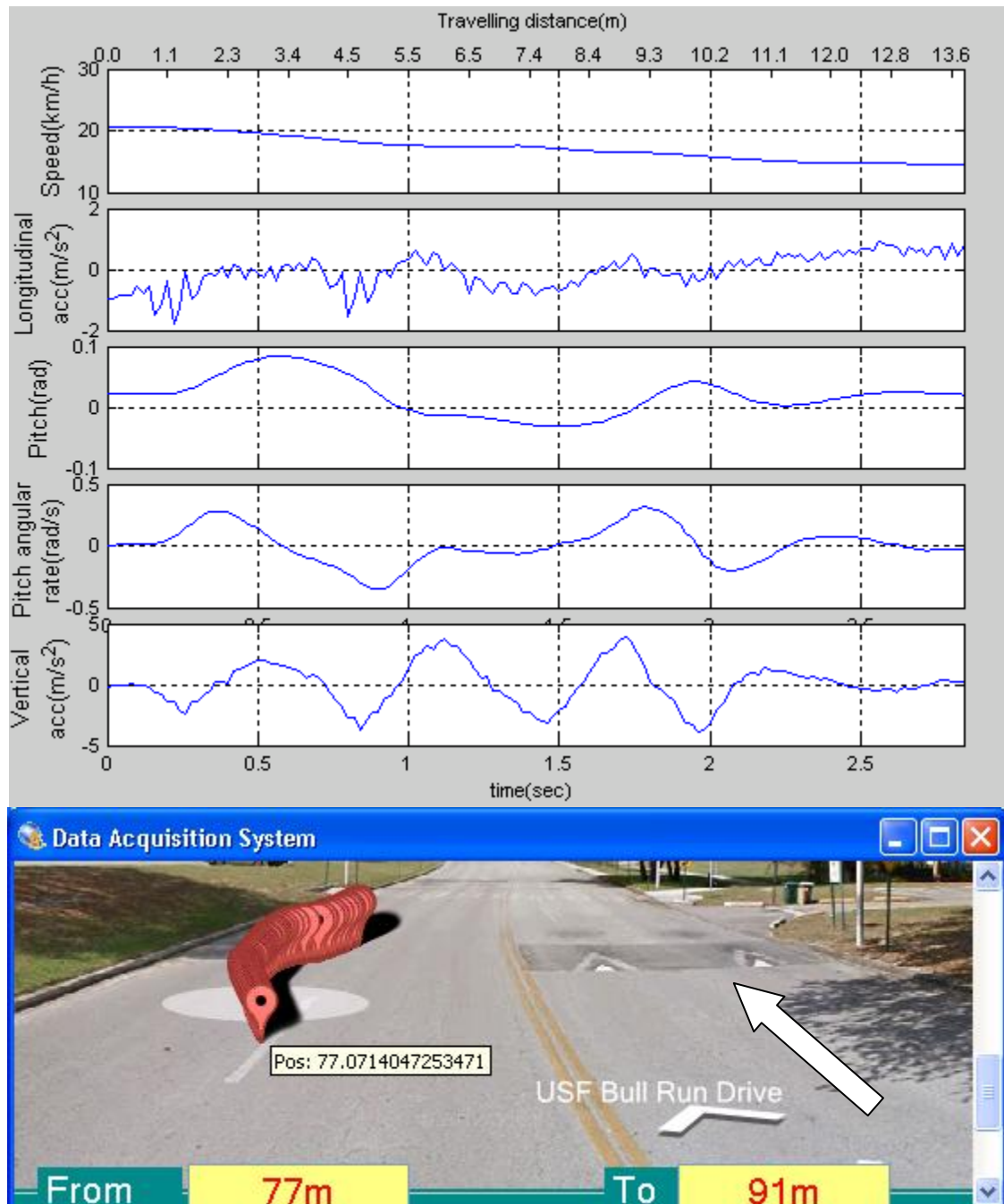
Figure 18 Hump Passing Over

Figure 19 shows a left-turn movement completed within 7.4 seconds. In the first 4 seconds, the brakes were applied before entering the intersection, with the steering wheel

www.manaraa.com

pulled slightly to the left. The most obvious changes in the curves happened in the last 3 seconds. Pitch angle and roll angle were significantly affected by the changes of speed and yaw angular velocity. In the whole process, the changes of lateral acceleration and yaw angular velocity were strictly synchronous. The fluctuations of pitch angular velocity, roll angular velocity, and lateral acceleration were caused by the uneven pavement.
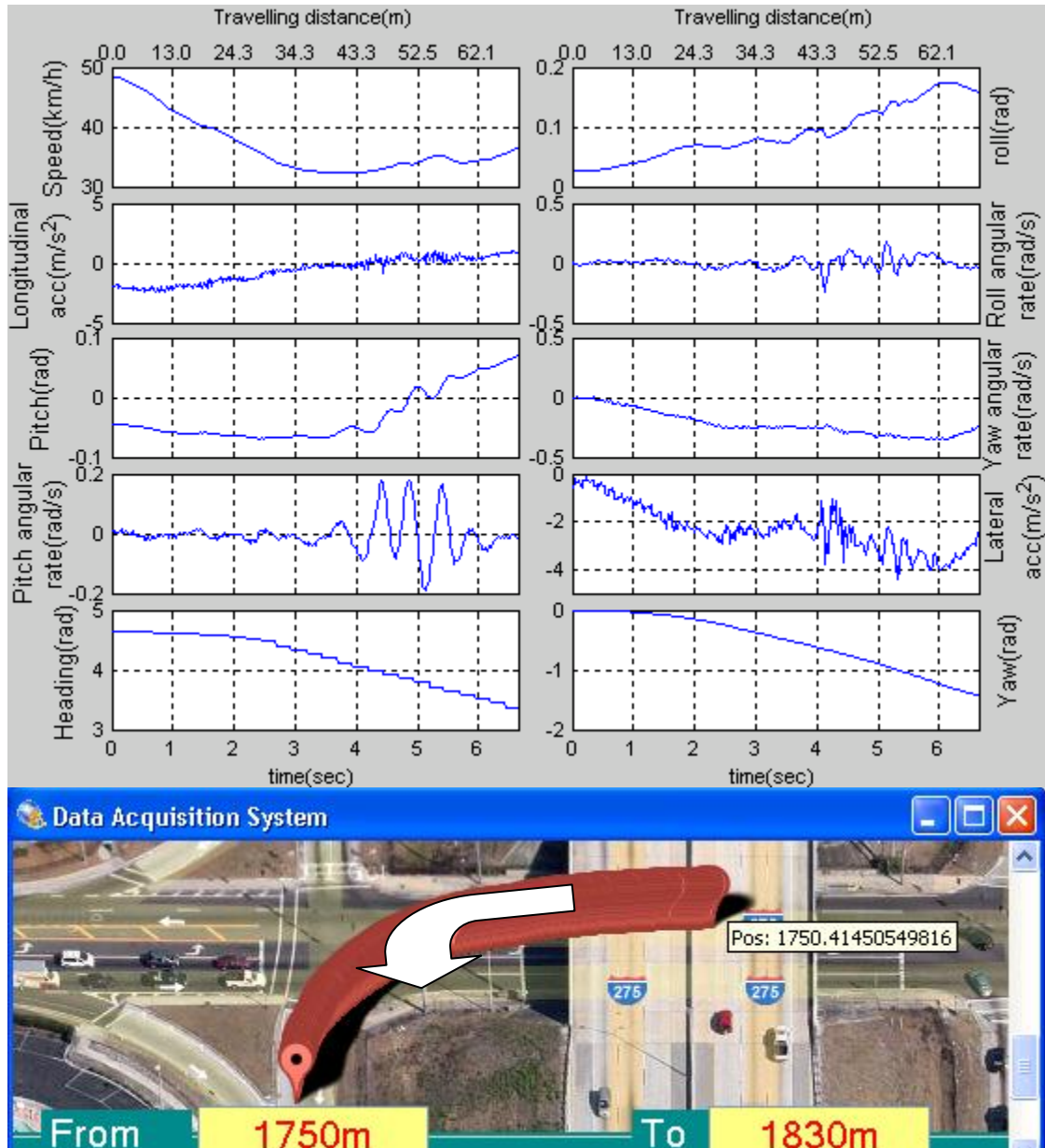
Figure 19 Left Turn

39

Lane-change patterns will be discussed in Chapter 6.

In summary, this chapter shows some correlations between features and normal driver behaviors; nevertheless, some defects of data collected by the IVDR can be easily found follows.

- The change of speed detected by the GPS lags behind that of longitudinal acceleration detected by IMU.

- The change of angle lags behind that of corresponding angular velocities.

- Yaw angle, roll angle and pitch angle take a while to attain stability after the driver behavior is completed.

- Raw data include high-frequency noise.

- Yaw angle sometimes changes unreasonably and unpredictably.

Chapter 5 Lane-Change Simulation

Sufficient lane-change data are essential to build a lane-change recognition model. Although an IVDR is developed in this dissertation, it is hard to collect a full-scale data set that includes lane-change data in different conditions, such as various speeds and distances. The most restrictive data collection issue in real traffic is safety. Frequently lane-change behavior comes with high risk, not only for the probe vehicle but also for other vehicles nearby. To avoid this, simulation is the best way to obtain sufficient lane-change data. A lane-change simulation model can imitate a human driver performing lane-change movement with given conditions. This chapter introduces a lane-change model that consists of a vehicle model, a speed controller, and a trajectory tracking controller.

5.1 Simulation Software and Vehicle Model

Appropriate simulation software can reduce the model development period and provide high quality simulation data efficiently. Simulink, a block diagram environment for multi-domain modeling and simulating, was used to develop the lane-change model. Its major interfaces are a graphical block diagramming tool and abundant block libraries. It is one of the most popular simulation tools widely used in control system design and digital signal processing. It provides a series of complete function tools for data input, data output and observation. The appearance of the Simulink development environment is shown in Figure 20. The PID control is quite easy to implement in the friendly interface of Simulink.

41
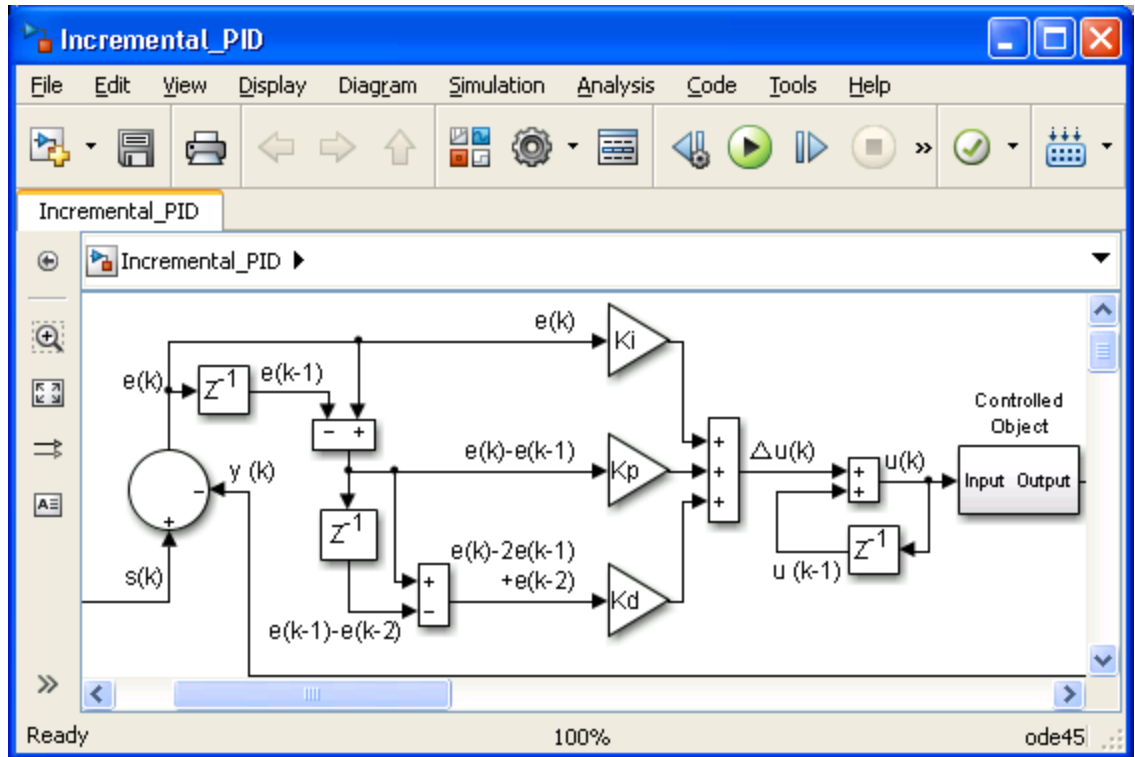
Figure 20 Simulink Interface

The vehicle model used in the lane-change simulation model was extracted from

the Automated Dynamic Analysis of Mechanical System (ADAMS). ADAMS is a

multibody dynamics software used for studying the dynamics of moving parts and how

forces and loads are distributed in mechanical systems. It is capable of simulating real

world physics. ADAMS provides a full-car model perfectly to meet the requirements of

developing the lane-change model. Unfortunately, there is no trajectory tracking

controller in ADAMS, so the full-car model was exported to the Simulink development

environment and integrated with a control system. Figure 21 shows the full-car model in

ADAMS, indicating four wheels, front and rear suspensions, an engine, a steering system,

and a body. A brake system is also integrated into the full-car model.

42
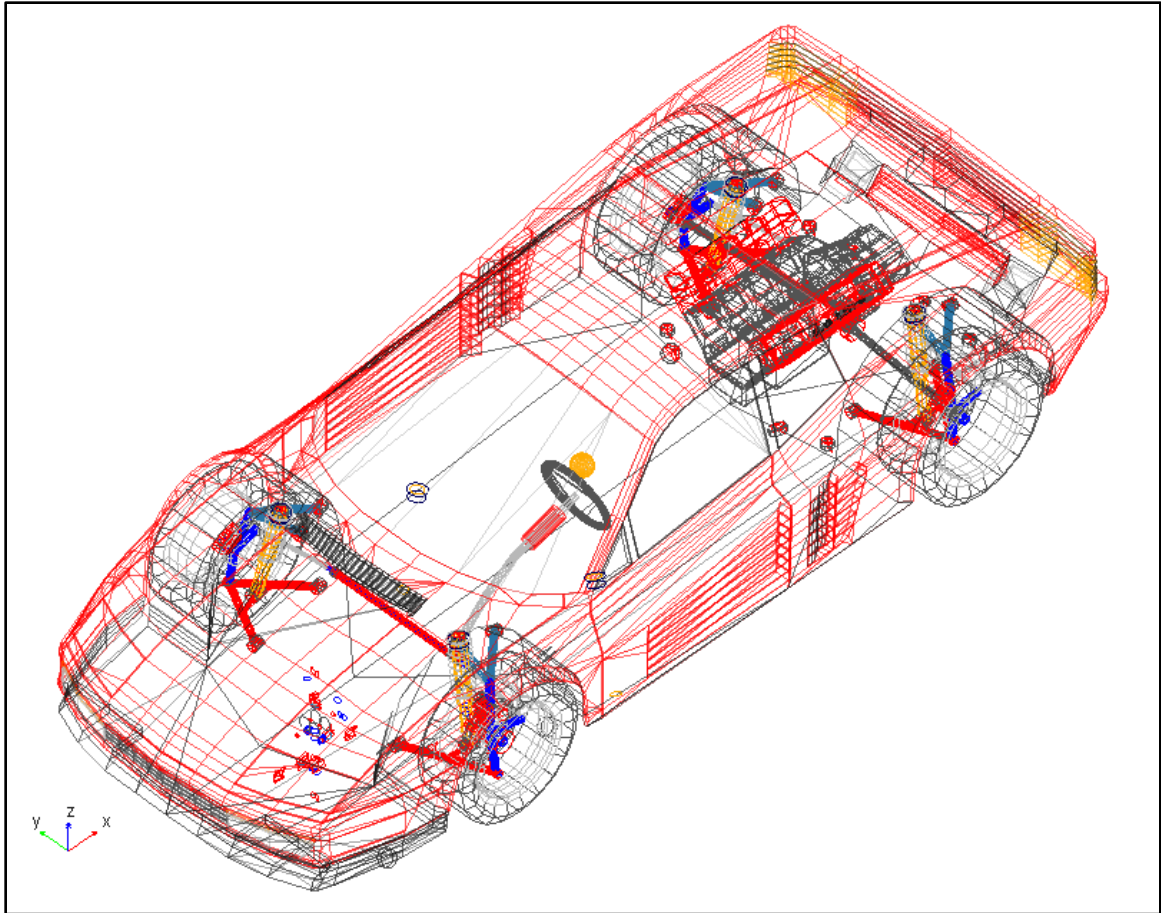
Figure 21 ADAMS Full-car Model

In Simulink, the imported full-car vehicle model is shown as a block diagram, with multiple inputs and multiple outputs. In Figure 22, the five elements on the left side of the model are full-car model inputs, similar to the five parts controlled by a human driver in a real manual vehicle; the elements on the right side are the sensors integrated into the full-car model, which provide accurate vehicle operation information, such as displacements, velocities, accelerations, etc. The coordinate system comforts to the right hand rule, and the x axis points to the advance direction of the straight road along the horizontal plane. The data collected from this full-car model are similar to the field data collected by the IVDR developed in this dissertation.

43
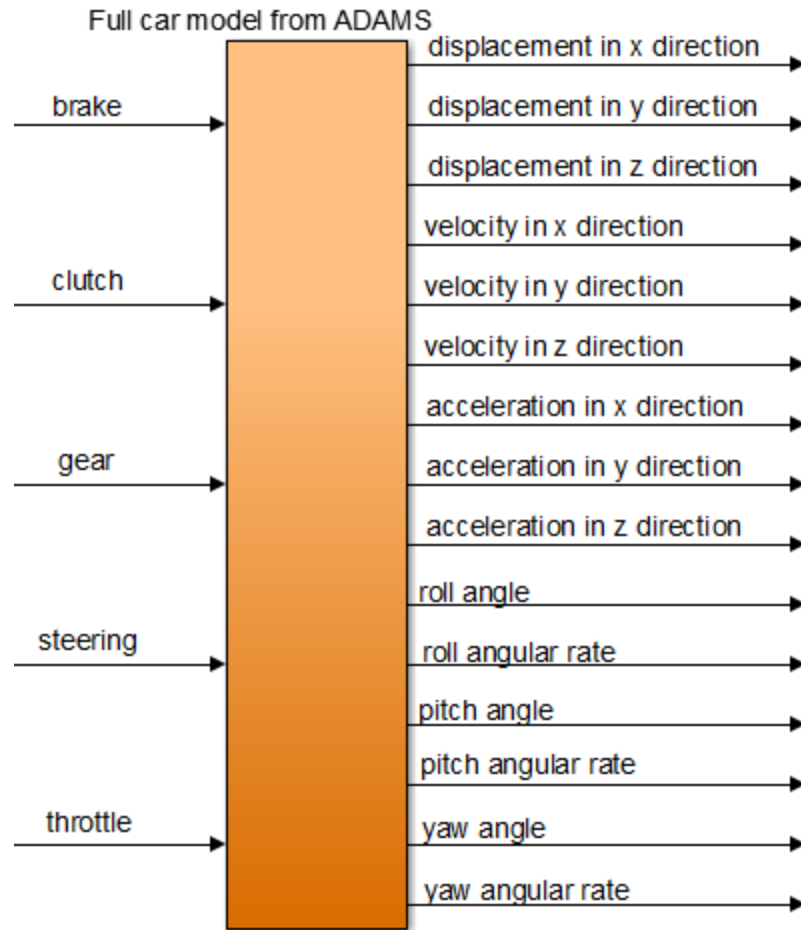
Figure 22 Full-Car Model in Simulink

According to the input features of the full-car vehicle model, a speed controller is designed to control the brake and throttle, and a trajectory tracking controller allows for steering input. These two controllers have the capabilities of tracking target lane-change paths and keeping a steady speed as the setpoint.

5.2 Speed Controller

The lane-change model should simulate lane-change movements at different given speeds. The initial speed of the full-car vehicle is fixed in this case, but it can be adjusted to the given speeds and kept steady by the speed controller. In actual driving tasks, a lane-change movement can last longer than 10 seconds. Considering that most

44

lane-change behaviors are completed in 10 seconds and the precision of the GPS/IMU is not good enough to detect small signal changes in slow lane-change movements, this dissertation focuses only on the lane-change data collected in 10 seconds. However, speed could change significantly without any speed control measures, even in a period less than 10 seconds. That will affect the lane-change data quality that should be collected at different speeds, other conditions remaining the same.

5.2.1 PID Speed Controller

To achieve a speed as a setpoint, the speed controller adjusts the throttle and brake of the full-car model, according to the computation results on speed error. The speed control loop is shown in Figure 23.
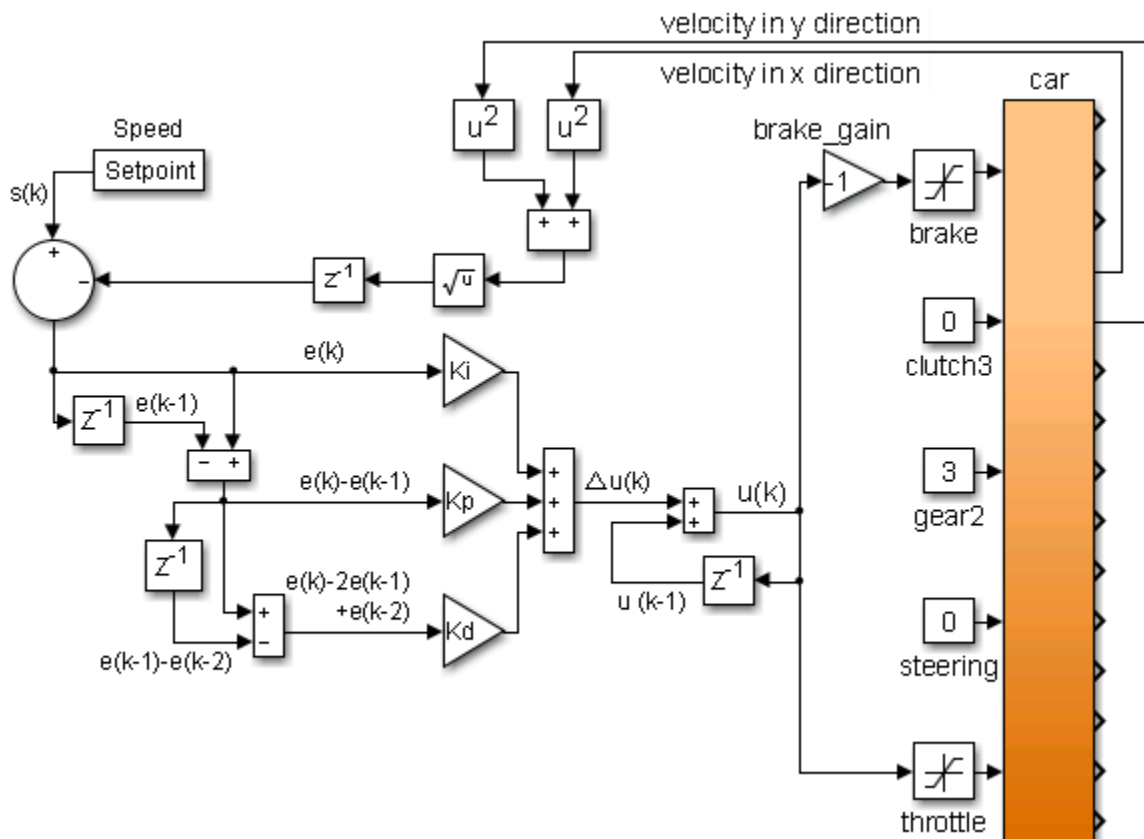

Figure 23 PID Speed Controller

45

In the block diagram, z^{-1} is a delay unit used to output the value calculated in the last computation cycle, and $\not{F}$ is used for limiting the input signal to the upper and lower saturation values. The vehicle is assumed to run on a zero grade pavement to ignore the vertical velocity of the vehicle, and the impacts on vehicle velocity from roll angle and pitch angel are also ignored. The control purpose is to stabilize the magnitude value of synthetic speed of the velocities in x and y directions (Vx and Vy).

The input ranges of the throttle and brake are between 0 and 100 percent, 0 indicates no input and100 is the other extreme. For the control value u(k), a negative value indicates deceleration, and a positive value indicates acceleration. Although the u(k) is connected to two inputs of the vehicle, only one input is available at the same time. Three parameters, Ki, Kp, and Kd, have to be determined for future implementation.

5.2.2 Speed Controller Parameters Tuning

The full-car model allows the user to set an initial speed and the simulation will starts at the given speed. Using a high initial speed will decrease the time length before the speed becomes steady. Decelerating takes less time than accelerating to achieve the same speed difference, and starting at zero speed require shifting the gear which makes the simulation more complicated. In this case, the initial speed was set at 20 meters per second (m/s). The simulation step is the time interval between each computation cycle; a large interval saves simulation time with lower control quality, and a small interval helps the controller to frequently read the error and adjust the controlled object. To achieve a balance, the simulation step was set to 0.025 seconds.

First, the valid value ranges of the PID parameters had to be determined. Usually, the lower bounds for all PID parameters are 0. PID parameters in different orders of

46

magnitude have to be set on the controller, and the upper bounds are decided according to the response of the vehicle speed. The following figures show the speed responses upon different combinations of PID parameters.

Comparing the actual speed curves in Figure 24 and Figure 25, the second one does not have overshoot at 1 second but oscillates below the given value in the last 4 seconds. The first keeps a steady speed after a few overshoots. The first response is preferred, and the upper bounds of the PID parameters should be less than 1.
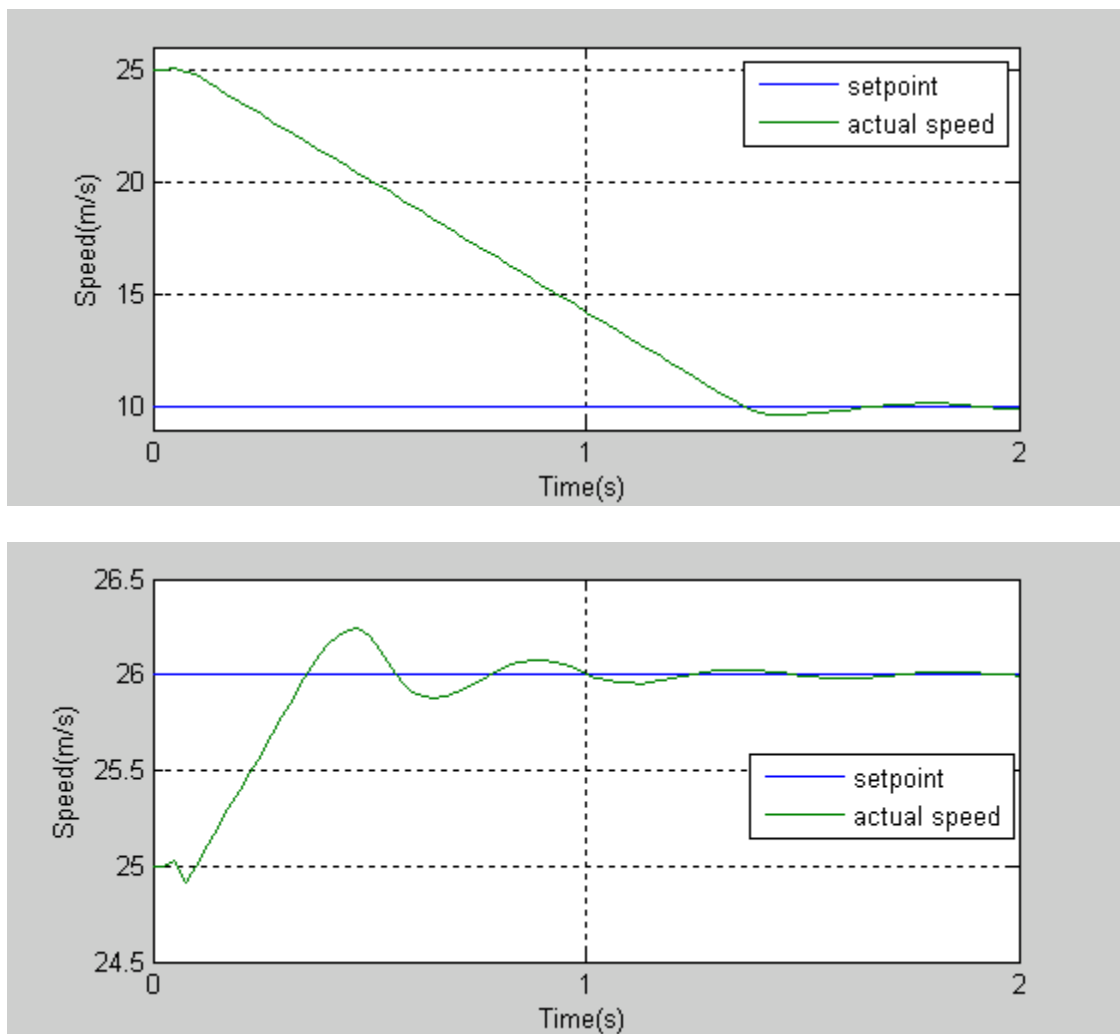
Figure 24 Speed Response, Kp=0.1, Ki=0.1, Kd=0.1
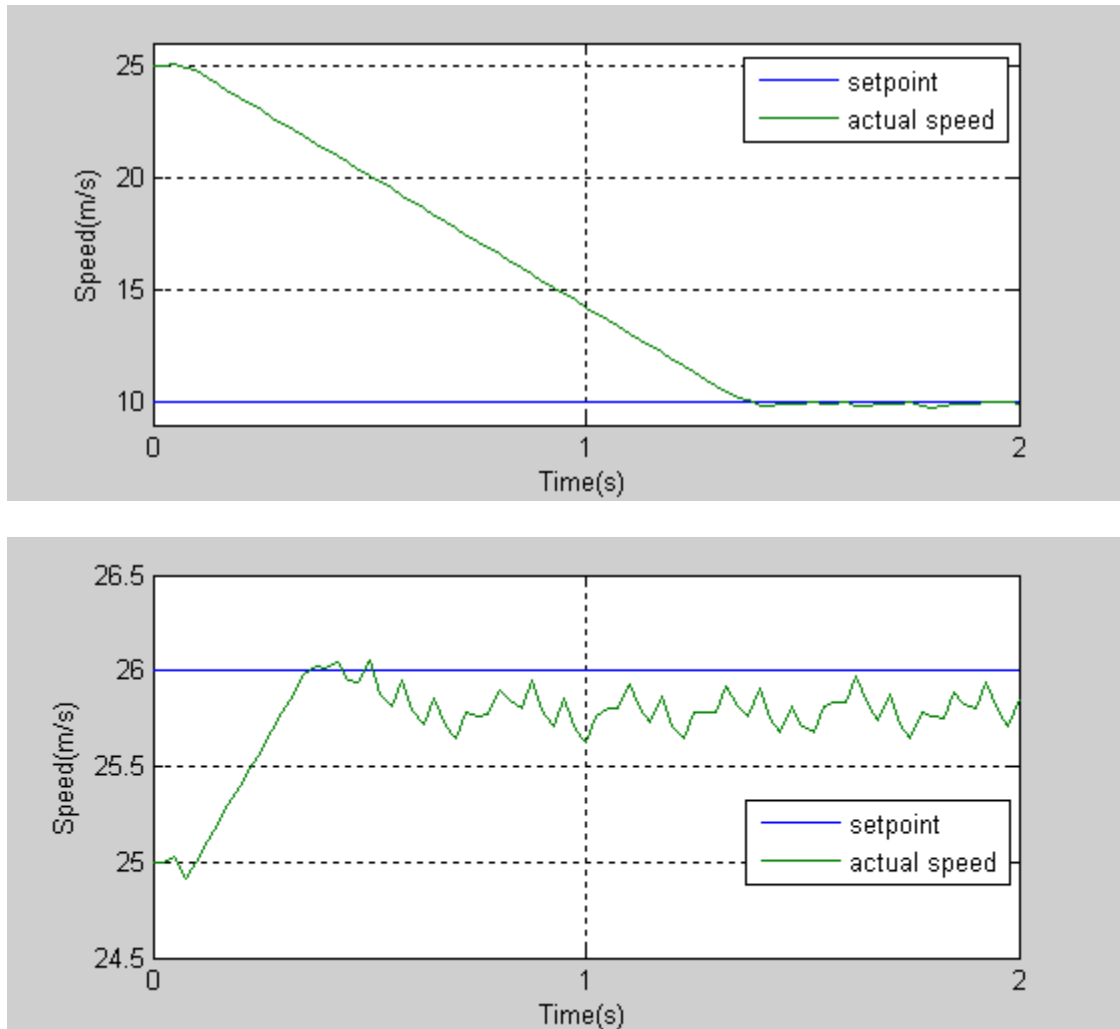
Figure 25 Speed Response, Kp=1, Ki=1, Kd=1

The next step is to tune the PID parameters around (0.1, 0.1, 0.1) and determine if there are any responses better than the ones shown in Figure 24. For convenience, GA was used to optimize PID parameters in the speed controller. The GA optimization process requires two basic inputs in this case the number of parameters and the range for each of them and also needs an evaluation index for each combination of parameters generated by the GA. The evaluation index was calculated in each simulating cycle according to the simulation result, and the GA aims to minimize the evaluation index.

48

In academic journal papers and simulation studies, the control measures are used to measure the controlled system performance for comparing different control schemes or different groups of controller parameters. They are also very useless for assessing the performance of real control system. One of the most commonly used measures is Integral Time-weighted Absolute Error (ITAE), and is defined in Equation 4.

$$ITAE = \int_0^{t_1} |e(t)| \, t \, dt \tag{4}$$

where,

e = error

t_1 = upper limit of integral

Equation 4 indicates that the ITAE index is the integral of time multiplied by the absolute magnitude of the error. The errors that exist after a long time are weighted more heavily than those at the beginning of the response. The features of the ITEA meet the requirement of tuning the PID parameters to keep steady speeds in lane-change movement. Because the lane-change movement is accomplished only after the vehicle speed is adjusted to the setpoint, ITAE emphasizes the errors occurring later in the speed response.

In the tuning process, shortening the simulation time and eliminating the impact of the errors occur before the vehicle speed reaches the setpoint, so the initial speed of the vehicle was set as 25m/s and the setpoint was 26m/s, and the simulation time was set to 2 seconds. A full-throttle input was applied to investigate if the setting as reasonable for parameter tuning; the results are shown in Figure 26. As indicated in the figure, it takes only about 0.25 seconds to reach 26m/s from 25m/s by 100 percent throttle input, so the PID controller takes only about 0.5 seconds to accelerate the speed to 26m/s The

49

remaining 75 percent of the simulation time is enough for ensuring that later errors

dominate the ITAE result.
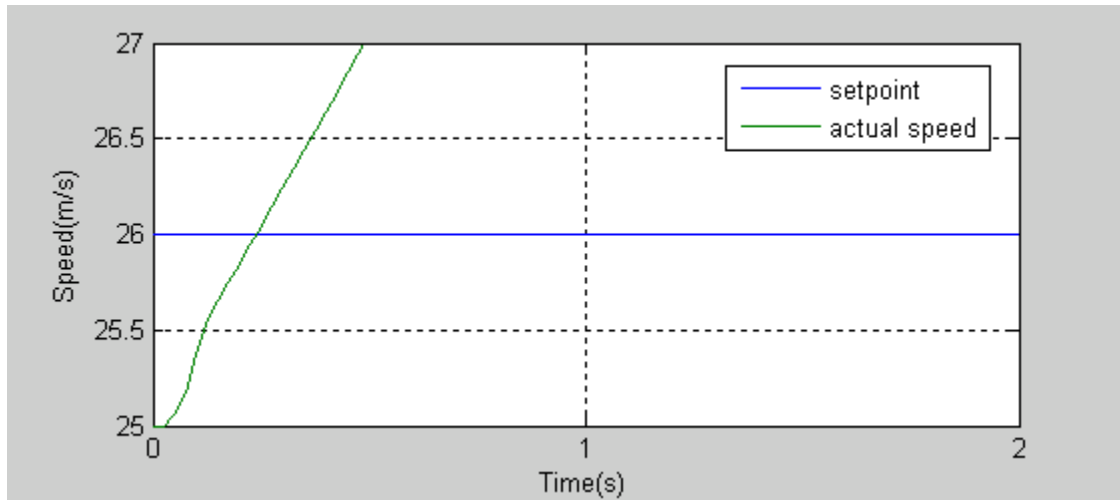
Figure 26 Speed Response, Throttle=100%

For the GA setting, the number of undetermined parameters is 3, and the lower

and upper bounds of all the parameters are 0 and 1. GA searches only for a good

combination, not the best one, so it was not necessary to set a large generation parameter

to the algorithm, which would result in a long optimization period, so only 1 generation

was set the GA. The ITAE integral range was set from 0 to 2 seconds as the full

simulation period. The GA optimization result is shown in Figure 27. The best ITAE
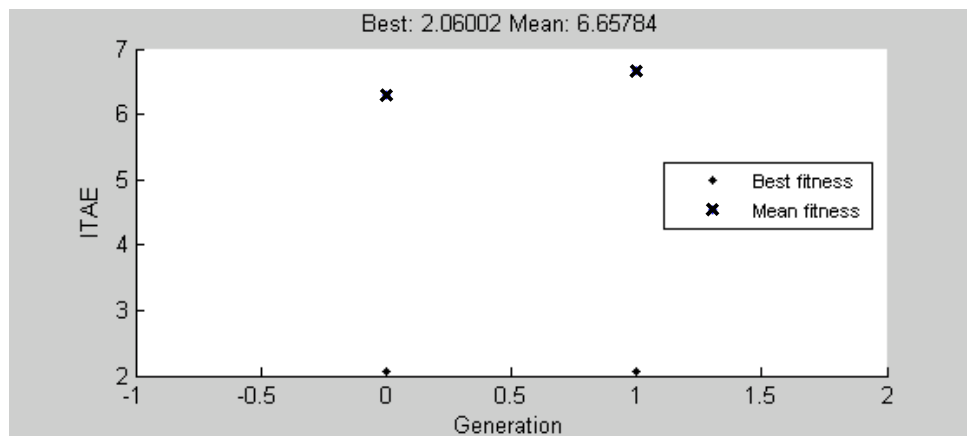
Figure 27 GA Result on PID Speed Controller

50

generation 0 and generation 1 were 2.06; the corresponding combination of parameters was the best and was used in the PID speed controller.

GA also lists all 40 combinations of parameters generated in the optimization process. The last outputs are presented as follows.

- Kp=0.10, Ki=0.06, Kd=0.74, ITAE=8.72

- Kp=0.72, Ki=0.42, Kd=0.55, ITAE=11.71

- Kp=0.82, Ki=0.40, Kd=0.95, ITAE=22.79

- Kp=0.89, Ki=0.52, Kd=0.29, ITAE=8.33

- Kp=0.50, Ki=0.97, Kd=0.43, ITAE=7.09

- Kp=0.24, Ki=0.66, Kd=0.34, ITAE=2.18

- Kp=0.71, Ki=0.43, Kd=0.11, ITAE=2.06

- Kp=0.06, Ki=0.13, Kd=0.49, ITAE=5.96

- Kp=0.18, Ki=0.15, Kd=0.66, ITAE=3.39

- Kp=0.98, Ki=0.45, Kd=0.17, ITAE=7.45

- Optimization terminated: maximum number of generations exceeded.

- Bestresult : Kp=0.71, Ki=0.43, Kd=0.11, best ITAE=2.06

In the list content, the combination Kp=0.24, Ki=0.66, Kd=0.34 has a small ITAE value, which is close to the best one. The combinations that have small ITAE values are already good enough for the speed controller in this case. GA makes the optimization much easier than the manual method which is a time consuming process.

51

5.2.3 Speed Controller Performance

The three parameters with the best combination were applied to the PID speed controller. The throttle inputs, brake inputs, and speed responses of the acceleration and deceleration process are shown in Figure 28 and Figure 29.
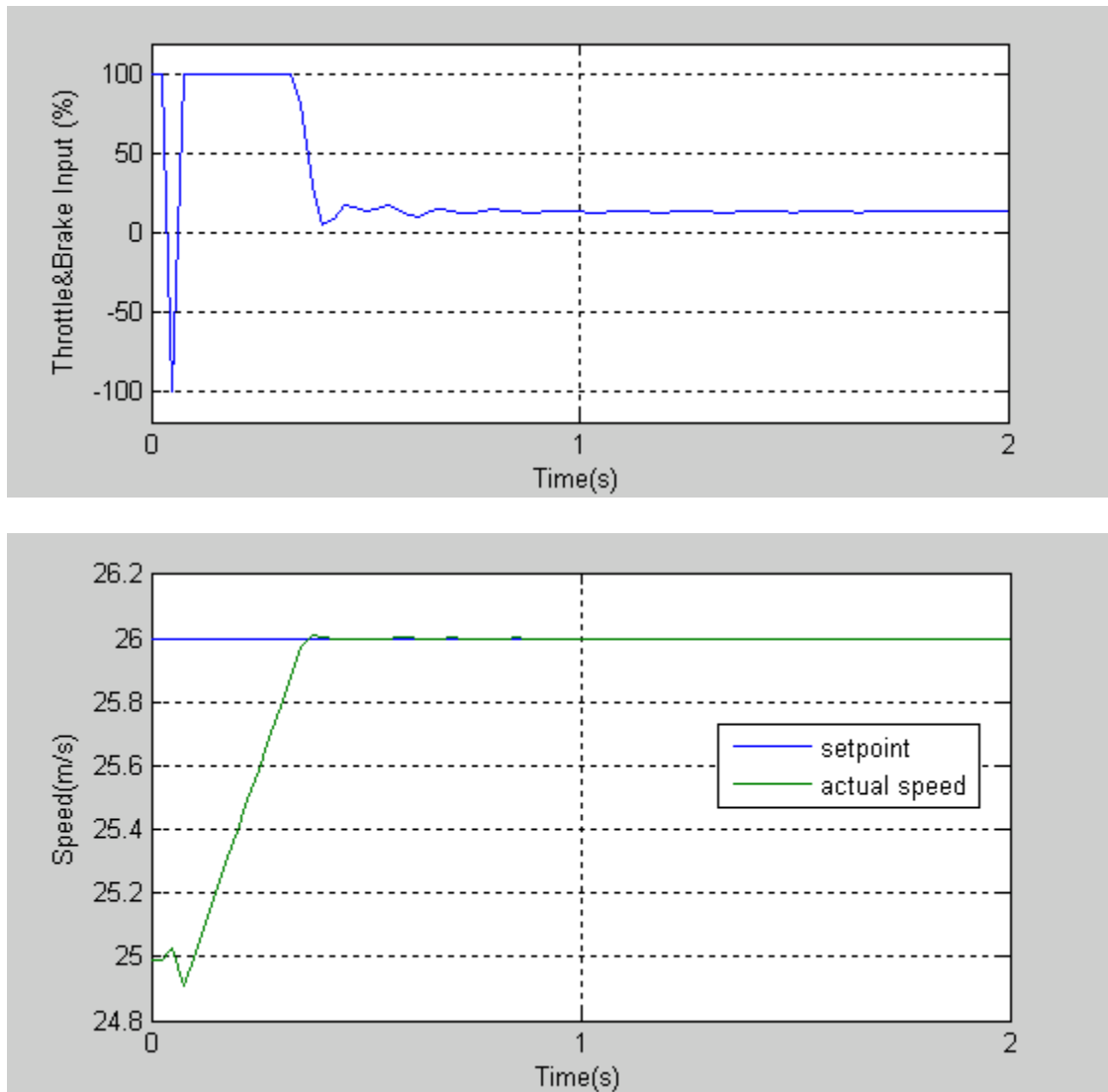
Figure 28 Acceleration, Kp=0.71, Ki=0.43, Kd=0.11

52
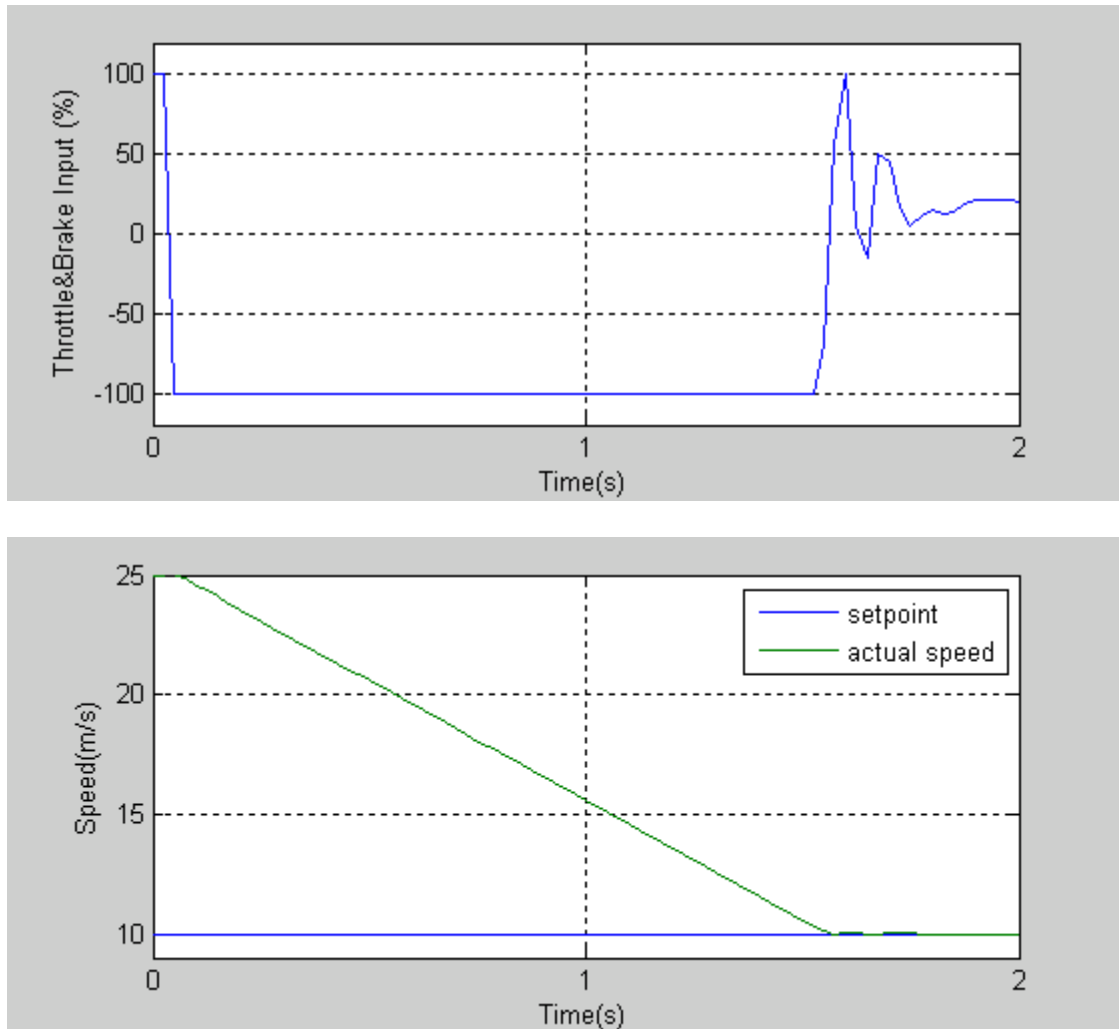
Figure 29 Deceleration, Kp=0.71, Ki=0.43, Kd=0.11

Figure 28 indicats that the vehicle speed is pushed by the 100 pecent throttle input in the first 0.4 seconds and rapidly climbs to the setpoint. Once the vehicle speed gets close but is a little slower that 26m/s, the PID controller reduces the throttle input to avoid a large overshoot, and then adjusts the throttle input within a small range to keep the vehicle speed steady. At about 0.8 seconds, the adjustment gets smaller, and the vehicle speed is remained by a stable throttle input.

Figure 29 shows the deceleration process with the same parameters as those in the acceleration process. The speed controller takes about 1.5 seconds to reduce 60 percent of

53

the initial speed with a 100 percent brake input. A large overshoot is eliminated in this process, but both brake and throttle are used to adjust the vehicle speed during the period between 1.5 - 1.9 seconds.

5.3 Trajectory Tracking Controller

For tracking the given lane-change path, there are two essential capabilities that the trajectory tracking controller must have: one is sensing the path information ahead of the vehicle, the other one is controlling the steering wheel according to sensed information. These two capabilities are also two basic skills that a good human driver has. The most ideal controller should be able to drive exactly along the given path.

5.3.1 PID Trajectory Tracking Controller

The trajectory tracking controller is a classic PID controller similar to the speed controller. There are two important features in the tracking controller that are not in the speed controller: a preview function and position feedback. The throttle and brake can be adjusted by the controller from 0 to 100 percent in one simulation step, simulating an actual situation. But for both the full-car model and a real vehicle, the steering wheel cannot be changed much in a short period, especially in 0.025 second. Compared to speed, the position of the vehicle cannot be adjusted so rapidly, because it involves other variables more than position, such as acceleration, speed, heading, etc. Thus, it requires that the controller be able to perceive the information in front of the vehicle similar to the preview capability of a human driver. The other difference from the speed controller is that the tracking controller needs information about car position, speed and acceleration. With more valid and necessary information are sent to the controller, the performance of the controller adjusting the vehicle will be better.

54

In Figure 30, a driver is attempting to reach point B at time $t + \Delta t$. At point A, the vehicle has an initial acceleration in y direction a_y and an initial speed in y direction v_y. If the driver keeps the same steering wheel angle, the vehicle only can reach point C at time $t + \Delta t$, so the driver has to turn the steering wheel to the left a little more to force the vehicle to the left. The adjustment of the steering wheel generates an extra acceleration in y direction a_{y1}, which helps the vehicle reach point B. The adjustment depends on the driver's experience, which can reduce the position error. In this case, the driver sees the destination in his sight distance, then adjusts the steering wheel based on the vehicle position, velocity and acceleration (by vehicle heading). Base on this analysis, the tracking controller should be able to detect point B early enough and provide a lateral acceleration difference value by adjusting the steering wheel. The difference between a human driver and a controller is that a human driver can determine when to respond to a destination ahead of the vehicle, but a controller adjusts the steering wheel once it detects the feedback information. The feedback to the tracking controller is defined in Equation 5.

$$feedback(k) = y(k) + v_y(k) \cdot \Delta t + \frac{1}{2} a_y(k) \cdot \Delta t^2 \tag{5}$$

where,

Feedback = input of the tracking controller

y = vehicle position in y direction

v_y = vehicle speed in y direction

a_y = vehicle acceleration in y direction

Δt = track controller preview time

55
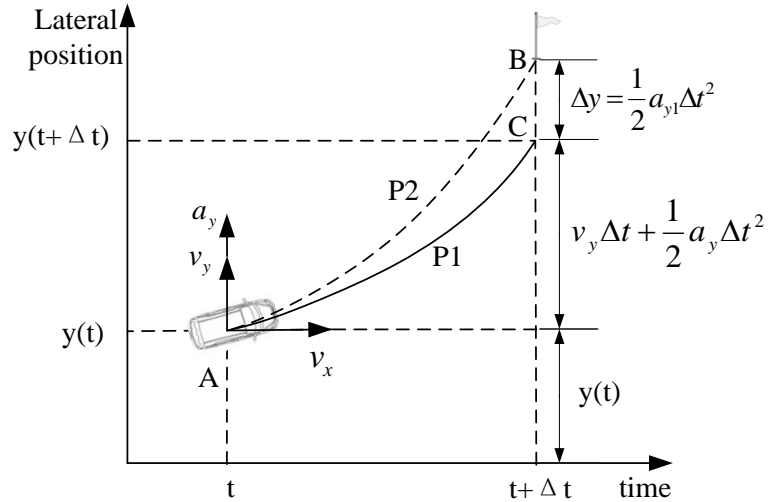
Figure 30 Feedback Information

To drive along a given path, the controller should at least have a constant sight distance. By a given preview time, a sight distance that is too long will make the controller act early, and the driving path will occur before the given path; if it is too short, the driving path will occur behind the given path. The best way to couple the preview time with the sight distance is to use vehicle speed multiplied by preview time.
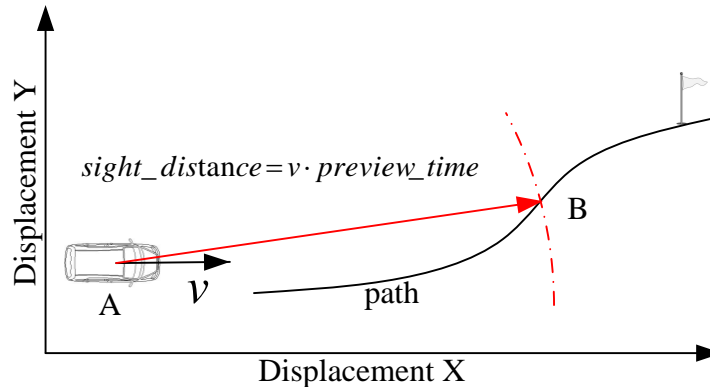
Figure 31 Sight Distance

The trajectory tracking controller is shown in Figure 32. The preview detector reads the given path and generates a preview path as a setpoint. The vehicle position, speed and acceleration in y direction are combined in Equation 5 as one part of the PID

56

controller input. Because the units of feedback information and steering input are at different scales, the output of the controller has to be multiplied by a conversion value.
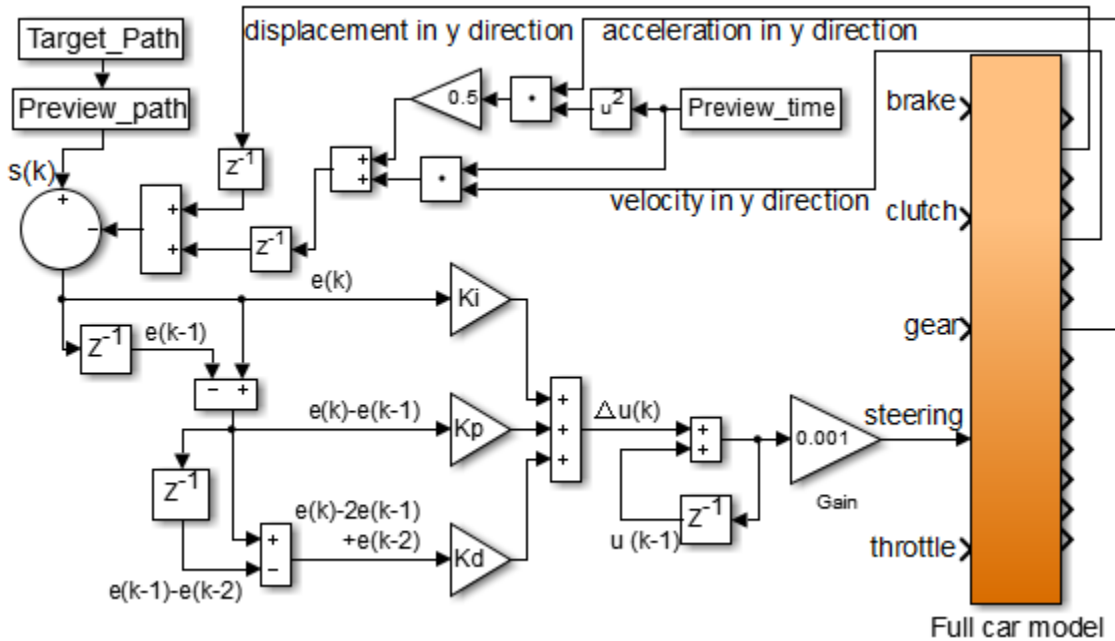
Figure 32 Trajectory Tracking Controller

5.3.2 Trajectory Tracking Controller Parameters Tuning

Usually, a controller that is capable of handling a severe situation has good performance in general, but not vice versa. A severe lane-change movement should be considered first in PID parameter tuning.

British Standard ISO 3888 provides a standard method of using a double lane-change to evaluation vehicle dynamics. The dimensions of a double lane-change track are shown in Figure 33. This standard suggests that vehicle speed on a double lane-change track is (80±3) km/h; higher or lower speeds may be used. In Florida, the speed limit for freeways in suburban and rural areas is higher than 110km/h and 88 km/h for highways. Field data may come from these roadways, so the highest speed for the lane-change simulation model is set to 110km/h.

57
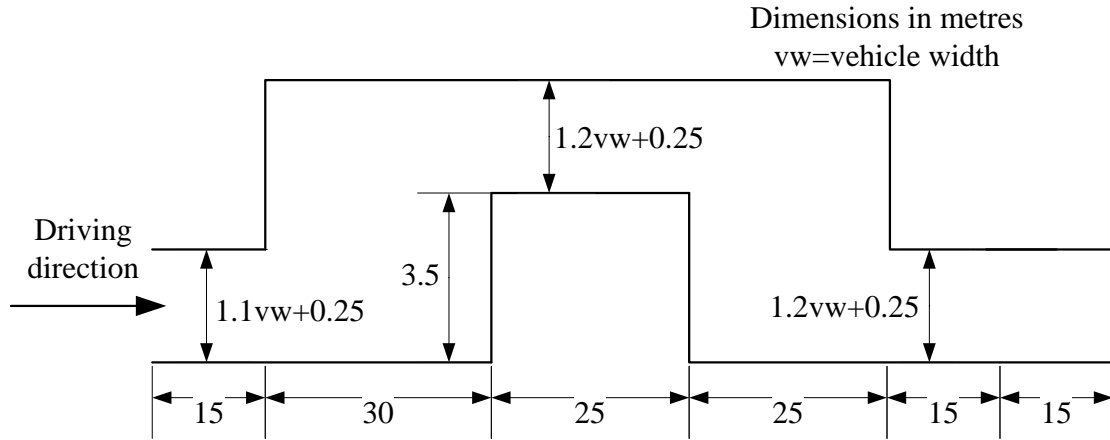
Figure 33 Double Lane-Change Track

For a double lane-change track, the second single lane-change is the most severe. Thus, in parameter tuning, the tracking controller must control the vehicle to accomplish a lane-change movement on the second lane-change track at 110km/h. The probe vehicle was a 1999 Corolla with a 1694mm body width, which is defined as the overall width of a vehicle without rear view mirrors. To simplify, all the tracks for the PID data tuning were 3.5m in lane offset and 25m in x direction, without lane width restrictions, as shown in Figure 34. The controller preview time was one second in the tuning process.
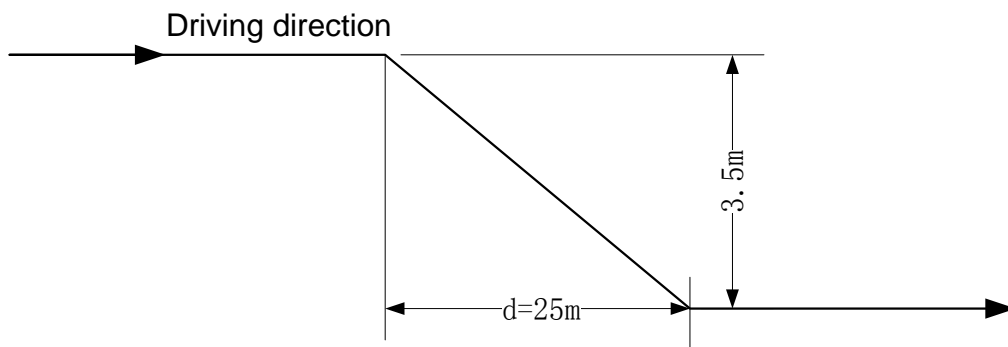
Figure 34 Lane-Change Track for Parameters Tuning

As with the speed PID parameter tuning, the first step was to determine the upper bounds of the three PID parameters. The test results are shown in the following figures.

58

Figure 35 shows a successful lane-change movement. Symbol d is defined as that in Figure 34, v is speed and pt is preview time. Because of the preview behavior of the PID controller, the vehicle attempts to move to the right lane at time=1s, although the target path has the first change at time=2s. A smaller preview time may force the actual path closer to the target path. There is a little overshoot between t=4s and t=5s; the vehicle still runs inside the lane. There is a small oscillation, showing only on the speed curve. The steady speed means the speed controller performed well in the severe lane-change movement.
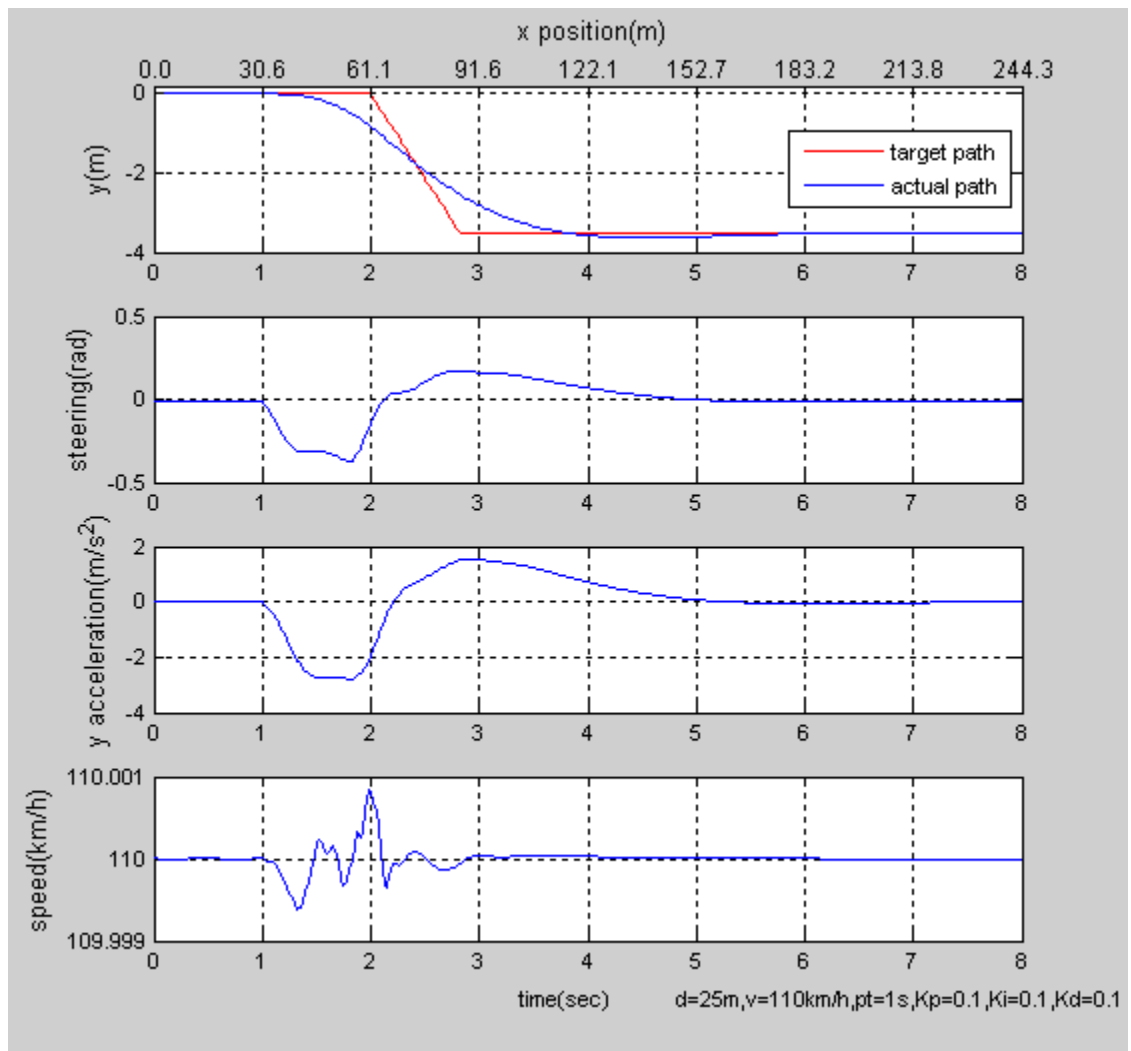
Figure 35 Lane-Change Simulation, Kp=Ki=Kd=0.1

59
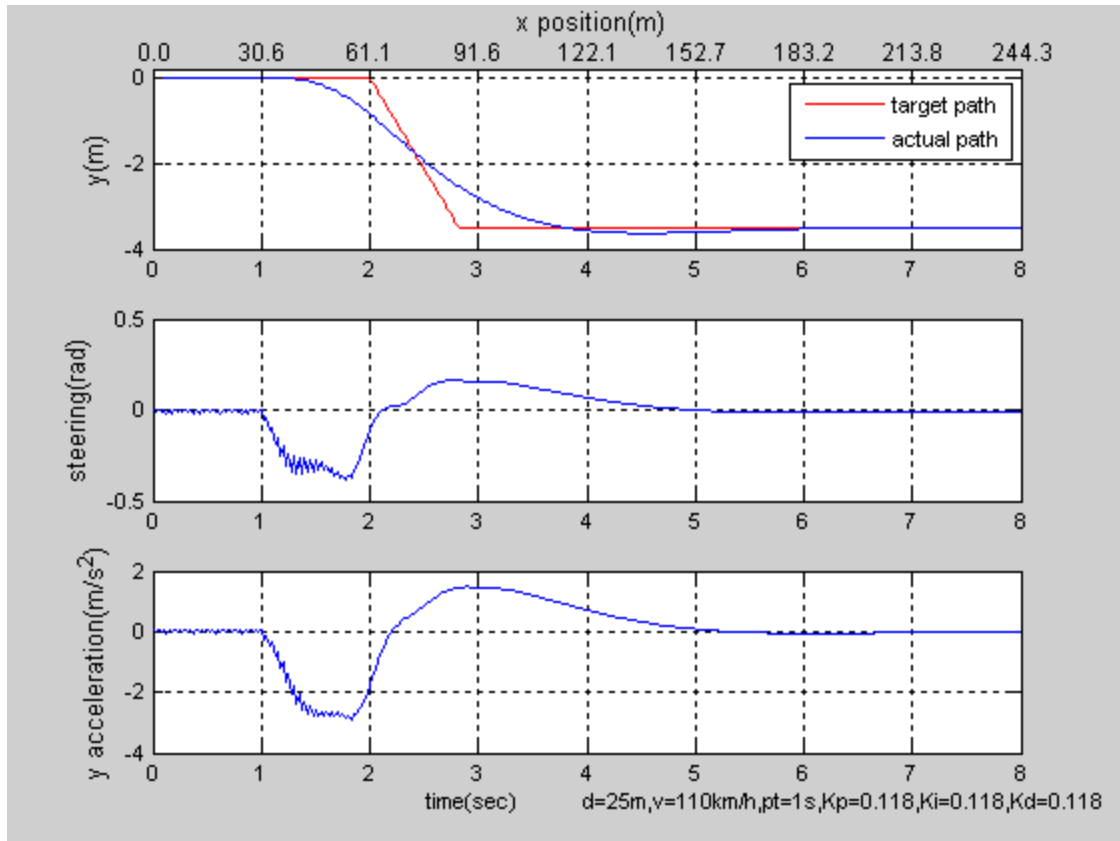
Figure 36 Lane-Change Simulation, Kp=Ki=Kd=0.118

When the parameters were all 0.118, the steering wheel input had obvious

oscillations, which means the parameters were close to the upper bounds.

Three parameters will make the optimization more complicated than fewer

parameters. The next step was to check if there were one or two parameters that dominate

the controller performance. Base on the case shown in Figure 35, two of the three

parameters were set to 0; the effect of the remaining parameter was observed through the

simulation response. The actual path in Figure 38 is very similar to the one in Figure 35

and the vehicles in Figure 37 and 39 are failure to tracking the given paths. The

parameter Ki dominates the trajectory controller to make a lane-change, and the

remaining two parameters impact only the steering pattern. In the remainder of the PID

60

tuning, the Ki is the only parameters needed to be optimized. The upper boundary of Ki

was determined again.
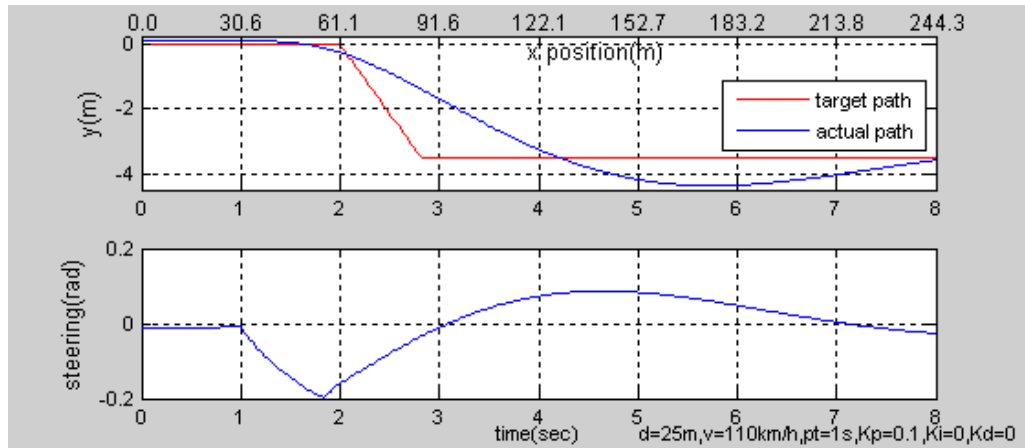
Figure 37 Lane-Change Simulation, Kp=0.1
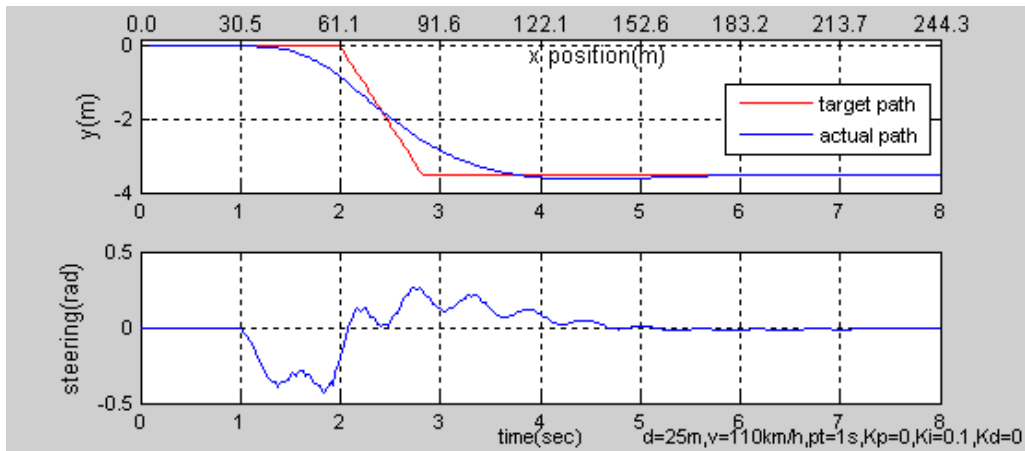
Figure 38 Lane-Change Simulation, Ki=0.1
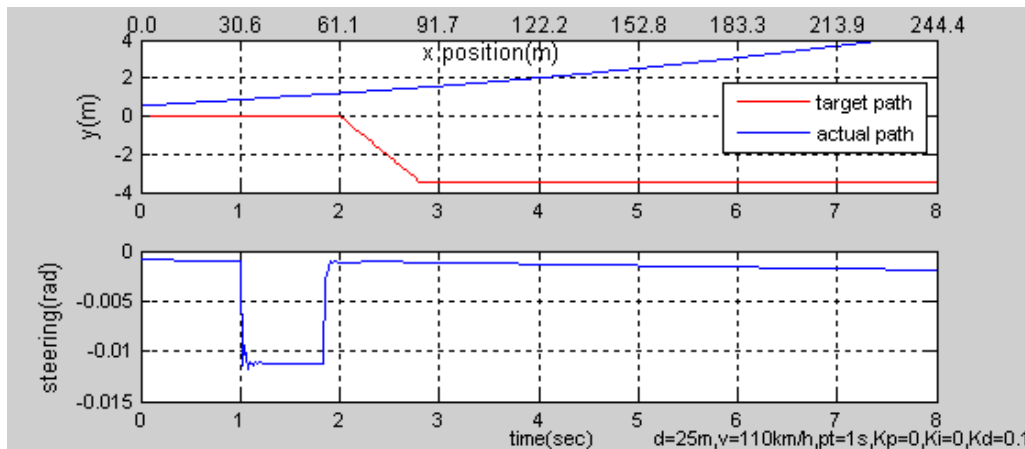
Figure 39 Lane-Change Simulation, Kd=0.1

61

When Ki=0.7, oscillations occur on the steering input and apparently impact the acceleration on y direction. The best value for Ki should be less than 0.7. The other simulation results show that the upper bound of Ki is 0.64 when the preview time is one second.
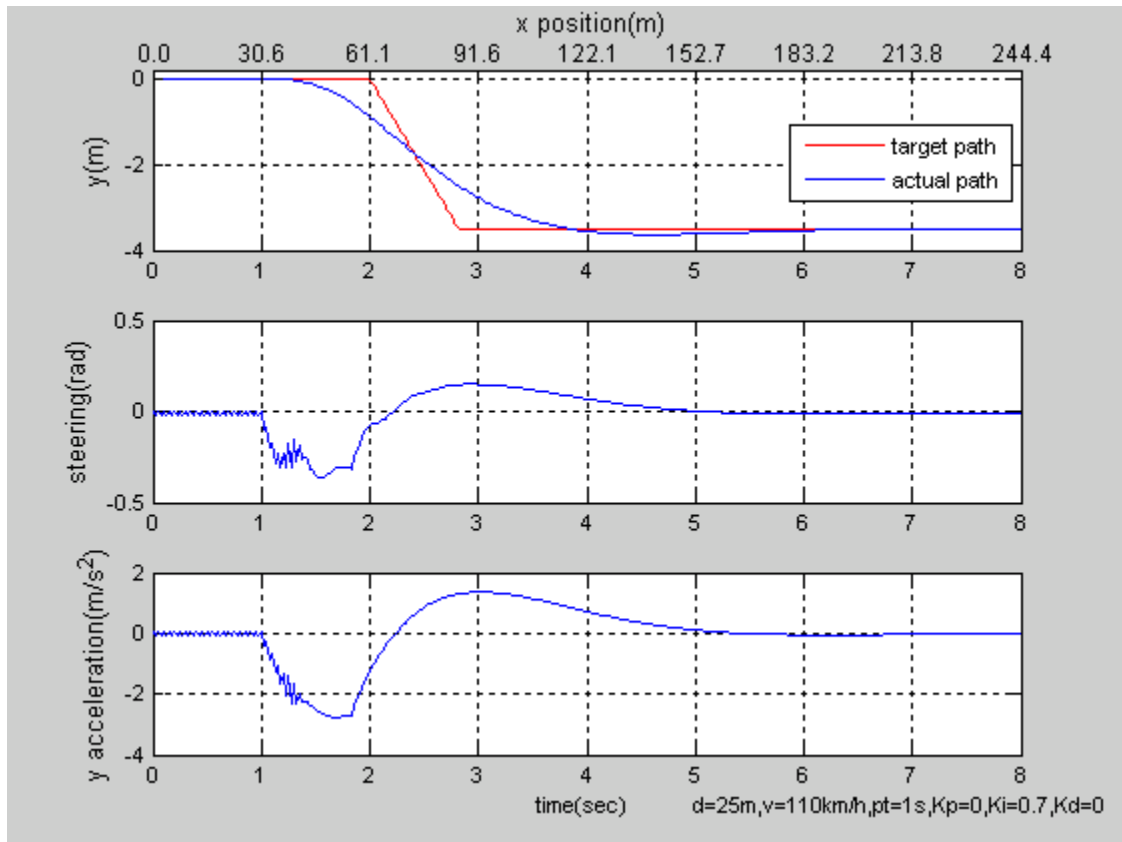
Figure 40 Lane-Change Simulation, Ki=0.7

The same optimization process used on the speed controller was applied to determine the PID parameters for the trajectory tracking controller. GA with only one generation was used to optimize only parameter Ki, and the setting of simulation was the same as in the preview case. A total of 40 Ki values generated by the optimization algorithm were attempted. The results are shown in Figure 41

62
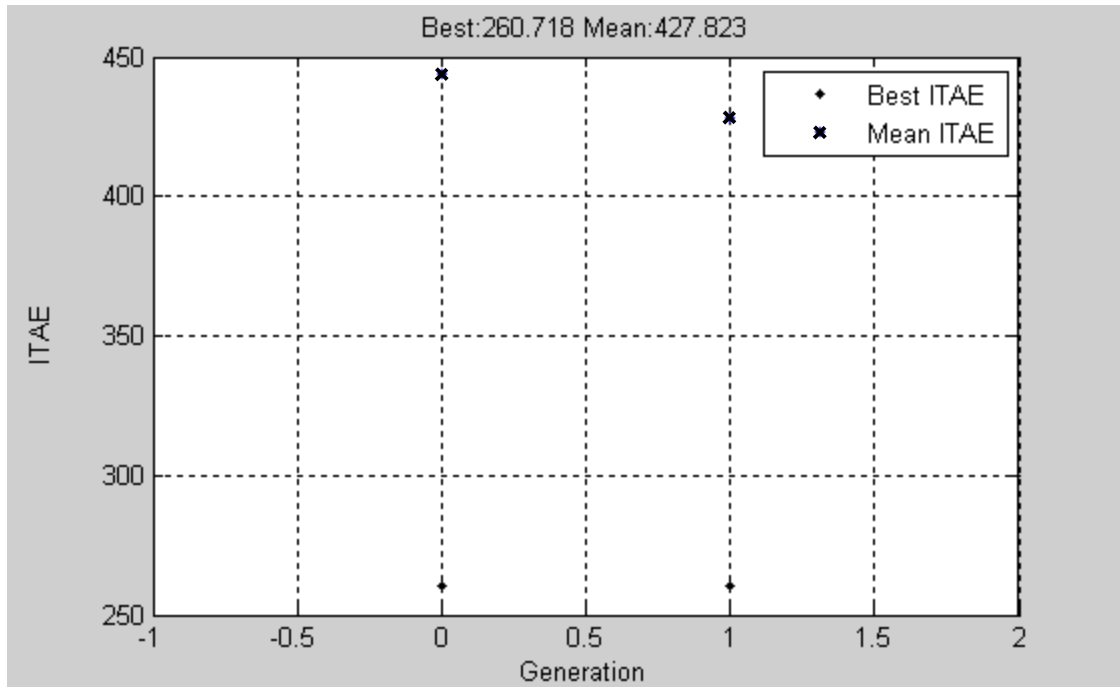
Figure 41 GA Result for Trajectory Tracking Controller

Fitness value in the figure is the ITAE result, the best one was 260.718 with

Ki=0.04. The last 10 optimization results are as follows.

- Ki=0.24 ITAE=329.11

- Ki=0.04 ITAE=260.72

- Ki=0.22 ITAE=327.71

- Ki=0.31 ITAE=331.76

- Ki=0.06 ITAE=294.26

- Ki=0.04 ITAE=260.72

- Ki=0.15 ITAE=320.43

- Ki=0.04 ITAE=260.72

- Ki=0.00 ITAE=2727.12

- Ki=0.06 ITAE=294.26

63

- Optimization terminated: maximum number of generations exceeded.

- Bestresult Ki=0.04, best ITAE=260.72

The optimization process produced the best result of the 40 Ki at 0.04, which has

good performance at 110km/h. Its performance at low speed is shown in Figure 42.
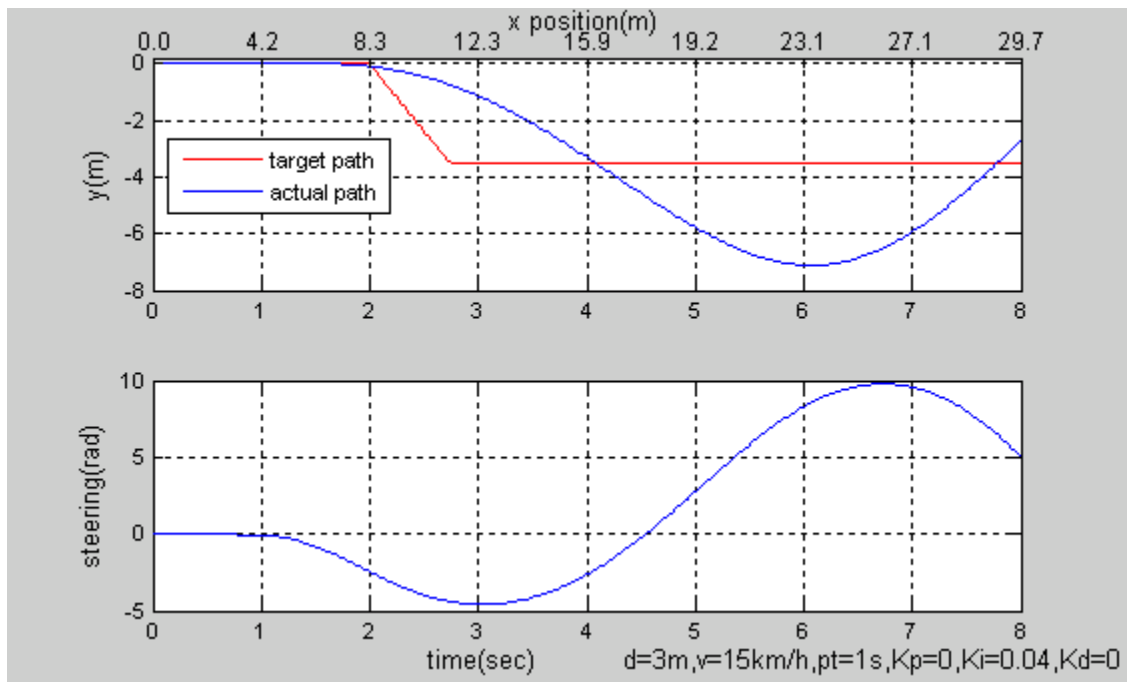
Figure 42 Lane-Change Simulation, Ki=0.04

The tracking controller with a small Ki value failed to make the lane-change

movement at low speed. Compare to Ki=0.04, the upper limit value, 0.64 enabled the PID

controller to perform much better lane-change movements at the same condition, as

shown in Figure 43. The other optimization results with different preview times indicated

that upper limit values at high speed assure that the trajectory tracking controller has the

capability of controlling the vehicle to track reasonable given lane-change paths at a wide

range of given speeds with one second preview time. Therefore, only the upper limit

values were used in the lane-change simulation model.

64
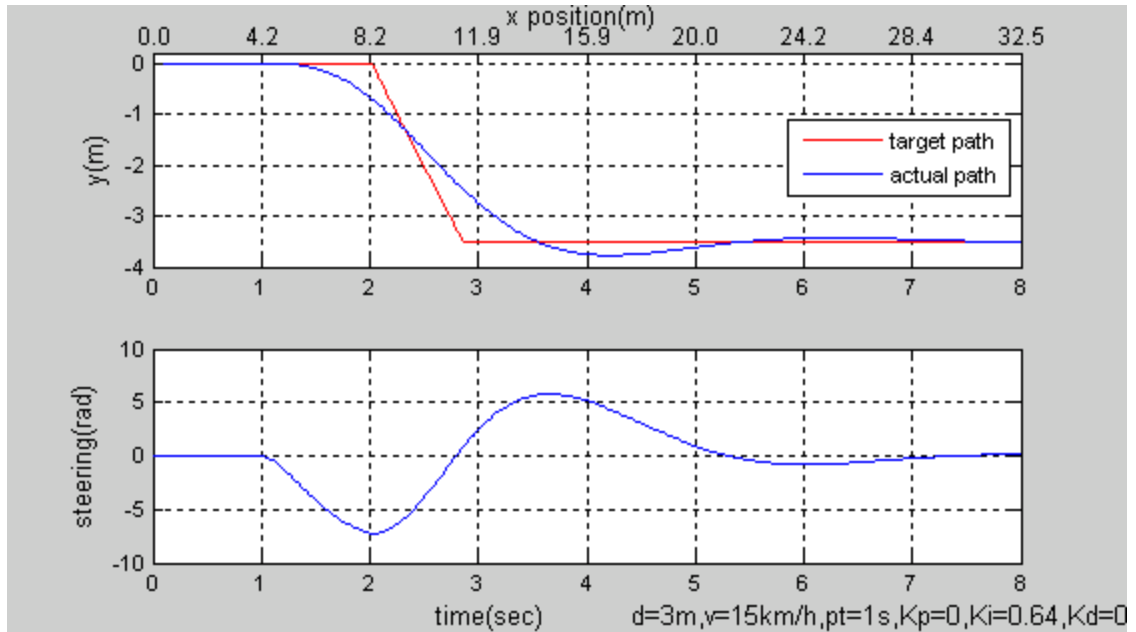
Figure 43 Lane-Change Simulation, Ki=0.64

Finally, 36 preview times, between 0.5 second and 4 seconds with an equal

interval 0.1 second, were applied to the PID controller, and the corresponding Ki values

were determined, as shown in Figure 44.
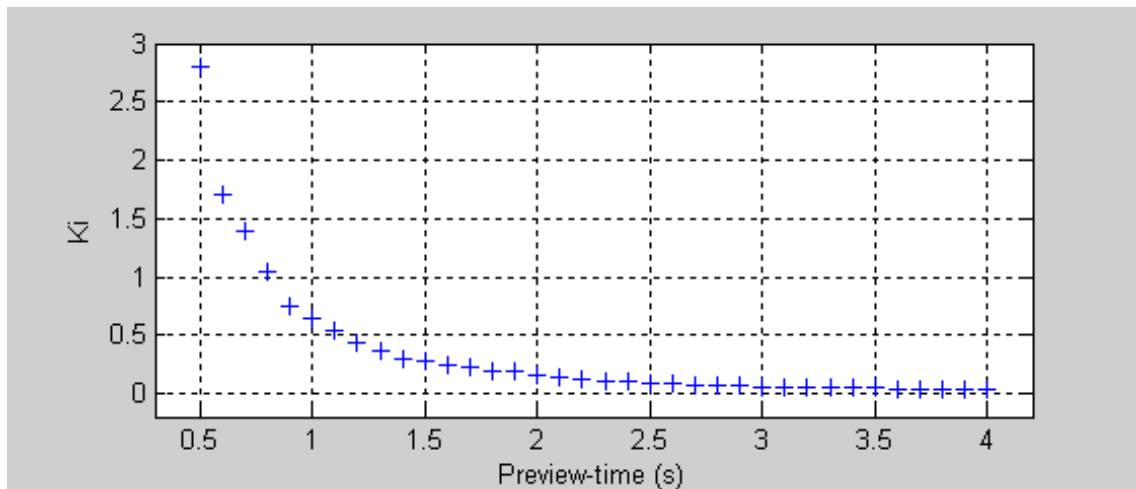
Figure 44 Ki vs. Preview-Time

The parameter tuning results indicate that the upper bound of the Ki parameter

attenuates as the power function against the increase of the preview time. The Ki value

65

reflects the sensitivity of the trajectory tracking controller on position error in y direction. The controller with a large Ki value is more sensitive than the one with a smaller Ki value. Therefore, the Ki can be considered as an indicator that describes the quickness of driving controller.

5.3.3 Lane-Change Simulation Model and Results

In summary, the lane-change simulation model is a multi-input and multi-output model including two PID controllers and a full-car model (Figure 45). In the tracking controller, Ki is determined by a given preview-time using the relationship described in Figure 44.
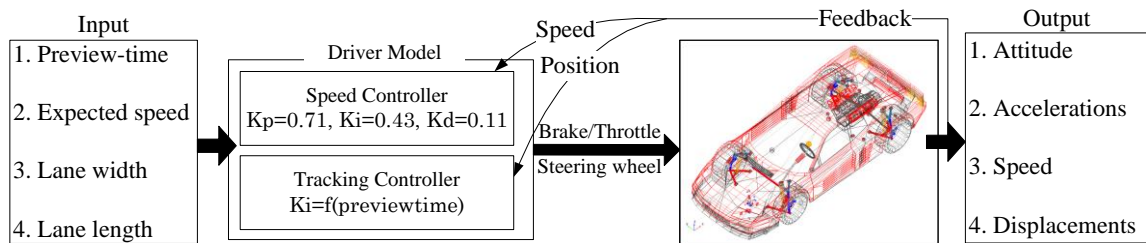
Figure 45 Lane-Change Simulation Model

Two lane-change simulations with different preview times are shown in Figures 46 and 47. For the same given lane-change path, the controller with a 0.5s preview time takes only 2 seconds to finish the movement, while the other takes about 8 seconds; the different preview times lead to totally different driving tracks. As shown in Figure 46, the change of the steering wheel input between t=1s - 1.9s is much quicker than the other time periods, because the target path switch to the right lane is between t=4s - 4.8s and is perceived by the driver with a 3-second preview time. After t=1.9s, the error input to the controller decreases as the target path is fixed and the vehicle gets closer the right lane.

66
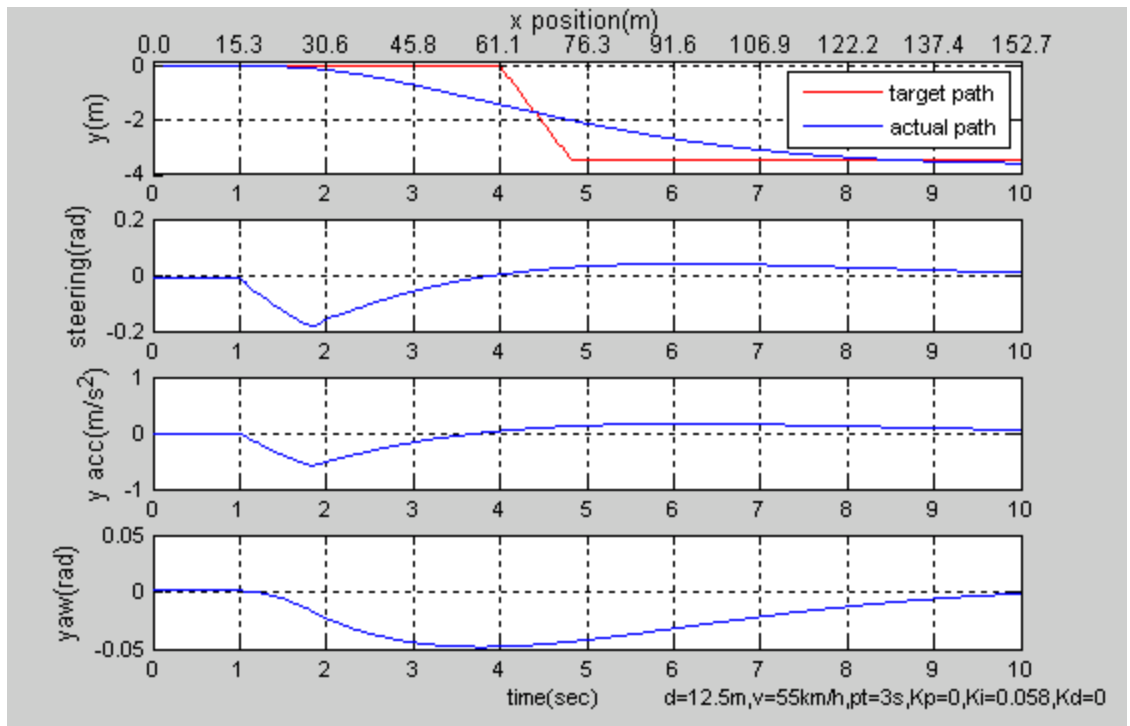
Figure 46 Lane-Change Simulation, Preview-Time=3s
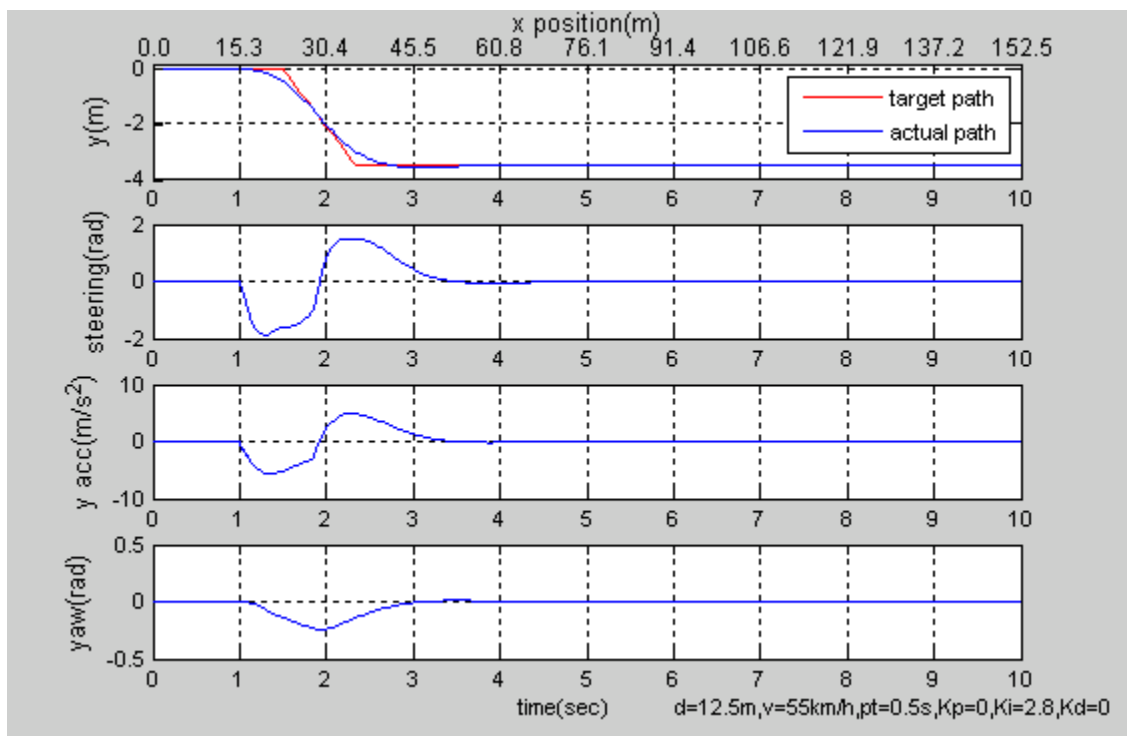
Figure 47 Lane-Change Simulation, Preview-Time=0.5s

67

Chapter 6 Lane-Change Recognition

This chapter presents the process of using SVM to develop a lane-change recognition model. Different features of both simulation data and field data were used to build the recognition model. Comparisons and analysis were performed on the recognition results for a better understanding of improvement methods.

6.1 Lane-Change Data Preparation

The first step for modeling the lane-change recognition system was selecting appropriate data features as the input elements of the recognition model; only those features that showed strong correlation to the lane-change maneuver were considered. Because the simulation data and the field data are all time series sampled in different rates, the data was processed and reorganized for obtaining new data sets with the same sample rates. For both modeling and recognition, it was not necessary to input every sample point and a long period data into the recognition model, which could increase the computation time and have little effect on accuracy improvement. A good input series should be able to represent lane-change behavior characteristics within a small timing window and a large time interval.

6.1.1 Field Data Noise Elimination

Unlike simulation data generated in an ideal situation, field data were collected in a real traffic environment by measurement units with finite precision. Without handling, noises can significantly influence computation accuracy. As shown in Figure 48, many noises are superimposed on the lateral acceleration and yaw angular rates, and the

68

amplitude of the noises is larger than 10 percent of the peak-to-peak value of both two

curves. To eliminate the noises, a moving average filter was used to smooth the data.
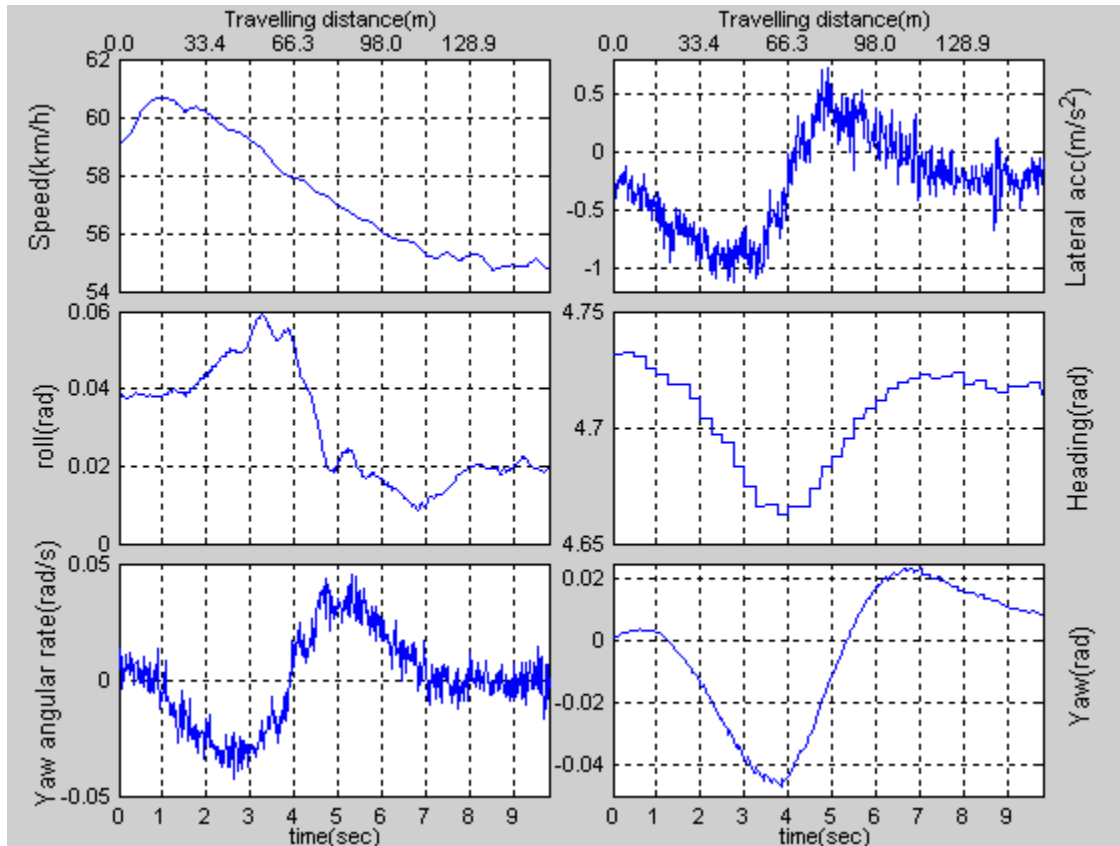
Figure 48 Lane-Change Field Data

The filter is defined in Equation 6.

$$n(k) = \frac{1}{s} \sum_{j=k-(s-1)/2}^{k+(s-1)/2} y(j) \tag{6}$$

where,

n = smoothed data series

y = raw data series

s = data span, an odd constant value

A large span value will drag the smoothed data series to the average of the whole

original data set; a small one will not be able to eliminate the noise. Tests showed that the

69

smoothed series kept the features of the raw data with small noise to a 25 span value. Figure 49 shows a smoothed data generated by the data shown in Figure 48.
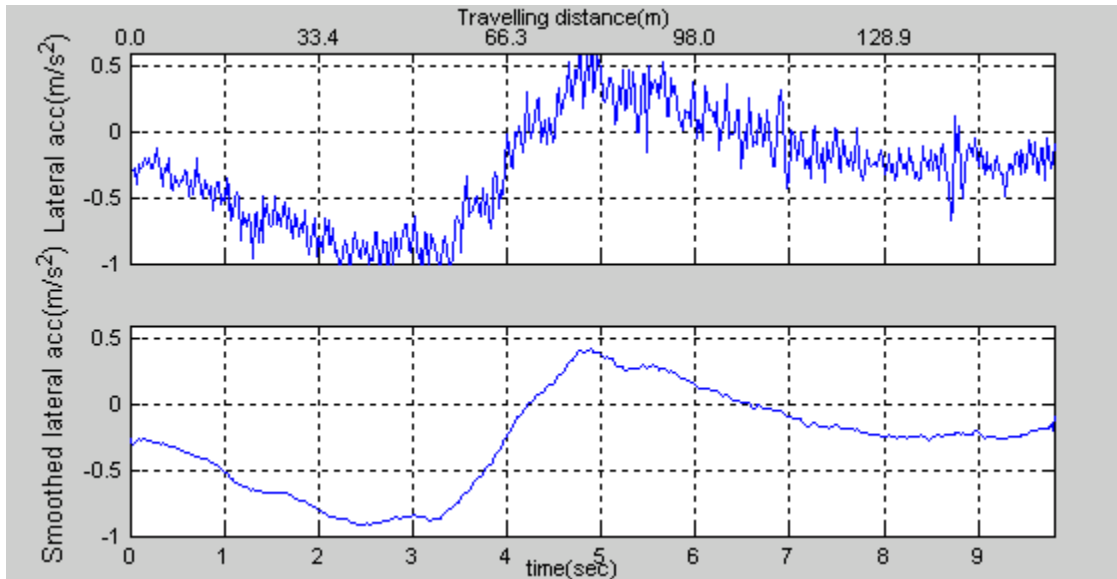
Figure 49 Smoothed Data

Comparing the curves of the raw data and the smoothed one, the peak-to-peak magnitudes are almost the same if the contribution of the noise in the raw data is ignored, and both curves have the same zero crossing points. The smoothed data could substitute for the raw data as the element input to the recognition model.

In Figure 49, the lateral acceleration values before and after a lane change movement are not zero. The whole curve is pulled down by a -0.25 m/s^2 bias value, which is caused by IVDR installation deviation or vehicle body sway. These bias values do not occur in simulation data because lane change movements are simulated on horizontal ground and the measurement sensors are installed perfectly. For increasing recognition accuracy, the bias value has to be removed. In a short period, the bias value can be assumed to be equal to the average value of the lateral acceleration. The values

70

input into the recognition model are the results of smoothed values minus their average value.

6.1.2 Dual Lane-Change Movement

To better understand the performance of the SVM model on the lane-change model and determining the most effective input data features and formats, dual lane-change movement were simulated by the lane-change simulation model with the same controller parameters. In this dissertation, dual lane-change is defined as a driving behavior that drives the vehicle directly to the second right or left lane, show as Figure 50. The width of each lane is 3.5m.
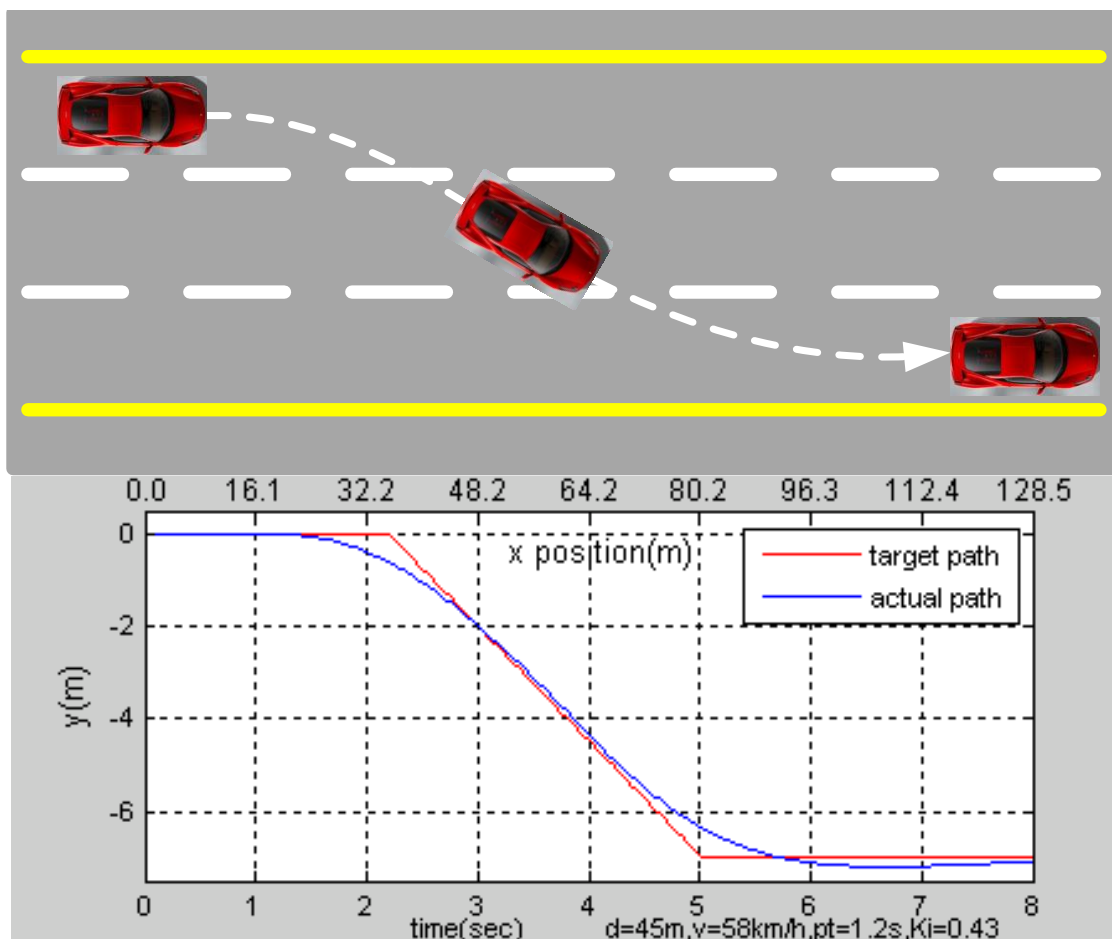
Figure 50 Dual Lane-Change

The trajectory tracking controller had good performance on dual lane-change control, as good as on single lane-change control. The actual path approached the target path with only a small overshot which still kept the vehicle driving in the lane.

6.1.3 SVM Input Data Reduction

A single data row in the training and validation data sets is known as a feature vector and includes two parts: instance data and label. Instance data contain one or more driving data features in a uniform form, and label indicates the category of each instance. The length, representation format and driving data features of the instance data have to be determined.

Like a human observing lane-change behavior, the SVM also needs a time window that includes the main information of the lane-change to classify the driving behavior. The input instance data length depends on the time window size and the time interval. One basic issue with using SVM for lane-change recognition is that the lane-change movements are not accomplished in a fixed time length. A longer lane-change shows a smooth and small change in the features values, such as steering input, heading angle, lateral acceleration, etc. It is difficult to detect the driving patterns of the instance data with small changes and collect them within the limit precision measurement units. This dissertation focuses on lane-changes completed in 10 seconds; most simulation data and field data of lane-changes are shorter than this, between 4 and 7 seconds. The field data were recorded at a 50Hz sampling rate, and the simulation at 40Hz, so the time interval of the input instance data was set to 0.1 second. Various window sizes and time interval were analyzed and compared to obtain the best one for this recognition task.

72

There are three data features that change as a sinusoidal function in lane-change movements-lateral acceleration, heading and yaw angular rate,. These features can be used to build the model, but they are related to each other and have similar patterns, so only lateral acceleration whose value is easily acquired and stable was used in this research. In addition to the previous features, speed is a feature that significantly affects a driver performing a lane-change movement. Thus, speed was considered as one of the input features included in the instance data. In the simulation data set, the maximum values of the lateral acceleration in some lane-change instances were larger than 4 m/s^2 and went far beyond the maximum value of the field data. These instances were removed from the simulation data set. All the lateral acceleration values of both the simulation data set and the field data set were scaled to the range [-1, 1] by multiplying a scaling factor of 0.25. The magnitude of speed value in km/h was as high as 120. To prevent the large value from reducing the effect of the other features and dominating the recognition model; the speed values were also scaled to the range [0, 1] by multiplying a scaling factor of 0.008.The data instance combinations and formats are defined as follows.

$$[Acc(t_1), Acc(t_2), \dots Acc(t_N)]$$

$$[Acc(t_1), Acc(t_2), \dots Acc(t_N), Spd(t_1), Spd(t_2), \dots Spd(t_N)]$$

where,

Acc = scaled lateral acceleration value

Spd = scaled speed value

The input data of the recognition model not only included single lane-change data and dual lane-change data, but also non-lane-change data. Non-lane-change data were chosen from the field data, which contain several irregular patterns on the input features,

73

as shown in Figure 51. The total number of the classes is three, and different labels are

assigned to each class; 0 for non-lane-change, 1 for single lane-change and 2 for dual

lane-change. There are 4,030 instances in the single lane-change simulation data set,

2,921 in the dual lane-change simulation data set, 41 in the single lane-change field data,

and 92 in the non-lane-change field data. The field data sizes were doubled by adding the

reverse value of all the field data instances.
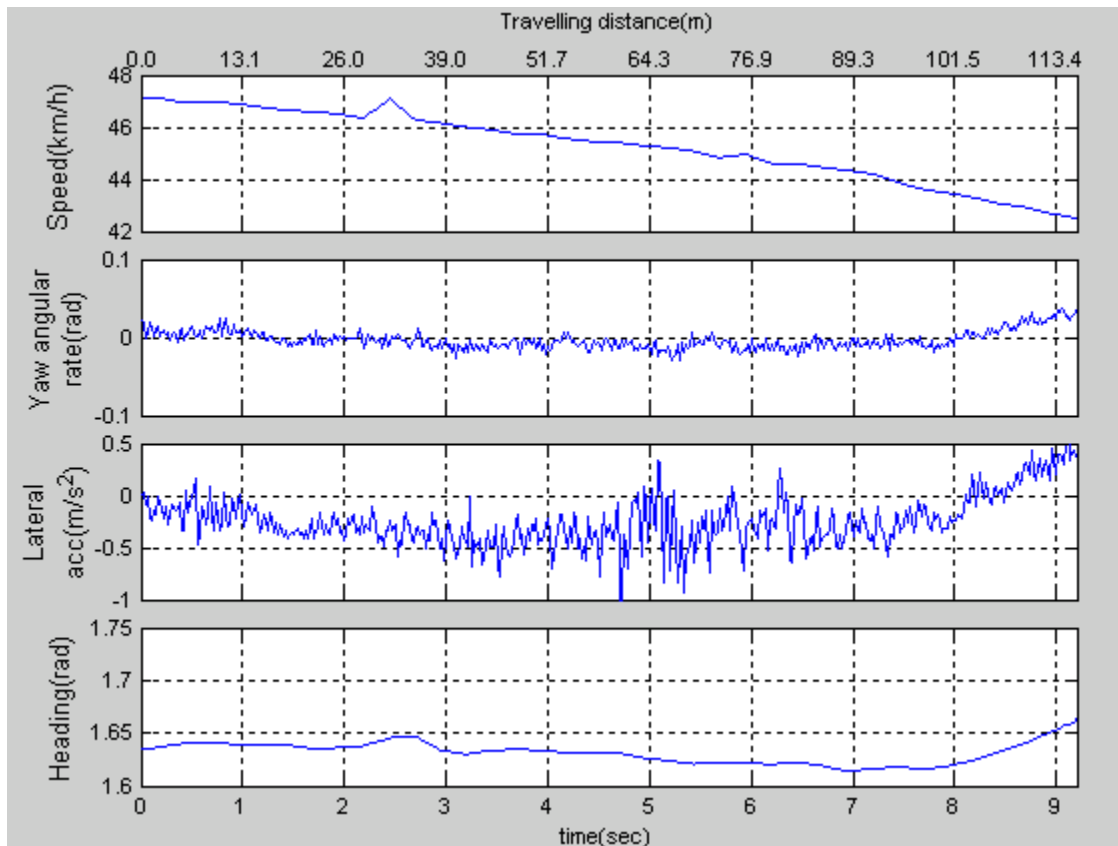
Figure 51 Non-Lane-Change Data

6.2 SVM Training and Results

6.2.1 SVM Model Training

Lane-change recognition model is shown in figure 52. Its input is an instance

array, and its output is one of the three category labels.

74

$$[Acc(t_1), Acc(t_2), \ldots Acc(t_N), Spd(t_1), Spd(t_2), \ldots Spd(t_N)]$$

or

$$[Acc(t_1), Acc(t_2), \ldots Acc(t_N)]$$

Input

SVM Lane-Change Recognition Model

Kernel functions:
Linear
Polynomial
Radial basis
Sigmoid

Output

0 (Non-Lane-Change)
or
1 (Single Lane-Change)
or
2 (Dual Lane-Change)
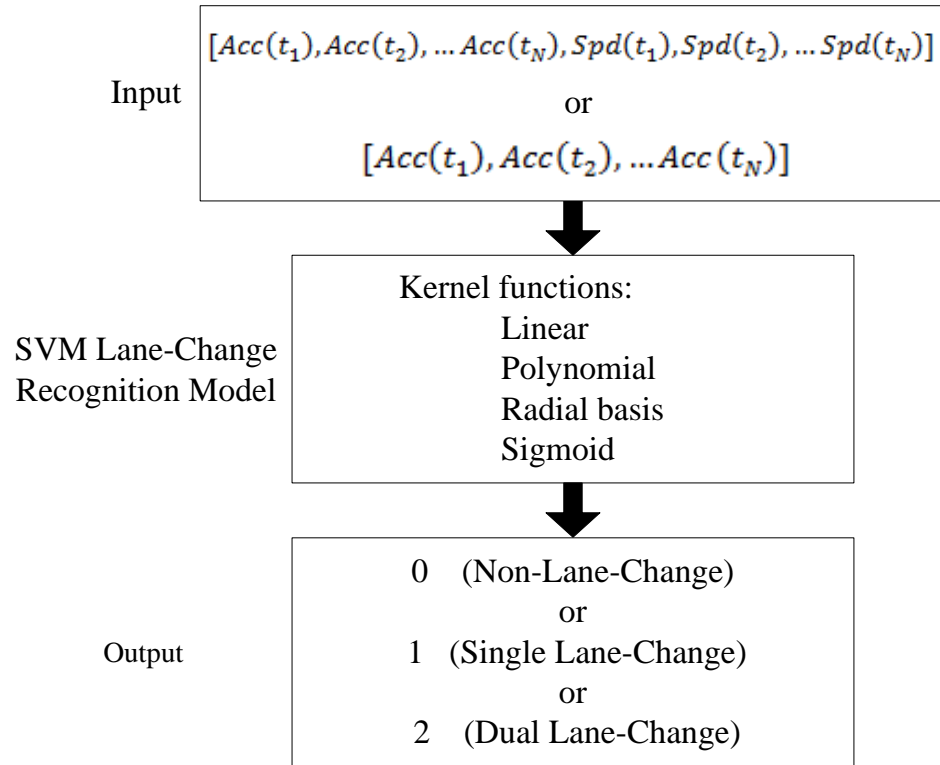
Figure 52 Lane-Change Recognition Model

A kernel function was chosen before training the SVM model. For classification, LIBSVM provides four types of kernel-linear, polynomial, radial basis function (RBF), and sigmoid. Unfortunately, the SVM theory does not provide a direct approach for selecting good kernels. To choose a correct kernel for this lane-change recognition task, the available data were tested against all the four types of kernel functions in LIBSVM to determine the performance of each of them experimentally.

Different kernels have different combinations of parameters. In LIBSVM, these parameters can be easily calculated by the training function with any input data, but without optimization. GA, a good method for solving both constrained and unconstrained optimization problems, is suitable to determine the kernel parameters. The SVM training flowchart is shown in Figure 53.

75
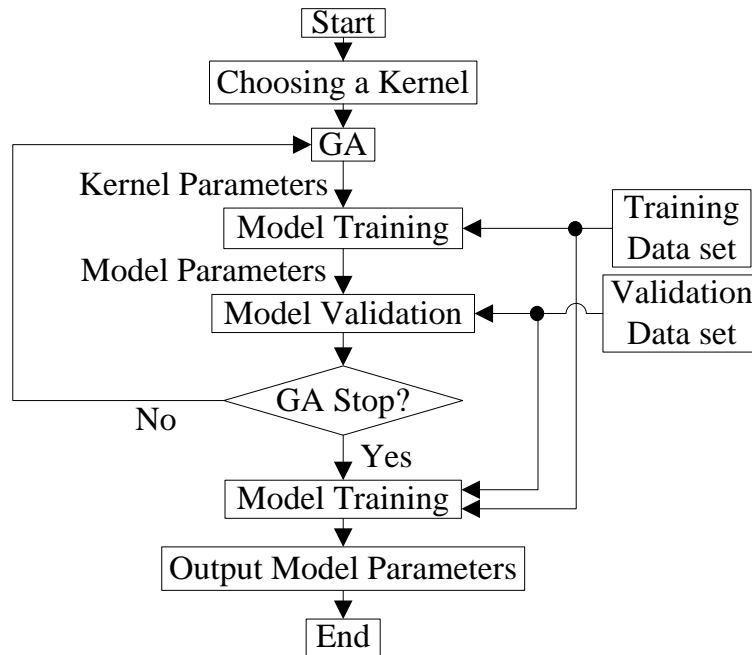
Figure 53 SVM Training Flowchart

As shown in the flowchart, GA generates the kernel parameters for SVM model training. Only the training data set was needed in this step to attain a set of model parameters. The model parameters and the validation data set were the inputs of the SVM model validation. The error rate was calculated based on a one-to-one comparison between the outputs of SVM model validation and the known results given in the validation data set. The model parameters were the output for the lane-change recognition model if GA stops the computation; otherwise, a new optimization cycle was started. In this process, GA attempted to minimize the average error rate of all data groups: single lane-change simulation data, dual lane-change simulation data, single lane-change field data, and non-lane-change field data. This data grouping and error minimization strategy prevented the large instance size groups from dominating the optimization process. Previous observations indicate that the value ranges for each parameter could be set as Table 9.

76

Table 9 SVM Parameter Ranges

Parameters Kernel type	c default 1 *range*	gamma default 0.05 *range*	coef0 default 0 *range*	degree default 3 *range*
Linear	(0, 5)			
Polynomial	(0, 5)	(0, 5)	(-5, 5)	(2, 4)
Radial basis function	(0, 5)	(0, 5)		
Sigmoid	(0, 5)	(0, 5)	(-5, 5)	

Figure 54 shows GA optimization results of training a three-class SVM

classification model.
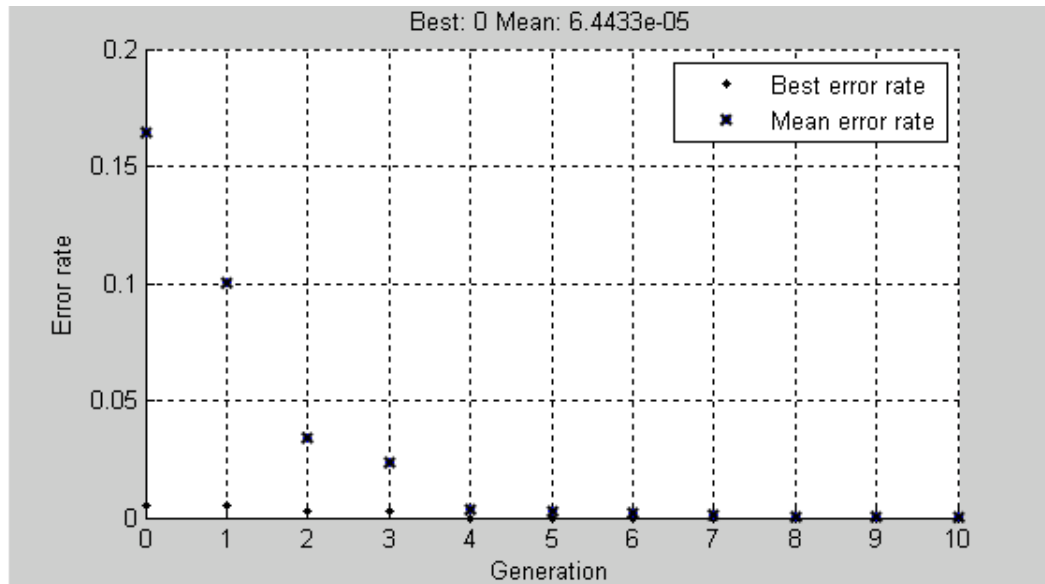
Figure 54 SVM Training with GA

In this training, there were more than 7,000 instances in the training data set, and

the validation data set was the same as the training data set. It was a C-SVC type SVM

training using a RBF kernel, and two parameters were needed to be determined. The

optimization process had 10 generations with the best error rate as low as 0, which means

that a well trained model could recognize all the training instances. The whole

computation process took about 7 minutes, included 200 GA optimization iterations.

77

6.2.2 Lane-Change Recognition Results

This research aims to build a lane-change recognition model primarily using simulation data and expects that model will have the capability of recognizing field data.

To find appropriate kernels, all available kernel types were used in the model training to find out if any of them was suitable for the recognition task. The capability of recognizing training data set was the direct basis for judging the quality of kernel. All the available data were put into the training data set, and the trained models were used to classify all of the training data to check if it had high accurate outputs, which indicates that the kernel type were good in this program. It is unnecessary to try to improve the models that trained by the whole data set but still show bad results.

Kernel performances are compared in Table 10, which indicates that the trained models with polynomial and RBF kernels recognize almost all the instances in the training data set. The two models with a linear kernel completely failed to classify the

Table 10 Kernel Test

Input Kernel		Input Feature: Lateral Acceleration Time Window Length: 6s Number of Sample Points: 15			
		Recognized Percentage/Number of Instances			
		S1 (4030) [a]	S2 (2921) [b]	F1 (82) [c]	F0 (194) [d]
Linear	Training	100%		100%	0%
	Training	100%	0%	100%	0%
Polynomial	Training	100%		100%	95.87%
	Training	100%	100%	100%	95.87%
RBF	Training	100%		100%	94.33%
	Training	100%	100%	97.56%	91.24%
Sigmoid	Training	100%		100%	9.27835%
	Training	84.3921%	89.0448%	97.561%	8.24742%

[a]: 4,030 instances of single lane-change simulation data
[b]: 2,921instances of dual lane-change simulation data
[c]: 82 instances of single lane-change field data
[d]: 194 instances of non-lane-change field data

78

non-lane-change and the dual lane-change data. The models with a sigmoid kernel well classified the lane-change data but could not recognized non-lane-change movements.

To better understand the dominant inputs and how to improve the recognition model performance, training and validation data sets with different number of instances were provided to build the recognition model, and all the trained models were used to recognize all the available instances. The following tables show selected recognition results of lane-change recognition models using RBF and polynomial kernel functions trained by various data combinations.

Tables 11 and 12 show performances of two-class classification and three-class classification models trained by lateral acceleration data, respectively. The time window lengths of the input data are 6 seconds, the total number of sample points in the fixed-length time window is 15. Without including any single lane-change field instance in the training or validation data sets, up to 85 percent of the single lane-change field data could be correctly recognized by the SVM models.

Table 11 Lateral Acceleration Two-Class

Input		Input Feature: Lateral Acceleration Time Window Length: 6s Number of Sample Points: 15			
		Recognized Percentage/Number of Instances			
Kernel		S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
RBF	Training	396/404			185/194
		98.0198%			95.3608%
	Test	99.0074%		70.7317%	
	Training	997/1008			181/194
		98.9087%			93.299%
	Test	99.0323%		81.7073%	
	Training	4030/4030			178/194
		100%			91.7526%
	Test			85.3659%	

79

Table 11 (Continued)

Kernel	Input	S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
RBF	Training	4030/4030		6/6	178/194
		100%		100%	91.7526%
	Test			90.2439%	
	Training	396/404		6/6	185/194
		98.0198%		100%	95.3608%
	Test	98.4615%		80.4878%	
Polynomial	Training	396/404			188/194
		98.0198%			96.9072%
	Test	98.4367%		71.9512%	
	Training	1008/1008			186/194
		100%			95.8763%
	Test	100%		73.1707%	
	Training	4030/4030			183/194
		100%			94.3299%
	Test			81.7073%	
	Training	4030/4030		6/6	184/194
		100%		100%	94.8454%
	Test			89.0244%	
	Training	396/404		5/6	187/194
		98.0198%		83.3333%	96.3918%
	Test			79.2683%	

Table 12 Lateral Acceleration Three-Class

Kernel	Input	S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
RBF	Training	399/404	293/293		188/194
		98.7624%	100%		96.9072%
	Test	98.536%	100%	70.7317%	96.9072%

80

Table 12 (Continued)

Input / Kernel		Input Feature: Lateral Acceleration Time Window Length: 6s Number of Sample Points: 15			
		Recognized Percentage/Number of Instances			
		S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
RBF	Training	996/1008	731/731		183/194
		98.8095%	100%		94.3299%
	Test	98.9082%	100%	81.7073%	94.3299%
	Training	4030/4030	2921/2921		176/194
		100%	100%		90.7216%
	Test			84.1463%	
	Training	4030/4030	2921/2921	6/6	183/194
		100%	100%	100%	94.3299%
	Test			90.2439%	
	Training	396/404	293/293	6/6	184/194
		98.0198%	100%	100%	94.8454%
	Test	98.3127%	100%	80.4878%	
Polynomial	Training	395/404	293/293		188/194
		96.9072%	100%		96.9072%
	Test	97.9404%	99.1099%	70.7317%	
	Training	997/1008	730/731		187/194
		98.9087%	99.8632%		96.3918%
	Test	99.0074%	99.1099%	73.1707%	96.3918%
	Training	4030/4030	2919/2921		180/194
		100%	99.9315%		92.7835%
	Test			79.2683%	
	Training	4030/403	2917/2921	6/6	184/194
		100%	99.8631%	100%	94.8454%
	Test			89.0244%	
	Training	398/404	293/293	6/6	187/194
		98.5149%	100%	100%	96.3918%
	Test	98.933%	99.3495%	80.4878%	

The performance of lane-change models that were trained by lateral acceleration data and speed data is shown in Table 13 and Table 14. With one more feature input, the

lane-change models did not perform much better than the ones in the previous two tables, although it took more time to train the model.

Table 13 Lateral Acceleration, Speed Two-Class

Input		Input Feature: Lateral Acceleration, Speed Time Window Length: 6s Number of Sample Points: 15+15			
		Recognized Percentage/Number of Instances			
Kernel		S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
RBF	Training	396/404			185/194
		98.0198%			95.3608%
	Test	98.4367%		73.1707%	
	Training	997/1008			183/194
		98.9087%			94.3299%
	Test	99.0323%		81.7073%	
	Training	4030/4030			178/194
		100%			91.7526%
	Test			84.1463%	
	Training	4030/4030		6/6	177/194
		100%		100%	91.2371%
	Test			90.2439%	
	Training	396/404		6/6	185/194
		98.0198%		100%	95.3608%
	Test	98.4367%		79.2683%	
Polynomial	Training	398/404			188/194
		98.5149%			96.9072%
	Test	98.8586%		73.1707%	
	Training	1008/1008			188/194
		100%			96.9072%
	Test	100%		75.6098%	
	Training	4030/4030			180/194
		100%			92.7835%
	Test			76.8293%	
	Training	4030/4030		6/6	184/194
		100%		100%	94.8454%
	Test			89.0244%	
	Training	396/404		5/6	187/194
		98.0198%		83.3333%	96.3918%
	Test	98.4367%		79.2683%	

82

Table 14 Lateral Acceleration, Speed Three-Class

Input / Kernel		Input Feature: Lateral Acceleration, Speed Time Window Length: 6s Number of Sample Points: 15+15			
		Recognized Percentage/Number of Instances			
		S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
RBF	Training	395/404	289/293		180/194
		97.7723%	98.6348%		92.7835%
	Test	96.9231%	98.973%	81.7073%	
	Training	1005/1008	729/731		180/194
		99.7024%	99.7264%		92.7835%
	Test	96.8734%	99.7946%	84.1463%	
	Training	4025/4030	2921/2921		170/194
		99.8759%	100%		87.6289%
	Test			90.2439%	
	Training	1006/1008	729/731	5/6	175/194
		99.8016%	99.7264%	83.3333%	90.2062%
	Test	96.8238%	99.7946%	87.8049%	
	Training	395/404	289/293	6/6	181/194
		97.7723%	98.6348%	100%	93.299%
	Test	96.6749%	99.1099%	86.5854%	
Polynomial	Training	401/404	288/293		190/194
		99.2574%	98.2935%		97.9381%
	Test	98.4367%	97.9801%	73.1707%	
	Training	1008/1008	728/731		187/194
		100%	99.5896%		96.3918%
	Test	99.7519%	98.9387%	75.6098%	
	Training	4030/4030	2920/2921		192/194
		100%	99.9658%		98.9691%
	Test			75.6098%	
	Training	1008/1008	728/731	6/6	187/194
		100%	99.5896%	100%	96.3918%
	Test	99.9007%	99.3838%	91.4634%	
	Training	398/404	288/293	4/6	189/194
		98.5149%	98.2935%	66.6667%	97.4227%
	Test	98.3871%	97.7405%	78.0488%	

83

Tables 11-14 indicate that the recognition models had high accuracy for recognizing the simulation instances, even if only 10 percent of the simulation instances were used to train the model. However, these models had much lower accuracy for recognizing a real single lane-change. Including a small amount of single lane-change field instances can improve the capability of reorganizing the single lane-change field instances. The SVM has parallel performances in the two-class and three-class classifications. The RBF kernel showed better performance than the polynomial kernel, especially in those SVM models that were trained without the single lane-change field data, although the models with the polynomial kernel had better classification rates than the others. Compared to those models trained by unique feature, the models trained by lateral acceleration data and speed data did not show significantly different recognition accuracy.

Instances with different sample point numbers and time window size were used to train the SVM models, and the results are shown in Table 15. Large time window size or too many sample points did not improve the recognition accuracy, and vice versa. An appropriate time window size and number of sample points can be determined experimentally; the basic principle is that the time window must include the most important pattern information of the lane-change movement and the sample points in the time window can represent the trends of the original pattern. The table indicates that for the three-class model using an RBF kernel, 6 seconds is a good length for the time window, and the number of sample points can be less than 15.

Table 15 Different Sample Point Number and Time Window Size

Kernel	TWL	NSP		S1 (4030)	S2 (2921)	F1 (82)	F0 (194)
			Input Feature: Lateral Acceleration **Time Window Length: TWL** **Number of Sample Points: NSP**				
			Recognized Percentage/Number of Instances				
RBF	8.8s	44	Training	4030/4030	2921/2921		181/194
				100%	100%		93.299%
			Test			73.1707%	
		22	Training	4030/4030	2921/2921		179194
				100%	100%		92.268%
			Test			76.8293%	
		11	Training	4030/4030	2921/2921		183/194
				100%	100%		94.3299%
			Test			78.0488%	
	6s	30	Training	4030/4030	2921/2921		181/194
				100%	100%		93.299%
			Test			82.9268%	
		15	Training	4030/4030	2921/2921		176/194
				100%	100%		90.7216%
			Test			84.1463%	
		7	Training	4018/4030	2921/2921		176/194
				99.7022%	100%		90.7216%
			Test			85.3659%	
	4s	20	Training	4030/4030	2921/2921		167/194
				100%	100%		86.0825%
			Test			54.878%	
		10	Training	4030/4030	2921/2921		160/194
				100%	100%		82.4742%
			Test			50%	

This chapter introduced the data preparation process for model training and presents recognition performance of the SVM models trained by different data sets. These SVM models have high accuracy for recognizing simulation data for both two-class and three-class classification, but lower accuracy for field data. The performance of

85

lane-change recognition models is affected by various aspects, such as training data

feature, length, time widow size, etc.

Chapter 7 Conclusion and Future Work

This study focuses on development of lane-change recognition model using SVM. To achieve data preparation for recognition modeling, an IVDR that integrates a GPS and an IMU was built for field data collection, and a lane-change simulation model was developed to avoid safety issues in field data colleting and generating a full-scale lane-change data set. Both field data and simulation data were used to build a lane-change recognition model. Comparison and analysis were completed for the whole process. Conclusions and results are summarized as follows.

- An IVDR that integrated a GPS and an IMU is capable of capturing lane-change patterns. The IVDR also provides non-lane-change data for training recognition models. Recognition results indicate that most field data can be successfully classified by a well trained lane-change recognition model.

- The lane-change simulation model consists of three essential elements: a full-car model, a speed controller, and a trajectory tracking controller. PID strategies were applied to these two controllers and their parameters were tuned with the assistance of GA. The speed controller can keep steady speeds by adjusting the brake and throttle of the full vehicle model. By imitating the preview behavior of real driving, the trajectory tracking controller is capable of controlling the steering wheel to follow a given lane-change path. The lane-change simulation model performed more than 7,000 lane-change movements at different speeds, on different given paths.

87

- GA is an appropriate for optimizing the model training. A well trained lane-change recognition model can effectively classify up to 85 percent of the real lane-change data not included in the training and validation data sets, better results can be achieved by including a small number of real lane-change instances in the model training process. Both field data and simulation data have great contribution to training lane-change recognition models, whose recognition accuracies can be increased by using more data in training and validation. The RBF kernel and polynomial kernel showed good recognizing performance in both the two-class and three-class classifications.

Future work includes the follows.

- More field data should be collected at different speeds, which will significantly improve lane-change recognition accuracy.

- The trajectory tracking controller could be upgraded by adjusting the feedback variables and their combinations to obtain lane-change patterns that are closer to real ones.

- In SVM modeling, more data features, data representations, and optimization methods should be used and analyzed to find better solutions for improving lane-change recognition capability.

- The method using simulation data to estimate field data can be used to develop other driver behavior recognition models.

References

320 Series User's Manual, Document 7430-0292-01 Revision A, August 2009. Crossbow Technology, Inc.

ANTARIS 4 GPS Module System Integration Manual, GPS.G4-MS4-05007-A1. 2007. u-blox AG.

ANTARIS Positioning Engine NMEA and UBX Protocol Specification, GPS.G3-X-03002-D. 2003. u-blox AG.

Bevly, D.M., Ryu, J., and Gerdes, J.C., 2006. Integrating INS Sensors with GPS Measurements for Continuous Estimation of Vehicle Sideslip, Roll, and Tire Cornering Stiffness. *Intelligent Transportation Systems, IEEE Transactions on Intelligent Transportation Systems*, 7(4).

Boyraz, P., Acar, M., and Kerr, D., 2007. Modeling and Hidden Markov Models for Driving Maneuver Recognition and Driver Signal Fault Diagnosis in an Urban Road Scenario. IEEE Intelligent Vehicles Symposium, Istanbul, Turkey.

Crashes vs. Congestion – What's the Cost to Society? Prepared for AAA. 2011. Cambridge Systematics, Inc.

Chang, C., and Lin, C., 2011. LIBSVM: A Library for Support Vector Machines.ACM Tr ansactions on Intelligent Systems and Technology. Software available at: http://w ww.csie.ntu.edu.tw/~cjlin/libsvm/.

Creating a Java, Eclipse, Tomcat, MySQL Environment. [Online]. Available at: http://w ww.sipages.com/jetv2.shtml.

Genetic Algorithm. [Online]. Available at: hhttp://en.wikipedia.org/wiki/Genetic_algorith m.

Genetic Algorithm. [Online]. Available at: http://www.mathworks.com/help/gads/ga.html.

Google Maps JavaScript API V3. [Online]. Available at: http://code.google.com/apis/ma ps/documentation/javascript/.

Guo, K., 1984. Driver-Vehicle Close-Loop Simulation of Handling by "Preselect Optimal Curvature Method". Automotive Engineering, China.

89

Hsu, C., Chang, C., and Lin, C., 2010. A Practical Guide to Support Vector Classification. [Online] Available at: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

HTML Tutorial. [Online]. Available at: http://www.w3schools.com/html/.

Kasper, D., Weidl, G., Dang T., Breuel, G., Tamke, A., and Rosenstiel, W., 2011.Object-Oriented Bayesian Networks for Detection of Lane Change Maneuvers. Intelligent Vehicles Symposium (IV), IEEE.

Kuge, N., Yumamura, T., Shimoyama, O., and Liu, A., 2000. A Driver Behavior Recognition Method based on a Driver Model Framework. Proceedings of the SAE World Congress, Detroit, MI.

Lee, C., and Park, B., 1994. Lane Change Simulation with Vehicle Driver Model using ADAMS. 1994 International ADAMS User Conference.

Liang, Y., Reyes, M. L., and Lee, J. D., 2007. Real-Time Detection of Driver Cognitive Distraction Using Support Vector Machines. *IEEE Transactions on Intelligent Transportation Systems*, 8(2).

Mandalia, H. M., Salvucci, D. D., 2005. Using Support Vector Machines for Lane-Change Detection. Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting.

Martini, R. D., 2006. GPS/INS Sensing Coordination for Vehicle State Identification and Road Grade Positioning. Master's thesis, Pennsylvania State University.

Measures of Controlled System Performance. [Online]. Available at: http://www.online-courses.vissim.us/Strathclyde/measures_of_controlled_system_pe.htm.

Miyajima, C., Ukai, H., Naito, A., Amata, H., Kitaoka, N., and Takeda, K., 2011. Driver Risk Evaluation Based on Acceleration, Deceleration, and Steering Behavior. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 1829-32.

Passenger Cars – Test Track for a Severe Lane-Change Maneuver. BS ISO 3888-1:1999. British Standard, 1999.

PID Controller. [Online]. Available at:http://en.wikipedia.org/wiki/PID_controller.

Salvucci, D., Mandalia, H., Kuge, N., and Yamamura, T., 2007.Lane-Change Detection Using a Computational Driver Model. *Human Factors*, 49(3), 532-42.

Toledo, T., Musicant, O., and Lotan, T., 2008.In-vehicle Data Recorders for Monitoring and Feedback on Drivers' Behavior. Transportation Research Part C: Emerging Technologies, 16(3), 320-331.

90

VB.NET Socket Programming.[Online]. Available at: http://vb.net-informations.com/co mmunications/vb.net_socket_programming.htm.

Weiss, K., Kaempchen, N., and Kirchner, A., 2004. Multiple-Model Tracking for the Detection of Lane Change Maneuvers. Intelligent Vehicles Symposium (IV), IEEE.

Xuan Y., and Coifman, B., 2006. Lane Change Maneuver Detection from Probe Vehicle DGPS Data. Intelligent Transportation Systems Conference, ITSC, IEEE.

91