

The Crewman's Associate for Path Control (CAPC) An Automated Driving Function

Final Report

U. S. Army Tank and Automotive Command

TACOM Technical Report No. 13673

UMTRI No. 95 - 35

Robert D. Ervin

Gregory Johnson

Paul Venhovens

Charles C. MacAdam

Engineering Research Division,

The University of Michigan Transportation Research Institute

A. Galip Ulsoy

David J. LeBlanc

Chiu-Feng Lin

Huei Peng

Chia-Shang Liu

Department of Mechanical Engineering & Applied Mechanics,

The University of Michigan

Garth Gerber

Robert DeSonia

Environmental Research Institute of Michigan

Thomas E. Pilutti

Ford Motor Company

November 3, 1995

UMTRI The University of Michigan
Transportation Research Institute





Technical Report Documentation Page

1. Report No. UMTRI-95-35		2. Government Accession No.		3. Recipient's Catalog No.	
4. Title and Subtitle The Crewman's Associate or Path Control (CAPC) An Automated Driving Function				5. Report Date October 13, 1995	
				6. Performing Organization Code	
7. Author(s) Ervin, R., Johnson, G., Venhovens, P., MacAdam, C.				8. Performing Organization Report No. UMTRI-95-35	
9. Performing Organization Name and Address The University of Michigan Transportation Research Institute 2901 Baxter Road, Ann Arbor, Michigan 48109-2150				10. Work Unit No. (TRAIS)	
				11. Contract or Grant No. DAAE07 93-C-R124	
12. Sponsoring Agency Name and Address U.S. Army Tank Automotive Command Warren, MI 48397-5000				13. Type of Report and Period Covered	
				14. Sponsoring Agency Code	
15. Supplementary Notes					
16. Abstract An integrated system for preventing road-departure accidents was developed and then implemented on a passenger car. The system images the roadway ahead by means of a digital Charge-Coupled Detector camera, predicts the vehicles's future trajectory based upon inertial sensing and model prediction, computes a "time-to-lane-crossing" or TLC value beyond which the vehicle's mass center will cross the road-edge line, and executes a decision to either warn the driver or initiate a control intervention. Although the control intervention feature, involving differential braking, is not implemented in the system prototype, its design is documented herein and evaluated using computer simulation. A driver-state-assessment feature is also discussed. Overall, the prototype system affords a testbed for examining the many parameters and features that may influence the utility of a road-departure, warning-and-intervention function for motorists.					
17. Key Words road departure, vehicle control, road imaging, safety, warning intervention, intelligent transportation systems				18. Distribution Statement Unrestricted	
19. Security Classif. (of this report)		20. Security Classif. (of this page)		21. No. of Pages 145	22. Price

Table of Contents

1.0	Introduction.....	1
2.0	CAPC Performance Goals & Objectives.....	3
2.1	CAPC Goal.....	3
2.2	Objectives of the System.....	3
2.3	Technological Emphasis.....	4
2.4	Requirements.....	4
2.4.1	Driver Activity.....	4
2.4.2	Application Environment.....	5
2.4.3	Warning and Intervention to Avoid Road Departure.....	6
3.0	Development of the System Design.....	7
3.1	CAPC System Concept.....	7
3.2	Design of the Lane-Mark Sensor (LMS) Subsystem.....	11
3.2.1	Design Requirements.....	11
3.2.2	Lane-Mark Sensor Design.....	11
3.2.3	Location and Tracking of Lane Marks.....	15
3.2.4	Calibration 16	
3.2.5	Software Documentation.....	17
3.3	Design of the Lane Margin Processor.....	18
3.3.1	Time-to-Lane Crossing Computation.....	19
3.3.2	Lane Geometry Modeling.....	21
3.3.3	Vehicle Trajectory Prediction.....	23
3.4	Design of the Driver Status Assessment.....	27
3.5	Design of the Brake-Steer Controller.....	29
3.5.1	Brake-Steer Authority.....	29
3.5.2	Near-Range Kalman Filter Design.....	33
3.5.3	Brake-Steer Controller.....	41
3.6	Design of the Decision Module.....	49
3.7	Design of a System Simulation Tool.....	58
3.7.1	Simulation Architecture.....	58
3.7.2	Simulation User Interface.....	60
3.7.3	Vehicle Model Evaluation.....	62

4.0	Implementation of CAPC prototype	69
4.1	Prototype System Configuration	70
4.2	Physical Installation in Taurus SHO	76
5.0	Testing of the CAPC Prototype	77
5.1	Test Plan and Rationale.....	77
5.2	Test Results	78
5.2.1	Experimental Data	78
5.2.2	Performance Constraints During Testing.....	87
6.0	Conclusions and Recommendations	90
7.0	References	93
8.0	Prototype Operating Instructions & Software Descriptions.....	94
8.1	Safety Assessment Report.....	94
8.2	CAPC Prototype Vehicle System Operating Instructions.....	97
8.3	CAPC Software Systems Manual	99
8.3.1	CAPC Prototype Vehicle Integrated User/Programmer's Manual	99
8.3.2	Software Flowcharts and System Interface Requirements	111
8.3.3	CAPC Prototype Software Code	118
8.3.4	CAPC Simulation Tool Code	127
8.3.4	Lane-Mark-Sensor Software Code.....	139
8.4	Drawings	140
	Appendices	146

1.0 Introduction

This document constitutes the final report on TACOM Contract No. DAAE07-93-C-R124 entitled, "The Crewman's Associate for Path Control (CAPC): An Automated Driving Function." The project has developed an electronic system for preventing road-departure accidents by motor vehicles. Insofar as the system concept involves supplementing driver control, but not replacing it, this project builds upon the Army's interest in limited automation of the driving function for the sake of reducing crew sizes in either wheeled or tracked military vehicles. It also addresses the interest of the commercial motor-vehicle industry in active-safety technologies for the private automobile.

The prime contractor has been the University of Michigan. The University's participation has come from both the UM Transportation Research Institute (UMTRI) and the College of Engineering's Mechanical Engineering and Applied Mechanics (MEAM) Department. The Environmental Research Institute of Michigan (ERIM) has served as a subcontractor. The Ford Motor Company's Research Laboratory has also participated as a collaborating partner.

The project has involved a two-year effort to develop, design, fabricate, and test a road-departure prevention system. The package design emerged with the aid of a complete system simulation. System computational features were incorporated within the simulation together with representations of the vehicle, sensors, roadway, and driver-assist function. When refined, the software necessary for effecting the function of road-departure prevention was downloaded onto a computer installed in the test vehicle. Equipped also with a suite of sensors plus recording instrumentation, this vehicle then became a testbed for a limited study of the road-departure issue using proving grounds and public roadways.

As with any system that intends to supplement the control activity of the human driver, it is highly important that the scope of system functionality be carefully defined, lest it be assumed that the remarkable robustness of the human operator is being somehow matched. In Section 2.0 of this report, the distinctions relative to the scope of the currently addressed system are delineated.

The development of a system configuration is covered throughout Section 3.0, presenting both the development and design considerations that were addressed for each of the principal subsystems of the package. The simulation tool is presented as both a stand-alone engineering aid and as a complete analytical statement of the contents of the CAPC warning-and-intervention system.

The developed system was built up on the platform of a Ford Taurus SHO passenger car. With road-edge sensing by digital Charge-Coupled Detectors (CCD) video camera and serial communication of data among three major modules of the system, the package provided a complete working prototype of the road-departure-prevention function. The

physical implementation of this prototype is presented in Section 4.0 of this report. The results of limited field testing are presented in Section 5.0.

It should be pointed out that the CAPC system was conceived as providing both a warning functionality and a direct intervention by means of differential braking, which effects a limited form of path control. While both functions were represented within the system simulation, only the warning function was implemented on the prototype vehicle within this project.

Conclusions and recommendations are presented in Section 6.0. The basic conclusion is that while the working prototype does indeed provide a reasonable testbed for examining this function, many opportunities for robustness improvement exist.

Finally, Section 8.0 includes an assessment of safety issues, a set of instructions for system operation, and documentation of both the hardware and software aspects of system design. A set of appendices provides supporting material.

2.0 CAPC Performance Goals & Objectives

This section presents statements of the system goal, objectives, and requirements that were used to guide the initial development of the CAPC prototype. Upon having implemented a working package in this project, it appears that substantial variation in the rules for warning-and-intervention functions is both possible and worth further exploration. Nevertheless, the reader may look upon the "Requirements" section as a set of assumptions underlying the CAPC prototype as it was implemented within this project.

2.1 CAPC Goal

The goal statement provides the highest-level description of the purpose of this system. The statement embodies the value judgment that it is safer to keep vehicles on the roadway than it is to allow inadvertent departure from the road into the potentially threatening roadside zone. The overarching goal of the CAPC system is stated as follows:

To limit the occurrence of events in which motor vehicles inadvertently depart from the travel lanes of limited-access highways and intrude into the adjacent roadside, perhaps risking rollover, collision with fixed objects, or uncontrolled reentry into traffic.

2.2 Objectives of the System

The objectives characterize the desired performance attributes afforded by design features of the system. Thus, the objectives serve to break down the goal in terms that point toward the design approach:

- 1) to monitor the location and orientation of painted road edges and lane delineators, so as to determine the layout of the road ahead of the host vehicle
- 2) to predict the path of the vehicle in near-future time
- 3) to determine the future time at which the predicted path of the vehicle departs from the travel lanes of the roadway ahead
- 4) to monitor the driver's road-keeping behavior to more reliably discern whether a certain variation in road-following performance is truly threatening or is benign for this driver
- 5) to provide an audible warning to alert the driver to a impending road-edge departure that is deemed truly threatening

6) to provide a control intervention that safely redirects the vehicle's path such that the driver can readily regain manual control

7) to manage the monitoring and decisionmaking functions such that false alarms are minimized and total "misses" of protective action are practicably nonexistent

8) to retain a level of control authority for the driver that exceeds that of the provided system at all times

2.3 Technological Emphasis

The CAPC system prototype is to provide a testbed platform upon which to gain empirical evidence of favorable approaches for predicting road departure, deciding on a protective response, delivering that response, and studying human interaction during the recovery phase of near-departure events. Accordingly, the sensing task is defined to have sufficient simplicity that a minimum investment in sensing technology is required. A rather ideal set of roadway conditions has been stipulated, below, in order to ensure that a relatively simple sensor can deliver reliable road-edge detection, permitting the concentration of resources onto the study of the driver-assistance functionality.

2.4 Requirements

The requirements constitute non-quantitative statements of system features upon which design specifications would be based. The requirements have been categorized in three groups—those dealing with the driver's activity, the highway environment, and the warning-and-intervention process.

2.4.1 Driver Activity

The prototype system interacts with driver activity in the following way:

- The type of driver control failure for which the CAPC system serves as a countermeasure involves inadvertent departure from the roadway due to inattention or drowsiness. Such departures are assumed to involve rather small angles of departure—being of the order of 1 degree of arc included between the resultant velocity vector of the vehicle and the local tangent to the roadway centerline. Departures in this class are thought to be amenable to correction by means of either an audible warning or a modest-authority control intervention.
- The system does not interrupt the normal road-keeping task. Although this provision suggests that lane-changing activity, for example, would not be interrupted in any way, the stipulation of highway conditions, below, further constrains vehicle travel to the

right-most lane only. Thus, while the concept of CAPC is such that free change of lanes would be supported in a fully mature system, the prototype vehicle does not incorporate the high-level logic needed to support lane-changing.

- The prototype system accommodates a wide range in the road-keeping behavior of drivers. As an example, "accurate road-keeping" behavior (such as a driver who steers quite precisely down the center of the lane) enables a more certain conclusion of pending road departures such that warnings come earlier in the departure sequence and a greater likelihood exists of departure avoidance. On the other hand, "sloppy road-keeping" behavior (i.e., weaving around in the lane) requires that a pending roadway departure be defined more narrowly in order to reduce the burden of false alarms. Warning-and-interventions both come later in the departure sequence and the likelihood of departure avoidance is somewhat reduced.
- Recognizing that driver control is to be preserved as the highest authority, the system will not intervene when the brake pedal has been applied and any intervention-in-progress will be discontinued whenever manual braking begins. In order to permit steering-only recovery by the driver, the system will "soften" the strength of its intervention controller (i.e., effectively reduce its gain) as it approaches the intended conclusion of the intervention sequence—such that any conflict between system-applied path corrections and driver-applied control via steering inputs will diminish toward zero as the system's authority is withdrawn. (While this initial statement attempts to address the issues of driver interaction with machine-delivered intervention control, it is fully recognized that a vast ignorance exists on this point, calling for extensive research with an operating prototype.)
- The driver will hear an audio warning just before and during any control intervention. The warning will continue until braking is applied or until the warning sequence times out.
- Even if path-control intervention has been initiated (by means of differential braking) a driver will be able to override the path correction by means of the considerably higher authority permitted via steering wheel actuation.

2.4.2 Application Environment

The prototype system applies to the following operating environment:

- limited-access highways, away from entrance and exit ramps at which road-edge striping is intentionally interrupted
- white striping that is in virtually as-new condition and which is painted as a) a continuous line on the right outside edge of the right-most lane and b) as a dashed delineator on the left side of this lane

- daytime, non shadowed, illumination of the pavement
- pavement free of water, snow, or other contaminants
- pavement in relatively good repair
- shoulder of road clear of harmful objects and debris for at least the distance of a car width
- other vehicles no closer than 50 meters in front of the CAPC prototype (implying a time headway of some 2 seconds or more, at highway speeds)

2.4.3 Warning and Intervention to Avoid Road Departure

The **Warning** function activates when the current value of Time-to-Lane Crossing (TLC) drops below a threshold value. The threshold value and the point on the vehicle to which the TLC determination is "referenced" are to be adjustable in the CAPC prototype.

A **Control Intervention** is initiated at the latest point in time within which the differential brake-steer system can act automatically and succeed without driver participation in preventing a pending Road Departure, given the control authority of the system. (Although within this phase of the project, this feature was not fully implemented a warbled form of audible warning was sounded when the "latest point" criterion was satisfied, as a simple indicator of the intervention event.)

3.0 Development of the System Design

3.1 CAPC System Concept

The CAPC system is designed to detect the impending departure of a motor vehicle from a roadway, and to first warn the driver of the danger. Then, if necessary, the system must actively intervene with the goal of keeping the vehicle on the road. The system is designed as a safety backup for situations in which a driver is not responding properly due to drowsiness, illness, or other lapses. Intervention is done by differential braking, which provides a limited path-control capability. The driver remains in the loop and is always able to steer the vehicle even if differential braking is activated.

To anticipate road departures and provide assistance at appropriate moments, the system includes the following functions and elements:

- a sensing system that records and processes the roadway geometry in front of the vehicle
- a prediction of the future trajectory of the vehicle
- an ongoing evaluation of the driver's lane-keeping performance
- logic for decision-making that considers information about the driver, the roadway environment and the vehicle
- a means of intervention

Figure 3.1.1 provides the basic layout of the lane-departure-avoidance system with modules as described above.

When in operation, the system observes the lane-edge lines using optical sensing (a video camera), converts images into a geometric rendering of the previewed roadway layout (Roadway Geometry DSP block), and compares this with the predicted path of the vehicle over the near-future time. Decisions for warning or intervention are made on the basis of rules that consider the anticipated time-to-lane crossing (TLC), the roadway configuration, and the deduced status of the driver. The driver-status model is developed, in a training sense, early in the trip sequence by means of a continuous state identification function which is based on vehicle and roadway states as a result of the control activities of the driver. While warnings might be issued through any variety of visual, audio, or kinesthetic cues, intervention control will be accomplished by differential braking.

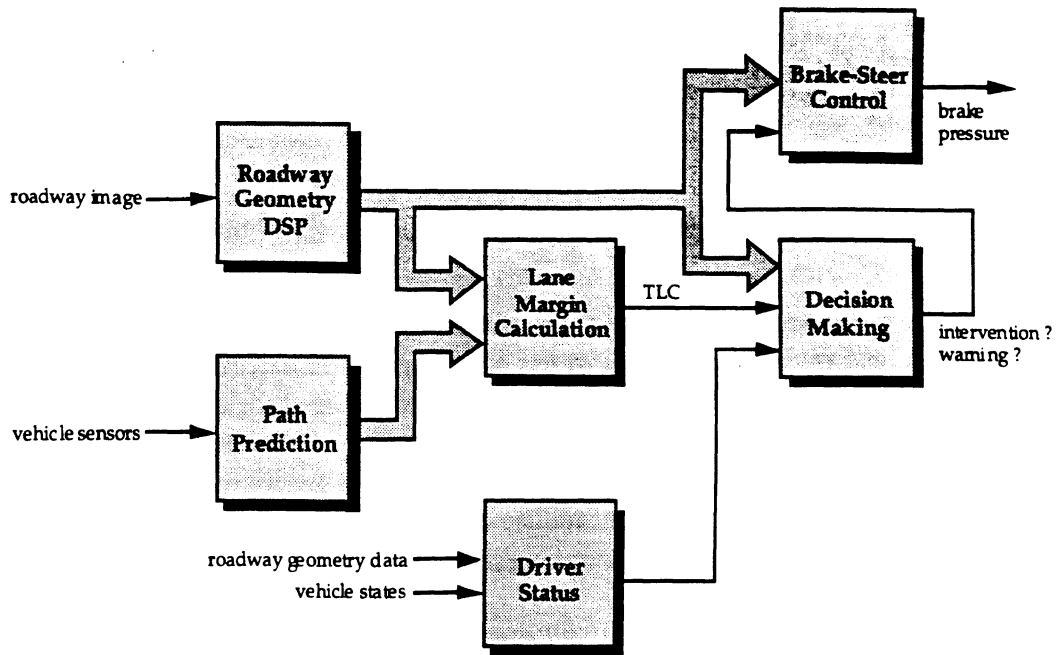


Figure 3.1.1 Basic CAPC system lay-out.

Figure 3.1.1 shows only the basic functionality of the system. A more detailed schematic of the CAPC system is shown in figure 3.1.2, which also includes the signals of interest for each module, especially a more detailed description of the roadway geometry signal processing.

The lane marker data is provided by the lane-mark sensor subsystem (LMS), which uses a single digital, high-resolution, black and white CCD camera and image processing to track lane-marks (paint marks indicating lane and road boundaries). The system uses a flat-earth model to convert image-coordinate locations of the lanemarks into vehicle coordinates; pitch and roll estimates are passed from the CAPC system to the LMS to correct for low frequency vehicle vibration. A distinction is made between near-range and far-range image data, both in this schematic and throughout this report. The near-range lane marker data consist of x - y coordinates of lane markers typically in the range of 5 to 20 meters ahead of the vehicle. The far-range lane marker data includes data from 20 up to 100 meters ahead of the vehicle.

The near-and far range data sets are used for different applications. The lane-marker data from the far range is used for the lane-margin (TLC) calculations. Intersections between the projected future vehicle path and the perceived road edges typically occur in the far-range if the CAPC system is operated on highways. When closing the loop during an intervention, the lane-marker data closer to the vehicle (near range) is of importance because the system closes the loop around errors in the current vehicle position and heading relative to the road edge.

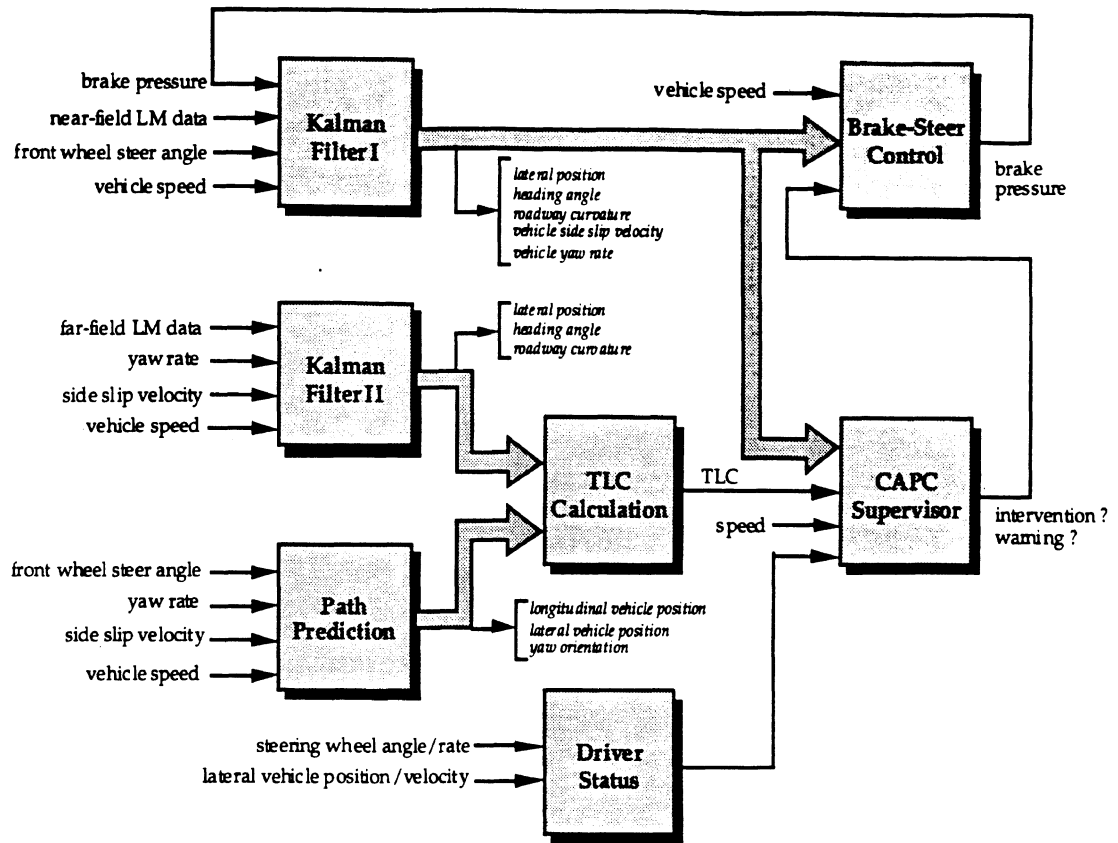


Figure 3.1.2 Detailed schematic of the CAPC system

Kalman filter techniques are used to match a model of the roadway geometry with the perceived lane-mark data. The Kalman filter is a recursive computation, which makes use of a state model to convert real-time measurements into updated estimates of system states. For CAPC, two Kalman filters are used -- a so-called far-range Kalman filter, which estimates road geometry in the far-range, and a near-range Kalman filter which estimates both the near-range road geometry and the vehicle position and heading with respect to the road edge. It also provides improvement of vehicle rate knowledge. Both near- and far-range Kalman filters contain a simple three-state roadway geometry model with the lateral relative position, relative heading, and curvature as states. Besides the roadway geometry model, the near-range Kalman filter contains a simple two-degree-of-freedom vehicle model.

The design of the Kalman filters will be described in detail in section 3.3 (far-range) and section 3.5 (near-range). Section 3.2 presents the lane-mark sensor (LMS) subsystem. The LMS includes a high-resolution CCD camera and image processing, and provides the lane marker data to both Kalman filters. Section 3.4 describes the driver-state assessment and section 3.5 discusses the subsystems necessary to close the loop during an intervention (near-range roadway geometry and differential braking controller). The decision module is presented in section 3.6. Finally, each of these modules are modeled and integrated in a

simulation tool. Section 3.7 describes the simulation requirements, architecture and features.

3.2 Design of the Lane-Mark Sensor (LMS) Subsystem

3.2.1 Design Requirements

The Lane-Mark Sensor (LMS) was specifically engineered to support the CAPC Prototype functional objectives. The following LMS requirements were stipulated:

- sense lane-marks up to 100 meters in front of a vehicle operating on a test track traveling at interstate highway speeds during good daylight weather
- update lane-mark data every 100 milliseconds
- detect the position of lane-marks at 2 meter intervals in the near-range (6 to 20 m), and at 10 meter intervals in the far-range (30 to 100 m)
- correct lane-mark positions for vehicle pitch and roll
- report lane-mark positions in vehicle coordinates, assuming a flat earth model of the road

3.2.2 Lane-Mark Sensor Design

Design Alternatives

A number of different design alternatives were considered. The following objectives were used in evaluating the alternatives: meet all requirements; support easy software development; and utilize simple, low-cost hardware.

The following sections outline a few of the alternatives considered for components of the LMS system.

The Camera

Since the CAPC objectives require imaging from a moving vehicle, the camera needed to possess a full-frame “snap-shot” exposure capable of capturing a full frame without any blurring or interlace problems commonly associated with a standard TV camera.

The camera also needed a highly accurate pixel registration since the (x,y) position was derived from a calibration of the image position. The best way to obtain this registration accuracy is by using a digital-readout camera. Digital readout also has the advantages of being immune to almost all electrical interference in the vehicle, as well as producing a superior image quality over an analog camera.

The pixel resolution of the camera was also a consideration. The resolution had to be fine enough to give adequate sampling of the lane-mark at the 100-meter range for reliable detection while giving sufficient roadway coverage. This resolution could have been accomplished by one or multiple cameras. The use of two cameras, each having a different field-of-view, was a design alternative considered.

Frame Grabber & Image Processing Hardware

Another option utilized DataCube image processing hardware on a VME-bus chassis. Besides its relatively high cost as a frame grabber, the architecture would require a costly computer specially made for the VME-bus. This type of system would not support easy software development, would be relatively expensive, and would be unnecessarily complex.

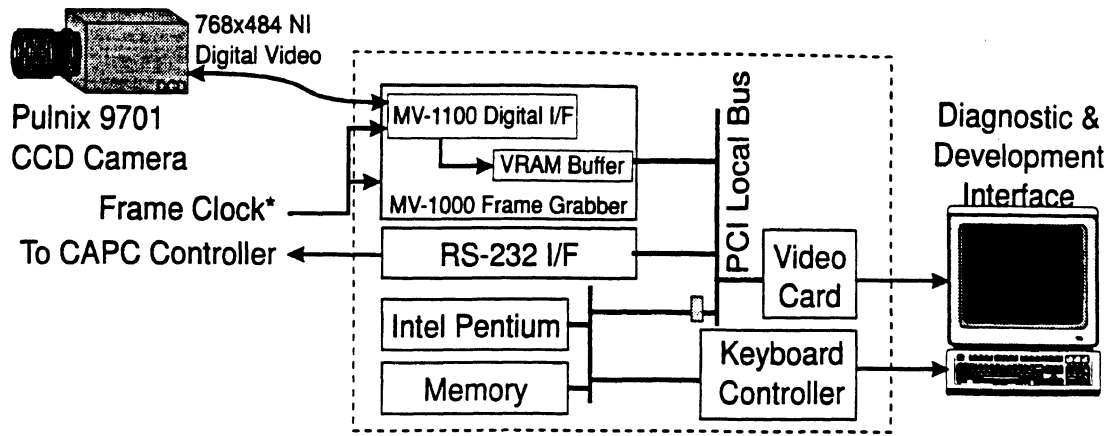
An ERIM-provided CYTO HSS specialized image-processing computer was considered, but the processing power of this machine far exceeded the requirements of the CAPC prototype. The cost of such a system would also be large, and future development of the LMS system would require it to be replaced due to the cost of such a computer.

Another option was a digital frame grabber with no special image processor that was accessed by a standard Pentium PC via the local-bus. The Pentium processor would perform all the required image processing for real-time operation. It has the advantage of being low-cost, capable of supporting easy software development, and supportive of future system utilization.

The Selected Design

The selected LMS system consists of a Pulnix 9701 digital CCD camera (768 x 484 pixels), a MuTech MV-1000/MV-1100 PCI-bus digital frame-grabber with interconnect cable, and a PCI-bus with an Intel 100 MHz Pentium computer. The LMS system components are shown in figure 3.2.1 and figure 3.2.2. Significant features of the digital camera are (1) exposure interval control, (2) exposure timing control (asynchronous reset), and (3) full-frame "snap-shot" readout (progressive scan). The frame-grabber provides software control of the exposure interval, the ability to trigger the timing of the exposure by an external TTL signal, and capture of the digital image into memory space that is directly accessible to the CPU. All LMS processing functions are performed by the Pentium CPU.

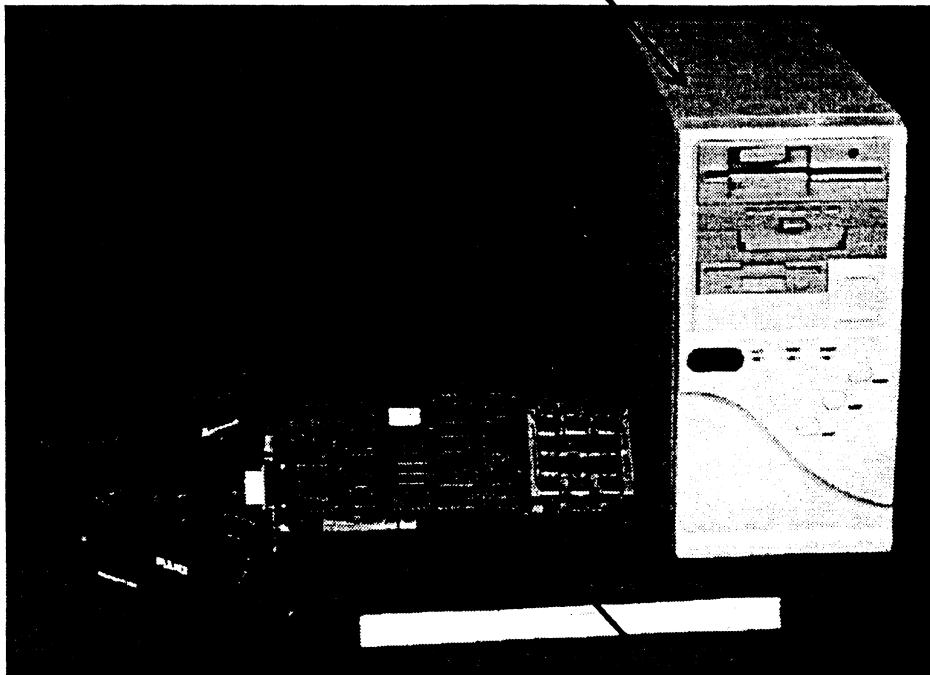
The camera is rigidly mounted inside the CAPC test vehicle near the rear-view mirror and views the roadway at a slight horizontal depression angle. Interior mounting provides the benefits of isolation from environmental elements and utilization of the vehicle's windshield wiper system. Maximizing the height of the camera position above the road surface improves the imaging perspective for long ranges; the CAPC system routinely locates highway-quality lane marks to a distance of 100 m and can function out to 150 m (under good lighting and pavement conditions).



* Frame Clock async. resets camera and forces a capture of the next immediate frame from the camera.

Figure 3.2.1: LMS System Block Diagram

PCI-bus 100 MHz Pentium PC (<\$2k)



Pulnix TM-9701
Digital CCD Camera
with Cabling (\$2.8k)

MuTech MV-1000
PCI-bus Frame Grabber
with MV-1100 Digital
Camera Interface (\$3k)

Figure 3.2.2: LMS System Components

Performance Attributes

The Pulnix digital CCD camera and MuTech framegrabber possesses unique capabilities which are essential to the detection of lane marks up to and in excess of 100 meters:

- 1) full-frame "snapshot" exposure initiated via external TTL signal
- 2) short "freeze-frame" exposure length controllable by software
- 3) analog-to-digital conversion synchronized with CCD pixel readout

Together, these features make possible the capture of sharp roadway images in spite of highway speeds and vehicle angular motion, as seen in Figure 3.2.3. Angular camera displacements are particularly troublesome for the detection of lane marks at long range; short exposure times mitigate the blurring. Software exposure control permits adjustment to enhance the detection of lane marks, in spite of bright objects (e.g. sky) in the scenes. Conventional auto-gain or auto-iris responds to peak scene values.

The lens selected to support CAPC provides a 40 degree horizontal field-of-view. Neighboring pixels subtend a horizontal angle of 0.96 mrad. At a range of 100 meters, the ground-sampled distance corresponding to the horizontal sampling is approximately 9.6 cm. Lane-mark widths are typically 10 cm, thus approximately one sample of lane-mark is expected at 100 m range.

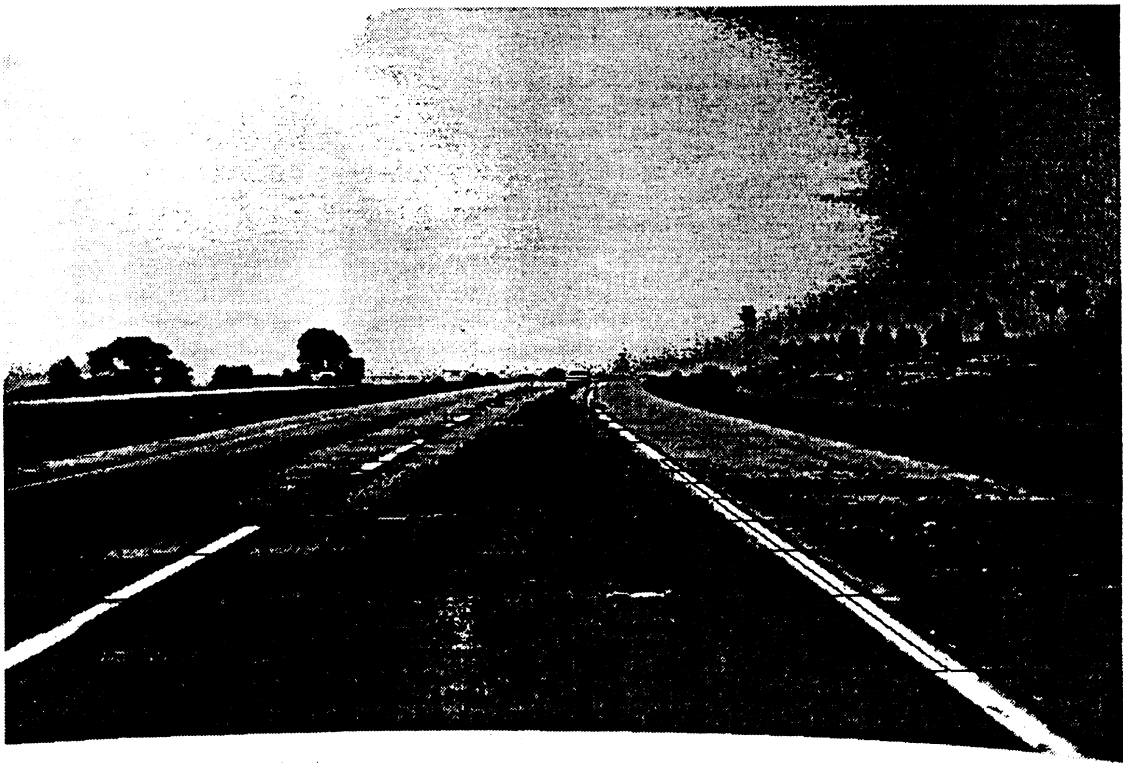


Figure 3.2.3: Example Image from LMS Camera

The lane-mark detection algorithm exploits the relatively uncluttered (and fine spatial sampling of the) near-range image to locate the lane marks over the range of 6 to 20 m. The near-range lane marks are then "followed" out to a distance of 100 m. The near-range algorithm also exploits the fact that the vehicle lateral deviation and heading angle undergo relatively small changes over a 100 msec interval. The knowledge of heading angle and lateral deviation from the previous frame is used to select small portions of the current frame for lane mark processing, minimizing computational requirements, and permitting the use of a conventional microprocessor.

3.2.3 Location and Tracking of Lane Marks

The horizontal regions overlaid in red in Figure 3.2.3 indicate the current search regions; the down-range separation is two meters. Due to the PCI bus architecture of the MuTech digital frame-grabber, the captured image is directly accessible to the Pentium processor. An assembly-level routine is used to locate the position of the lane-mark within the linear search region. The lane mark search proceeds as follows:

- 1) A vector representing the differences between successive pixels is calculated.
- 2) The maximum positive and negative transitions are recorded.
- 3) The mean of the absolute value of the maximum and minimum transitions is calculated.
- 4) The pixel midway between the maximum and minimum transition is located.
- 5) If the mean of the transition exceeds a (suitable) threshold and the width of the lane mark is less than 3 times the expected width. The pixel location is passed to a routine which marks the location in the image with a vertical yellow line (see figure 3.2.2) and converts the coordinates to the vehicle coordinate system.

Tracking Lane Marks

If four or more lane marks are found by the near-range, lane-mark search, a least squares linear fit (in vehicle coordinates) is made to determine lateral deviation of the vehicle CG from the right lane mark and the vehicle heading angle. Curvature in the lane mark (which is minimal over the near-range) is ignored. The heading angle and lateral deviation values are used to position the search regions for the successive frame. Once initialized, the search windows track the vehicle motion.

Initialization

Real-time, lane-mark location (and tracking) at 10 Hz (or faster) becomes tractable using a conventional microprocessor if the initialization process is permitted to exceed the 100-msec cycle time; initialization times up to a few seconds are insignificant relative to operating times of minutes (or hours). During initialization the following process occurs:

- 1) exposure setting
- 2) initial lane mark acquisition

The exposure setting process involves the acquisition of successive frames of images while adjusting the exposure time to achieve selected grey-scale values in the image. The exposure is first set to achieve a road surface value of 80 counts. Next, the lane mark search is initiated and lane marks located. The exposure is then adjusted to achieve a value of approximately 200 for the right hand (white) lane mark.

Although it is possible to vary (the assumed) heading angle and lateral deviation during the search process, the width of the search allows for considerable latitude in vehicle position during the initial acquisition of the lane marks; the vehicle must be nominally in the center of the lane and parallel to the roadway during initialization.

3.2.4 Calibration

The LMS converts the lane-mark locations in the image (row, column) to displacements in the vehicle coordinate system. As shown in Figure 3.2.4, the vehicle coordinate system origin is at the vehicle CG and the x-axis is aligned with the vehicle heading axis. The y-axis lies in the plane of vehicle motion forming a right-handed orthogonal coordinate system (as viewed from below).

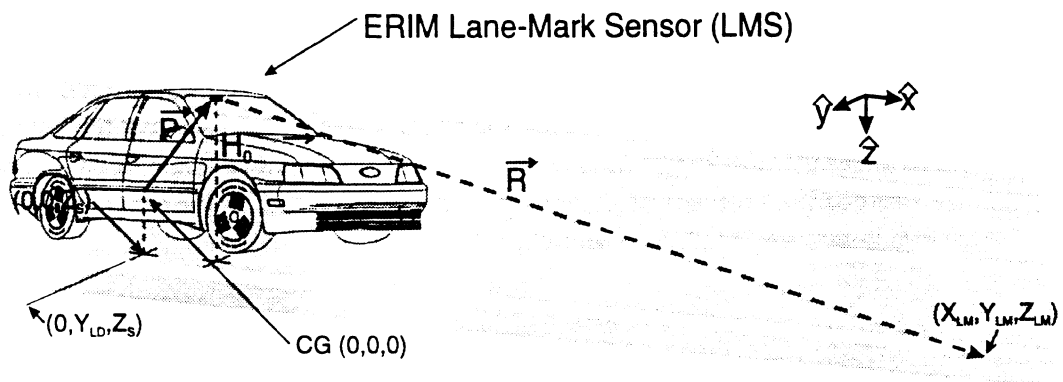


Figure 3.2.4. Vehicle Coordinate System

Since the LMS CCD sensor measures angular displacements to objects in the scene (two parameters) transformation to a rectilinear vehicle coordinate system (three parameters) requires the assumption of a model for the road surface. For implementation with CAPC, the road surface was assumed to be flat (or reasonably close) for this transformation.

Assuming a "flat earth" each pixel (row and column) can be associated with a down-range (x) and cross-range (y) displacement from the vehicle CG. Transformation from image row and column to the vehicle coordinate system proceeds as follows:

- 1) Row and column values are transformed (rotated) as required by the vehicle roll angle (provided to the LMS).
- 2) Row values are transformed (translated) as required by the vehicle pitch angle (provided to the LMS).
- 3) The transformed row and column values are converted to rectilinear coordinates (assuming a flat earth.)

Calibration of the LMS system was accomplished and verified at the Ford Dearborn (dogbone) test track . A reflective hemisphere was placed at the center of the roadway at a 50 m displacement down-range from the vehicle CG; the vehicle was positioned so the sun was illuminating from the rear of the vehicle and a bright reflection of the sun was observed to illuminate a single pixel on the LMS engineering display. From the row location of this pixel, the depression angle of the camera optical axis (with respect to horizontal) could be accurately determined; from the column position and a sighting along the vehicle longitudinal axis (using plumb bobs) to determine the lateral displacement, any offset in azimuth (or yaw) angle of the optical axis with respect to vehicle heading could be determined. The LMS software was updated with the precise angles and the calculated downrange and lateral offset displacements were observed to yield the correct results. If the camera is removed from the mount or adjusted in position, this calibration must be repeated.

3.2.5 Software Documentation

The LMS software is documented in this report in four places. The specifications for communication with the CAPC host are included as appendix B; mechanical and electrical interface specifications are presented in appendix C. A flowchart is also presented in section 8.3.2, and a listing of code is included as appendix E.

3.3 Design of the Lane Margin Processor

The time to lane-crossing (TLC) is used to measure the "lane tracking margin" of a vehicle, and is computed after the arrival of each set of data from the far-range portion of the lane marker subsystem (LMS). The TLC is defined as the remaining time until the center of mass of the vehicle will reach either edge of the roadway, and is estimated by comparing the road geometry model (derived from vision data, as described in section 3.2) and the predicted vehicle path, which will be described later in this section. In the current CAPC vehicle prototype, the decision to issue a warning or indicate when an intervention would occur is based upon the TLC. Our plan for the future is to include a driver-state assessment in the decision making process.

This section presents a brief summary of how TLC is computed. Much work has been done to develop algorithms for computing TLC and to assess the performance. In this section we refer to several appendices which describe in detail both the heart of the algorithms, the performance predicted by simulation, and supporting studies. Much of this is included in appendix I -- a doctoral dissertation defended in 1995 and based on the CAPC problem. Additional details can be found in numerous papers based on this work, which are referenced as appendix I.

There are several motives for using TLC as a decision criterion instead of, for example, the lateral position of the vehicle in the lane. First, TLC is a *predictive* measure which considers not only the current vehicle position and the current roadway shape, but also considers the steering input and the roadway far ahead of the vehicle to anticipate future road departures. This prediction aspect makes the computations and the sensing more complex -- more sensor range is required, a steering sensor is needed, and computations must be done to predict the intersection of the roadway and the vehicle's trajectory. But many practical advantages are gained. The use of TLC accommodates different driving behaviors by allowing drivers to wander about the lane, in any normal pattern of steering, as long as complete lane-departure is not imminent. For example, some drivers may hug one side of the lane which would result in false alarms if only the lane position were considered. But the use of TLC allows this behavior -- until the driver either turns the steering wheel too far toward the edge or until the driver is not reacting to changing roadway curvature. Therefore the predictive nature of TLC provides improvement in performance of the system.

Additionally, given a road-departure system which uses prediction, the use of TLC simplifies the decision process. TLC is a single variable -- one that can be understood intuitively -- and the decision to warn or intervene becomes a simple process. Tuning the decision process in the prototype vehicle, for example, was accomplished in a matter of hours. This simplicity, of course, derives from the fact that the TLC computation has boiled down all the analog transducer information and all the roadway image information into a continually updated scalar measure of the vehicle's proximity to a road departure.

In appendix I, Lin provides a review of the use of TLC for vehicle safety systems. Significant advancements in the understanding and computation of TLC and its levels of

uncertainty have occurred as part of the CAPC project. Lin developed many of the TLC algorithms used in both the CAPC prototype vehicle and in the CAPC simulation code. In addition he has completed work not included in either, and has established important facts about TLC used directly in the prototype vehicle (for example, the use of 10Hz image rates). This section summarizes the work on TLC for CAPC. Throughout this section we make a distinction between algorithms that are included in the CAPC prototype vehicle and other algorithms that have been tested extensively in simulation, but which are not included in the prototype vehicle. For now, the vehicle includes only the most necessary parts of the TLC algorithm; this is to help increase computational speed and simplify the processes of tuning and testing.

The TLC is computed by comparing the estimated road geometry with the predicted vehicle path to find any intersections in the near future (up to four seconds). The road geometry is estimated using measurements at 10Hz; the data includes LMS reports of lane edge coordinates, vehicle yaw rate and forward speed, and front wheel steer angle. Road edge geometry is estimated as either a second or third order polynomial curve fit. The predicted vehicle path is computed from using current vehicle motion estimates (yaw rate, vehicle speed, steer angle) and propagating the path into the future, assuming that the steer angle is constant. The propagation is done using a simple two-degree-of-freedom dynamic model with a linear tire/road interface model. Both lane-geometry estimation and vehicle-path prediction are described in more detail in the subsections following this section. The reader is referred to appendix I for more details. First, however, the following subsection provides an overview of the TLC computation.

3.3.1 Time-to-Lane Crossing Computation

A flow chart describing the TLC calculation algorithm is shown in figure 3.3.1. The TLC is obtained by first projecting forward the vehicle path until it intersects the lane boundary. This provides a rough estimate of the TLC. Then, based on this TLC estimate, an interpolation scheme is applied to refine the TLC such that the TLC error due to projection time step can be nearly eliminated. If the predicted vehicle path does not intersect the estimated edges of the roadway within four seconds, the TLC is assigned the saturation value of four seconds. TLC values of greater than four seconds are not useful, since these values indicate that the vehicle is in no immediate danger of leaving the road.

Appendix I describes the interpolation scheme used for TLC values less than four seconds. This new method reduces the computations required to obtain an accurate TLC value. Also in the appendix, the bandwidth of TLC is identified and used to show that the required sampling rate of measurements supporting the TLC computation must be 10 Hz. This sampling rate is implemented in the CAPC prototype system.

The major sources of error affecting the TLC are discussed in appendix I. These include the image measurement errors of the roadway (due to pitch and roll, superelevation, or grade) and vehicle-path-projection errors (due to wind and roadway disturbances, as well as vehicle-motion-measurement error). The CAPC prototype includes on-line estimation of

pitch and roll using LVDT deflection transducers at each wheel. These LVDTs measure the travel between the sprung and unsprung masses at each wheel; section 8.3.3 describes the approximation of pitch and roll from these measurements. The pitch and roll is used in the LMS to correct the reported positions of points on the lane marker. Superelevation effects, which are only significant on test tracks, are reduced by using a Kalman filter to combine the image information in the far-range with vehicle motion data. This same Kalman filter acts as a lowpass filter to further reduce effects of random measurement error in pitch and roll, as well as the vehicle-motion measurements (yaw rate, vehicle speed, and steering) used in projecting the vehicle path. In addition, Lin has developed an on-line estimation scheme for computing the magnitude of external disturbances acting on the vehicle which affect the vehicle path projection. This scheme,

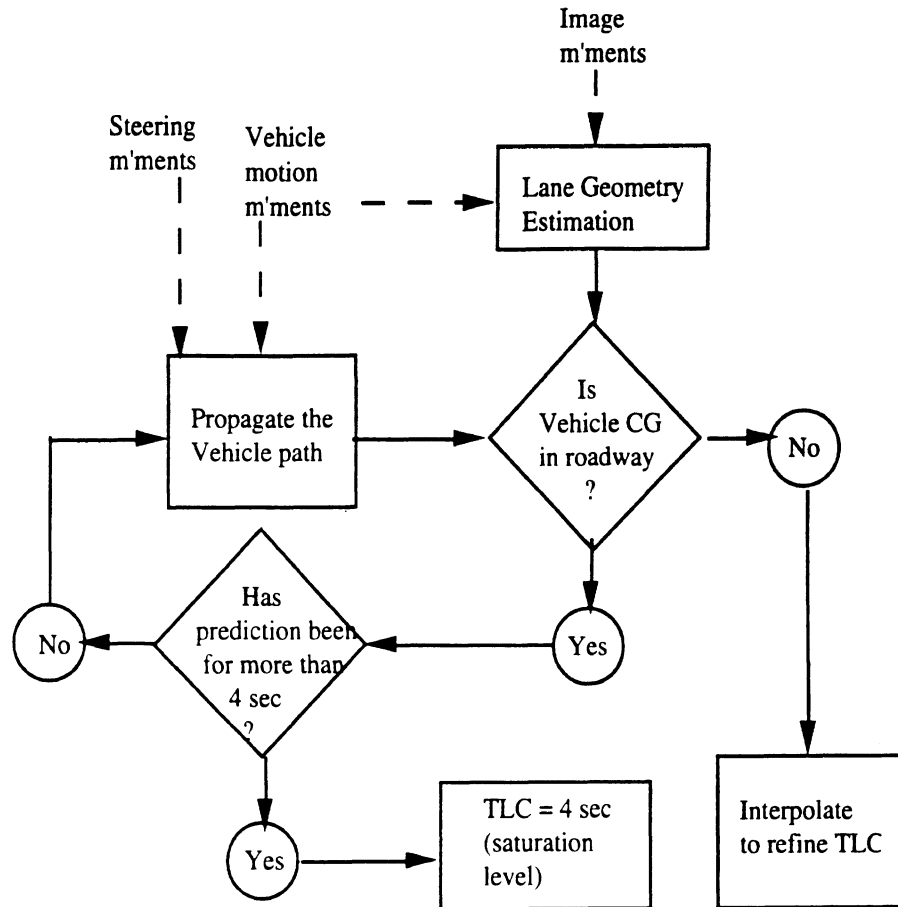


Figure 3.3.1: TLC Calculation Algorithm

however, is not included in the CAPC vehicle, but it is discussed later in the subsection on vehicle path prediction.

3.3.2 Lane Geometry Modeling

Lane-geometry modeling is the part of the TLC computation that represents the road edge in the neighborhood of the predicted point of intersection. Lane geometry modeling is performed after the arrival of each set of LMS data -- that is, at 10Hz. The output is a set of three coefficients, which represent each of the two road edges as a polynomial, as seen in vehicle coordinates. The inputs include the lane marker positions reported by the LMS, yaw rate, vehicle speed, and front wheel steer input. The steering and vehicle motion inputs are required because lane geometry modeling is computed using a Kalman filter which combines all the measurements using a model of vehicle dynamics and kinematics. The Kalman filter was developed to increase the performance of the system by reducing the sensitivity of lane geometry modeling to errors due to pitch and roll, superelevation, and so on.

Furthermore, an uncertainty characterization for the lane-geometry representation has also been developed by Lin so that the TLC uncertainty can be predicted on-line (see Appendix I). This has not yet been implemented on the vehicle. The uncertainty of the lane geometry model is a three-by-three covariance matrix for each road edge.

Figure 3.3.2 shows the structure of computations for lane-geometry modeling and the characterization of its associated uncertainty. Image processing (described in section 3.2) reports locations of points on the lane markers. Points in the near-range -- up to 20 m ahead of the vehicle -- are incorporated along with vehicle motion and steering sensors into a near-range Kalman filter. This Kalman filter provides estimates of the vehicle's position and heading with respect to the roadway, as well as the yaw rate and lateral velocity. These vehicle motion estimates are used in lane geometry estimation, vehicle path prediction, and brake-steer intervention control. A detailed description of this filter is provided in section 3.5.2.

Lane geometry is estimated using a second Kalman filter, which is referred to as the far-range Kalman filter. The far-range Kalman filter uses the estimates of vehicle position and heading and vehicle rates provided from the near-range Kalman filter to combine data reported by the LMS over successive images. This data describes far-range lane marker positions -- 20 to 100 m ahead. The purpose of using two Kalman filters is to optimize the accuracy of both local information (vehicle motion and position within the lane) and previewed information (roadway geometry far ahead of the vehicle), since the models of roadway geometry only approximate the true shape.

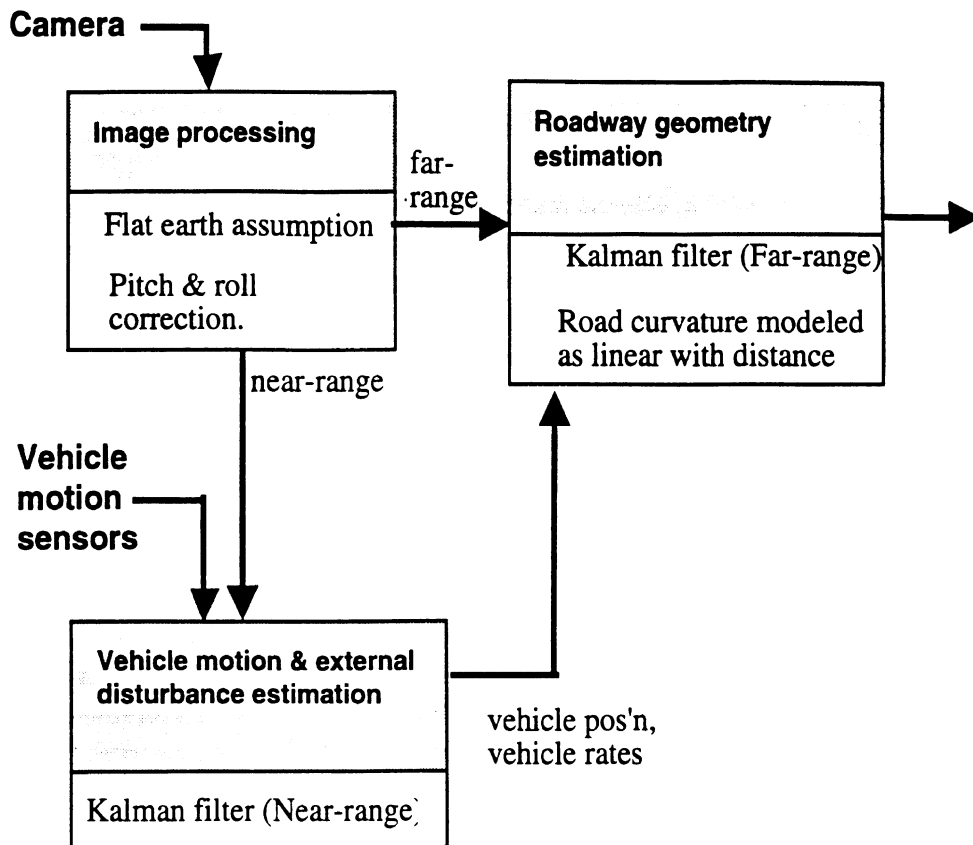


Figure 3.3.2: Structure of the Lane Geometry Modeling Computation

The use of a far-range Kalman filter is the result of long examination of estimation approaches. In Appendix I, simulation is used to compare the Kalman filter used in the CAPC prototype with a simpler approach -- using only LMS data from a single frame -- and with a more complicated model of the lane geometry. The design used was chosen for performance and simplicity. In that appendix, the effects of large superelevations often found on test tracks are examined as well. The Kalman filter provides some advantages for highly superelevated roadways, although there can be a loss of performance in tight transitions between curves and straightaways that often occur on test tracks.

The TLC uncertainty is due in part to the uncertainty in the lane geometry estimate. The lane geometry uncertainty, in turn, is due mainly to errors in the lane marker positions reported by the vision-based LMS subsystem. A model of the errors in lane marker position determination, as a function of lane marker position, is used to compute a set of standard deviation values for the polynomial coefficients. This uncertainty algorithm uses the model of LMS uncertainties, which considers the effects of the camera's pixel resolution, vehicle vibration, and roadway roughness on the uncertainty of the lane marker locations. An algorithm has been developed for the characterization of the lane marker location uncertainty and subsequently the uncertainty of the roadway geometry, however the CAPC vehicle does not contain the uncertainty computations. Details about the computation of uncertainty of the lane geometry model are in Appendix I.

3.3.3 Vehicle Trajectory Prediction

The vehicle's trajectory is predicted using a simple model of lateral vehicle dynamics and measurements of the steer angle and the vehicle's current heading and displacement with respect to the road edge. Figure 3.3.1 earlier showed that vehicle path projection is used to compute the TLC by propagating the path (based on constant steer angle) until either the vehicle CG crosses the roadway edge or until projection has continued for over four seconds. Here we describe the process of propagating the vehicle path.

Figure 3.3.3 below shows that vehicle path projection uses the near-range Kalman filter described in the previous section (and in Section 3.5 in detail) to provide the current vehicle heading angle, lateral displacement, and vehicle rates. The steer angle is assumed constant during the prediction interval, and takes a value provided on-line by a transducer measuring the displacement of the steering rack. Vehicle longitudinal speed is also assumed constant during the interval. The path is projected by numerically integrating the vehicle displacement, using a simple two-degree-of-freedom dynamics model of the vehicle, which includes the yaw angle and the vehicle lateral motion. Appendix I shows that this simple model is sufficient for TLC purposes.

The vehicle lateral dynamics can be approximately described by a two-degree-of-freedom lateral dynamics model, the so-called bicycle model:

$$\begin{Bmatrix} \dot{v} \\ \dot{r} \end{Bmatrix} = \begin{bmatrix} \frac{1}{mu} (C_2 + C_1) & \frac{1}{mu} (-bC_2 + aC_1 - mu^2) \\ \frac{1}{I_z u} (aC_1 - bC_2) & \frac{1}{I_z u} (a^2 C_2 + b^2 C_1) \end{bmatrix} \begin{Bmatrix} v \\ r \end{Bmatrix} + \begin{Bmatrix} \frac{-C_1}{m} \\ \frac{-aC_1}{I_z} \end{Bmatrix} \delta + \begin{Bmatrix} e_1 \\ e_2 \end{Bmatrix} \quad (1)$$

where v is the vehicle lateral velocity, r is the yaw rate, δ is the steer angle, the C 's are the cornering stiffness of the tires (relating wheel slip angle and lateral forces on the tires), m is the lumped vehicle mass, u is the vehicle forward speed, l is the length of the vehicle wheel base, I_z is the moment of inertia about the z axis of a standard SAE coordinates system, and a and b are the distances from the front tire axis and rear tire axis to the vehicle's center of gravity respectively. The $\{e_1, e_2\}^T$ term represents disturbances acting on the vehicle's lateral motion. Major disturbance sources include wind and superelevation of the road, while minor disturbances derive from road crown, suspension misalignment, unequal tire pressures, etc. Approaches to estimating components of these disturbances are presented later in this subsection, so for the moment this term will be neglected.

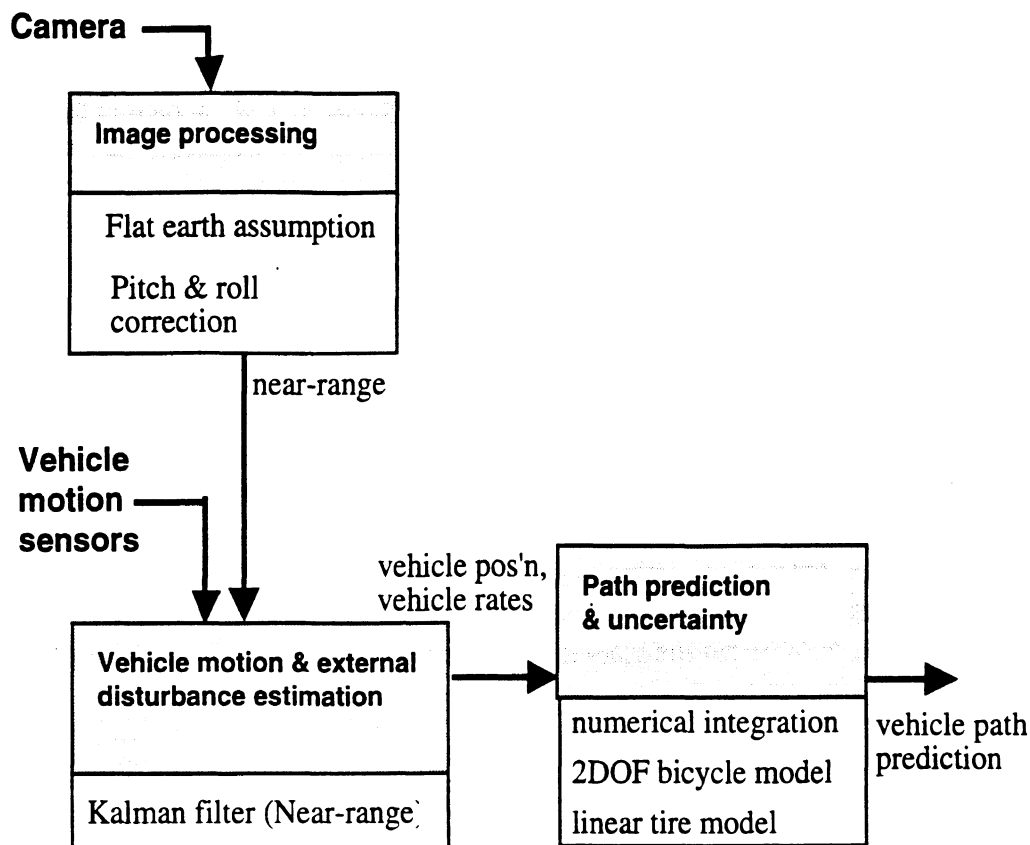


Figure 3.3.3: Structure of the vehicle path prediction computation

Three more states are added to integrate the yaw rate and the vehicle lateral velocity to provide lateral and longitudinal displacements of the vehicle. Let x and y denote, respectively, the longitudinal and lateral displacements of the vehicle in the vehicle's coordinate system, and let ψ denote the heading angle of the vehicle. The heading angle is assumed to be small; Appendix F contains a simulation study which supports the validity of this assumption for moderate vehicle maneuvers at highway speeds. Three states can then be added:

$$\dot{x} = u$$

$$\dot{y} = v + u\psi$$

$$\dot{\psi} = r$$

The propagation of the vehicle path is done using an interpolation scheme which first uses a large time step for propagation, then, after the approximate time of lane-crossing is identified, uses a smaller time step to compute a refined value. This approach was developed to provide accurate TLC values while minimizing computation time. The applied interpolation scheme for refining the predicted vehicle path is introduced in Appendix I.

Additional computations have been developed to estimate the uncertainty in the predicted path. These are not yet included in the CAPC vehicle, but are included in the simulation tool. Appendix I describes the characterization of uncertainty of the path projection.

Disturbance Estimation

A further refinement to vehicle path projection is the on-line estimation of the lateral-force and yaw-moment disturbances. These disturbances, which arise from effects such as wind, crown, and superelevation, are modeled as unchanging over the period of path projection (less than four seconds). Implementation involves augmenting the Near-range Kalman filter with two additional states to be estimated: the lateral disturbance force acting on the vehicle, and the yaw moment disturbance. In Appendix I, extensive simulation studies show that for situations in which these disturbance sources are strong, the use of on-line disturbance estimation can significantly improve the accuracy of TLC.

Adaptive Vehicle Modeling

The underlying assumption for the path projection algorithm and the disturbance characterization algorithm is that accurate estimates of vehicle parameters (inertia and tire cornering stiffness) have been given. To obtain these parameter estimates accurately, it is necessary to monitor the vehicle status and road environment, and to update these vehicle parameters accordingly. As part of this, on-line estimation of road/tire characteristics is important for implementation of the TLC calculation and the overall lane-departure warning system for a large fleet of vehicles. This is referred to as the adaptive vehicle modeling issue, and is the main concern of this subsection. This work has not been incorporated into the CAPC prototype vehicle, but has been presented in a conference paper, Appendix F. An introduction to the work is provided here.

A disturbance observer has been developed to identify the road-surface friction coefficient. The effect of the road surface condition on vehicle path prediction (measured in TLC) is the main performance evaluation metric. The vehicle path prediction is obtained based on the 2-DOF lateral dynamic model (bicycle model). To calculate the TLC the longitudinal tire force is first estimated from a single wheel model. The road friction coefficient and the tire lateral force are then calculated based on an anisotropic model. Based on the estimated lateral force, the cornering stiffness is then obtained. This updated cornering stiffness is then used in the bicycle model to compute the TLC. In the estimation scheme, the wheel speed and torque are assumed to be available (measured or calculated). From the measured wheel speed and the estimated vehicle velocity the tire-slip ratio can be calculated. Vehicle velocity is derived by the undriven wheel under driving conditions or estimated during braking conditions.

The methods to estimate the road surface conditions involve a recursive least square method using a forgetting factor and an enhanced adaptive observer which can improve the performance of the observer based on a linear relationship between the output and input signals. From simulation results we found that the inclusion of the road friction estimation

scheme will improve the TLC accuracy significantly when the road is slippery and there is a fair amount of steering applied. The TLC can be improved up to 0.5 second. We observe that the TLC can be improved more significantly when TLC is large. This means that the updating of the cornering stiffness is more useful for warning than intervention control. See Appendix F for details.

3.4 Design of the Driver Status Assessment

On-line identification of driver state is used in the decision to issue a warning, initiate intervention, or do nothing. On-line identification of the driver state is also valuable for other driver assistance features such as automatic cruise control, collision warning and road departure warning. These driver assistance features are centered around the ability to perform a control-type task while at the same time keeping the driver in the control loop. Automatic cruise control takes over the function of headway control, but it remains the duty of the driver to steer the vehicle. Road departure warning can be viewed as a further extension where an additional driver support system acts as a co-pilot to monitor lane keeping performance.

A system which includes driver behavior may make it possible to accommodate different driving styles. If driver actions can be monitored, it would be possible to “personalize” the warning criteria according to driving style. In addition, it would also be possible to track changing driver parameters during a long drive. This would enable a driver assistance system to provide warning as a function of changing driver *state*.

This section provides a brief overview of this driver assessment work; Appendix G, which is a paper delivered to the 1995 American Control Conference, provides more details. At this time, the CAPC prototype vehicle does *not* include a driver-state-assessment module. The CAPC simulation tool includes a very simple driver-state-assessment module based on the work of W. W. Wierwille at Virginia Tech [4], however, because driver drowsiness is a very slow process and because desktop simulators do not easily lend themselves to testing such effects, this feature is not often used. There are plans to include the new driver-state-assessment work into the CAPC vehicle as soon as development is more complete, but for now investigative work has proceeded using the driving simulator at the Ford Research Laboratories.

Goal:

We seek a method to detect changes in driving patterns so that an assessment of driver alertness/performance can be made. This assessment will be used first as an input to a lane departure warning system. Our hypothesis is that system identification techniques can be used to form a set of driver parameters that can be correlated with various levels of lane keeping performance. Variations of these key parameters will then permit us to monitor driver state. We are currently pursuing a system identification approach, which is described briefly below and in more detail in Appendix G

Results & Discussion

Driver models found in the literature are used primarily for vehicle dynamics studies for maneuvers such as lane changes, emergency maneuvers, and general tracking [1,2,3]. The function of these driver models is to provide steering inputs in simulation so that vehicle dynamics models can be evaluated. We have investigated driver steering frequency distributions, stationarity issues, and comparisons of literature driver models and simulator

data, and have found that driver models for vehicle dynamics studies are not readily amenable to describing straight-line driving behavior.

Appendix G compares straight-line steering commands and vehicle lateral position obtained from a simulation using a driver model with the same signals obtained from a driver using the Ford Research Laboratory driving simulator. We have discovered what we call a *complacency zone* where driver model output (steering position) remains constant while lane deviation and heading angle errors build. Past some undefined threshold, the driver makes a correction. Complacency is a function of disturbance level, and will also be a function of other factors such as traffic density and road curvature. Additionally, real drivers steer with a lower frequency spectrum than driver models from the literature. Therefore these models cannot be used in on-line driver state assessment for a system such as CAPC.

Additionally, driving-simulator tests over one hour duration have shown evidence that the driver begins to allow the vehicle to wander about the lane more. This has helped to motivate the use of a system identification approach to on-line driver modeling. We have assumed an ARX structure as the candidate model structure, as shown in Figure 3.4.1:

$$A(q)y(t) = B(q)u(t - nk)$$

where $y(t)$ is the driver-model, steering-position output (δ in the figure) and $u(t-nk)$ is the delayed driver-model input (in this case lateral vehicle position, y in the figure). Indicators such as parameter values and pole/zero locations are used to track changes in driver state. Appendix G describes the work in detail. It is concluded from driving simulator experiments that such an approach may be capable of identifying changes in driving behavior with simple on-line computations.

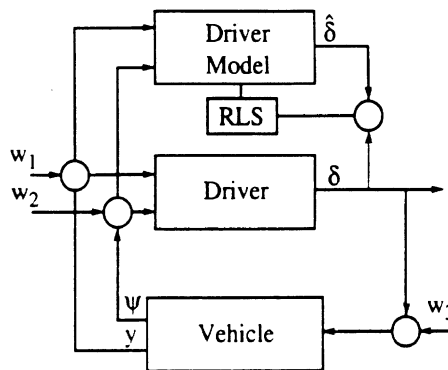


Figure 3.4.1: Driver Identification Framework

3.5 Design of the Brake-Steer Controller

This section deals with the design of a controller that operates the rear brakes to correct the path of the vehicle in case of an intervention by the CAPC system. The CAPC prototype vehicle does *not* include a completed implementation of this differential braking (or brake-steer) approach. The vehicle does include a dual hydraulic system for later brake-steer implementation, and the controller described in this section is also included in the software onboard the vehicle. The prototype, however, is missing the necessary electronic driver board to translate the controller's commanded brake pressure into electrical signals to open and close the appropriate solenoid valves. Section 4 describes the existing hardware and Section 8.4 includes schematics for an eventual implementation.

The use of brake-steer for active intervention is a natural step in the trend toward using brake valving to achieve various vehicle control safety functions. To date, ABS, 4WS, and traction control are widely available. Yaw-rate control using differential braking is available in production models from Mercedes and Toyota. Pilutti et al. first proposed this in conjunction with the CAPC project Appendix H is a conference paper describing the first work on brake-steer for lateral path control. The final brake-steer design presented in this section follows that work; more sophisticated control design was performed to account for the prototype's discrete-time implementation and long (0.20 second) delay between image information and controller command update.

First, the authority available with differential braking for lateral path control is discussed in Section 3.5.1. Second, the design of an estimator is described in Section 3.5.2. This is the "near-range" Kalman filter which has been referred to in previous sections. This uses onboard vehicle motion transducers and near-range LMS data to estimate the current vehicle position and orientation with respect to the local road edge, the local road curvature, and the side-slip velocity and yaw rate. Finally, the design of the controller which effects the differential braking during an active intervention is presented in Section 3.5.3. The controller is a state-feedback controller designed using the theory of linear optimal control. More states can be added to reject disturbances, as described in Chapter II of Appendix I. The designs of the state estimator and the controller are performed separately, as the *separation principle* of linear optimal control theory allows.

3.5.1 Brake-Steer Authority

Before discussing the details of controller design for active intervention, it is useful to determine the authority available when using brakes to steer the vehicle. A simple experiment using the CAPC simulation tool is used to determine the relation between cornering control using steer inputs and cornering control using differential braking. The relation between the front wheel steer angle δ_{fw} , the lateral acceleration A_y , and the vehicle speed U can be derived from a simple 2 DOF vehicle model with linear tire characteristics and is given by

$$A_y = \frac{\delta_{fw}}{\left[\frac{(a_v + b_v)}{U^2} - \frac{(C_{F\alpha 1} \cdot a_v - C_{F\alpha 2} \cdot b_v) \cdot m_v}{2 \cdot C_{F\alpha 1} \cdot C_{F\alpha 2} \cdot (a_v + b_v)} \right]} \quad (3.5.1)$$

where

- δ_{fw} is the front wheel steer angle (rad)
- A_y is the lateral acceleration (m/s²)
- U is the vehicle speed (m/s)

The vehicle model parameters for the prototype vehicle a Ford Taurus SHO are given in Table 3.5.1. The values for the cornering stiffnesses are valid for this particular vehicle-tire configuration. Due to compliance effects (suspension bushings, steering elasticity) the cornering stiffnesses are lower than the values valid for a stand-alone tire and therefore these numbers do represent effective cornering stiffnesses.

$C_{F\alpha 1}$	cornering stiffness of 1 front tire	53,731	[N/rad]
$C_{F\alpha 2}$	cornering stiffness of 1 rear tire	66,440	[N/rad]
m_v	total mass of vehicle	1,814	[kg]
I_{zv}	total yaw moment of inertia	3,962	[kg-m ²]
a_v	distance CG to front axle	1.073	[m]
b_v	distance CG to rear axle	1.620	[m]
T_{w2}	track width of rear axle	1.521	[m]
K_{b2}	brake gain of 1 rear brake	3.549E-4	[N/Pa]

Table 3.5.1 Vehicle parameters ('94 FORD Taurus SHO)

To characterize the performance of brake-steer actuation, a set of simulations are run in which a single front or rear wheel of a Ford Taurus SHO is activated such that the dimensionless longitudinal brake slip $\kappa = -10\%$. This slip value corresponds to a longitudinal tire force that is slightly smaller than the tire force limit. The vehicle model used in this simulation experiment contains 14 DOFs and the tire model is based on the Magic Formula with combined slip. Figures 3.5.1 and 3.5.2 show two plots of interest. With the above equation it is easy to calculate the lateral acceleration due to steering alone, as a function of the vehicle speed. These lines are shown in both plots for front wheel steer angles ranging from 0.5 to 3.0 deg. Overlaid on these lines of constant steer angle are the lateral acceleration responses simulated for three different, initial, vehicle speeds (60, 90, 120 km/h) while one wheel (front or rear) is braked with 10 percent slip. The steering wheel is held fixed in the straight ahead position. Due to the longitudinal braking force at one wheel, the vehicle decelerates and corners at the same time. The plotted responses illustrate that 10 percent rear wheel slip provides a cornering ability similar to steering the front wheels with about 1.1 degree of front-wheel steer angle. Ten percent slip of a single front wheel corresponds to about 1.5 degrees front-wheel steer angle.

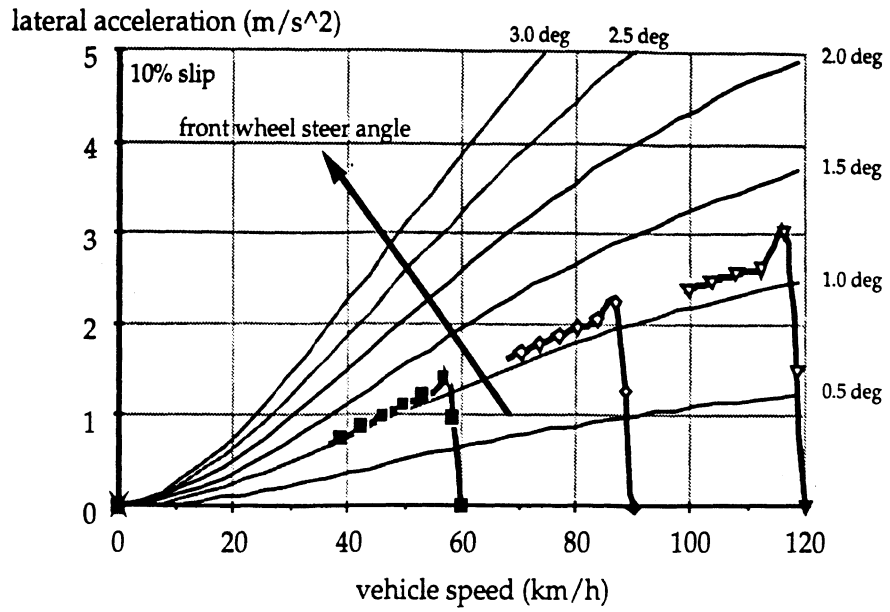


Figure 3.5.1 Brake-steer authority with 10% rear wheel slip.

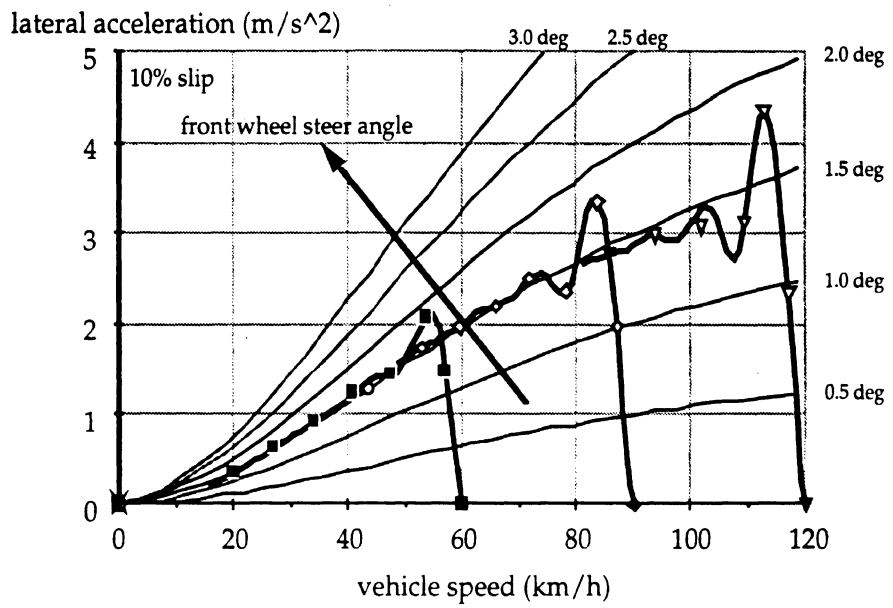


Figure 3.5.2 Brake-steer authority with 10% front wheel slip.

The deceleration that occurs when braking one front wheel is much larger ($\approx 0.3g$) than when braking a rear wheel ($\approx 0.15g$). This is due to the fact that the static front-tire load is about 50% larger than the static rear-tire load. The vertical tire load of the rear wheel that is braked is also reduced due to braking (load transfer from rear to front wheels) and due to cornering (load transfer from right to left wheels in this case). The higher the speed of travel, the higher the level of lateral acceleration is and thus the larger the tire side force. This means that the brake force will also decrease due to combined slip effects. So, while braking a single rear wheel, the normal load of this particular wheel is the lowest of all four wheels on the vehicle.

For typical highway applications the authority of braking one rear wheel should be sufficient to recover the vehicle from inadvertent road departures given that the angle of attack (heading) between vehicle longitudinal axis and roadway axis is reasonable shallow. If more steer performance is required, the front wheels might be used to accomplish a more powerful steer effect under the requirement that the steering wheel is held firmly fixed. Depending on the front suspension geometry, the vehicle might counteract the brake-steer induced concerning motion if the driver does not hold the wheel fixed. In that case brake-steer corrections will fail and a road departure cannot be avoided.

This analysis as presented above doesn't account for any influence of the geometry of the suspension (such as the scrub radius) or elastokinematic properties (bushing compliances) on the cornering ability using brakes. Both compliance and geometry effects might affect the authority of brake-steer control significantly. The presented result did account for the compliance in the steering system associated with aligning moments.

3.5.2 Near-range Kalman Filter Design

A Kalman Filter [2] is used to estimate unknown or unmeasurable states of the vehicle-roadway system. The following roadway model related states (Figure 3.5.3) are of interest:

y_e	lateral position error	[m]
Ψ_e	heading error	[rad]
κ	curvature	[1/m]

The vehicle dynamics related states of interest are:

v	side slip velocity	[m/s]
r	yaw rate	[rad/s]

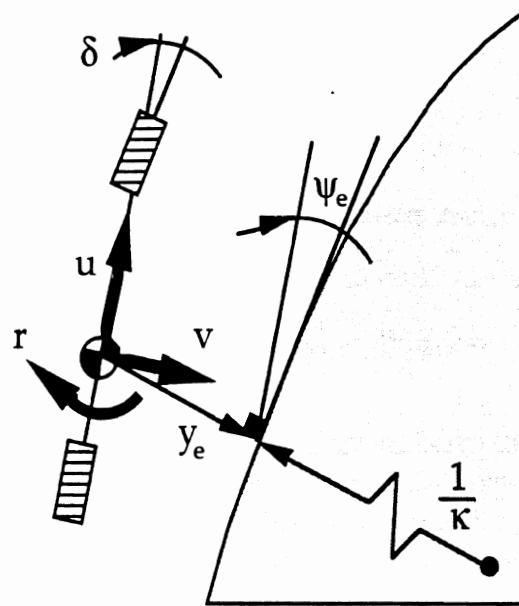


Figure 3.5.3 Roadway geometry and vehicle model.

The relation between the states and the derivatives of the states are described by the equations of motion of the vehicle and roadway model. The road-model-related differential equations are

$$\dot{y}_e = -v - U\Psi_e \quad (3.5.2)$$

$$\dot{\Psi}_e = U\kappa - r \quad (3.5.3)$$

$$\dot{\kappa} = 0 \quad (3.5.4)$$

where U is the speed of travel of the vehicle (m/s), r the yaw rate of the vehicle (rad/s) and v the side slip velocity (m/s) of the vehicle. This road model assumes a constant curvature (κ). The dynamics of the vehicle may be described by a 2 DOF flat vehicle model with equations of motion given by

$$\dot{v} = -\left(\frac{2C_{F\alpha 1} + 2C_{F\alpha 2}}{m_v U}\right)v - \left(\frac{2C_{F\alpha 1}a_v - 2C_{F\alpha 2}b_v}{m_v U} - U\right)r + \frac{2C_{F\alpha 1}}{m_v}\delta_{fw} \quad (3.5.5)$$

$$\dot{r} = -\left(\frac{2C_{F\alpha 1}a_v - 2C_{F\alpha 2}b_v}{I_{zv}U}\right)v - \left(\frac{2C_{F\alpha 1}a_v^2 + 2C_{F\alpha 2}b_v^2}{I_{zv}U}\right)r + \frac{2C_{F\alpha 1}a_v}{I_{zv}}\delta_{fw} + \frac{T_{w2}K_{b2}}{2I_{zv}}P_b \quad (3.5.6)$$

where δ_{fw} is the front wheel steer angle (rad) and P_b the brake pressure applied at a rear wheel in (Pa). The vehicle parameters are given by Table 3.5.1.

The equations of motion can be described as a matrix (state-space) form given by

$$\dot{\underline{x}} = \mathbf{A} \cdot \underline{x} + \mathbf{B} \cdot P_b + \mathbf{G} \cdot \delta_{fw} \quad (3.5.7)$$

with state vector

$$\underline{x} = [v \quad r \quad y_e \quad \psi_e \quad \kappa]^T \quad (3.5.8)$$

The linear time invariant state-space equation of the Kalman filter is given by [2]

$$\begin{aligned} \dot{\hat{\underline{x}}} &= \mathbf{A} \cdot \hat{\underline{x}} + \mathbf{B} \cdot P_b + \mathbf{G} \cdot \delta_{fw} + \mathbf{L}(\underline{y} - \mathbf{C}\hat{\underline{x}}) \\ &= (\mathbf{A} - \mathbf{LC}) \cdot \hat{\underline{x}} + \mathbf{B} \cdot P_b + \mathbf{G} \cdot \delta_{fw} + \mathbf{L}\underline{y} \end{aligned} \quad (3.5.9)$$

As can be seen from this equation, the Kalman filter is identical to the state-space model of the vehicle-roadway system except for the input $\mathbf{L}(\underline{y} - \mathbf{C}\hat{\underline{x}})$. Vector \underline{y} is a vector containing the lateral position of the lane markers (coming directly from the LMS) at various longitudinal distances in front of the vehicle, and $\mathbf{C}\hat{\underline{x}}$ are the estimated lateral positions at the same longitudinal distances. With the state-space description of the vehicle-roadway model as defined above, the output vector can be composed as

$$\underline{y} = \mathbf{C}\underline{x} \quad (3.5.10)$$

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 & -x_1 & \frac{1}{2}x_1^2 \\ | & | & | & | & | \\ 0 & 0 & 1 & -x_n & \frac{1}{2}x_n^2 \end{bmatrix}$$

where x_i are longitudinal coordinates in the vehicle frame (e.g., 5, 7.5, 10, 12.5, and 15 meter ahead of the vehicle). The dynamic behavior of the estimation error $(\underline{y} - \mathbf{C}\hat{\underline{x}})$ is prescribed by selecting a feedback matrix \mathbf{L} . Matrix \mathbf{L} may be designed to ensure that the eigenvalues of the closed-loop Kalman filter $\text{eig}(\mathbf{A} - \mathbf{LC})$ correspond to a rapid well behaved decay of any estimation error. The state estimation becomes faster, but also more sensitive to measurement noise. Thus the Kalman filter provides a compromise between the speed-of-state reconstruction and immunity to measurement noise. The balance between these two properties is determined by the covariance matrices \mathbf{Q}_f and \mathbf{R}_f in the design stage.

Matrix Q_f is related to the known inputs (front wheel steer/brake pressure) and matrix R_f to the noise coming from the lane marker sensor.

The design of the Kalman filter is fairly straightforward. After determination of the state-space matrices, the noise covariance matrices have to be chosen. A method of computing the Kalman filter feedback gain matrix L is required. The method chosen is that one due to Kalman and Bucy [1] in which it is assumed that the system to be observed is driven by white noise and that the observed signals are corrupted by white noise too. Feedback matrix L can be found by solving the algebraic Riccati equation. The only problem with the above vehicle-roadway model is that this model is not directly driven by white process noise. Furthermore the model is not observable because the curvature is modeled as an integrator without input. To overcome the white noise input and observability requirement a slight alteration of the model in the Kalman filter design phase has been made. First it is assumed that the model is driven by white noise steer and brake pressure inputs. Furthermore a third white noise source has been included representing the input for the curvature state. This means that the curvature is expected to behave as integrated white noise. A new process noise input vector H and white noise source w can now be determined

$$H = \begin{bmatrix} \frac{2C_{F\alpha 1}}{m_v} & 0 & 0 \\ \frac{2C_{F\alpha 1}a_v}{I_{zv}} & \frac{T_{w2}K_{b2}}{2I_{zv}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad w = \begin{bmatrix} w_{\delta_{fw}} \\ w_{p_b} \\ w_k \end{bmatrix} \quad (3.5.11)$$

The Kalman filter design should be based on matrices A , H , C , Q_f and R_f . After some trial and error the following covariance matrices resulted in a good Kalman filter performance.

$$Q_f = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{4C_{Fa1}a_v}{T_{w2}} & 0 \\ 0 & 0 & 10 \end{bmatrix} \quad (3.5.12)$$

$$R_f = \begin{bmatrix} 5.0e+4 & 0 & 0 & 0 & 0 \\ 0 & 7.5e+4 & 0 & 0 & 0 \\ 0 & 0 & 1.0e+5 & 0 & 0 \\ 0 & 0 & 0 & 1.25e+5 & 0 \\ 0 & 0 & 0 & 0 & 1.5e+5 \end{bmatrix} \quad (3.5.13)$$

Note, the elements of the white measurement noise covariance matrix R_f are proportional to the longitudinal spacing of the lane marker data (5, 7.5, 10, 12.5 and 15 meter ahead of the vehicle). Matrix L (5x5) can now determined by solving the matrix Riccati equation using a routine such as the LQE function in the commercial package *Matlab*. Figure 3.5.4 - 3.5.7 illustrate the performance of the Kalman filter when it is implemented in the CAPC

simulation tool. Random (low frequency) steering has been applied while the vehicle was traveling 108 km/h on a straight uneven road. The vehicle model used was the 14 DOF model with a non-linear transient tire model (Magic Formula).

From equations (3.5.2) - (3.5.6) it can be seen that the differential equations of the vehicle model as well as the roadway geometry model are vehicle speed (U) dependent. This implies that the time invariant Kalman filter as discussed above is valid for only one vehicle speed because matrix A is not updated for the speed of travel. Although a Kalman filter is normally robust for certain parameter changes, it is not expected that it will operate satisfactorily when designed for one vehicle speed and operated at another speed that is significantly different from the design speed. Therefore the equations of motion of the roadway as well as the vehicle model are updated based on the speed of travel.

Two options are available. One might consider designing a time-variant Kalman filter which implies that the feedback gain matrix L has to be determined on-line. The other option is to calculate several feedback matrices off-line -- each valid for one specific vehicle speed. The latter method implies that the feedback gains are determined by a table look-up or polynomial fit. To reduce the computational burden the latter option (gain scheduling) has been chosen. The gains of Kalman filter feedback matrix L have been determined for speeds from 5 to 55 m/s and each gain has been fitted by a 3rd order polynomial fit as a function of the vehicle speed. The coefficients of the fit are then transferred to the CAPC control code. It has been chosen to update the feedback gain matrix L every time the speed of the vehicle changes by five percent. The equations of motion of the vehicle and roadway geometry model are updated every time step for changes of the speed of travel. Figure 3.5.8 show the change of the five state-feedback gains as a function of the vehicle speed.

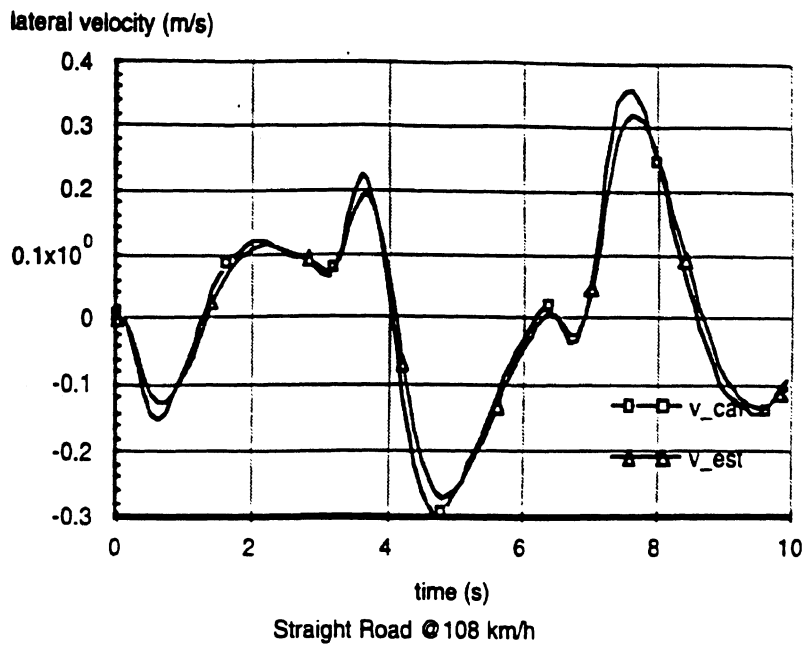


Figure 3.5.4 Estimation of the vehicle side-slip velocity.

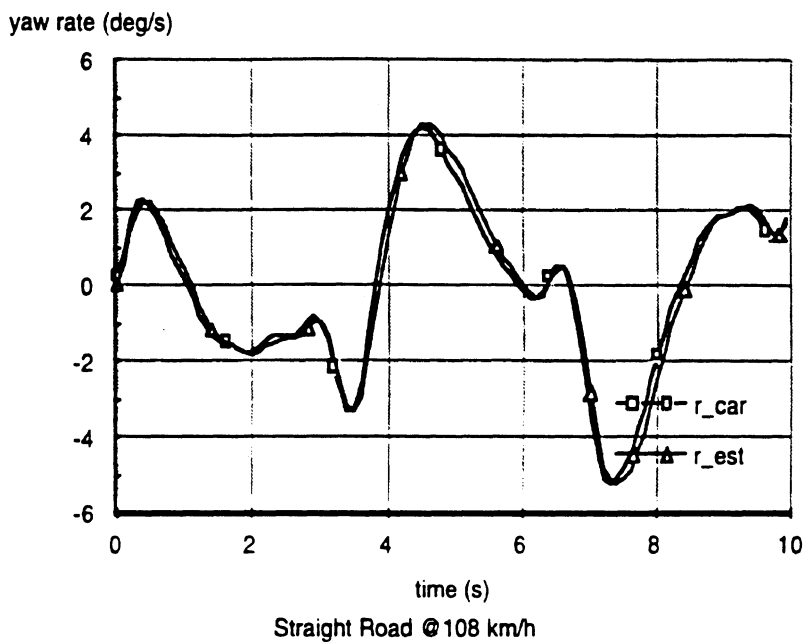


Figure 3.5.5 Estimation of the vehicle yaw rate.

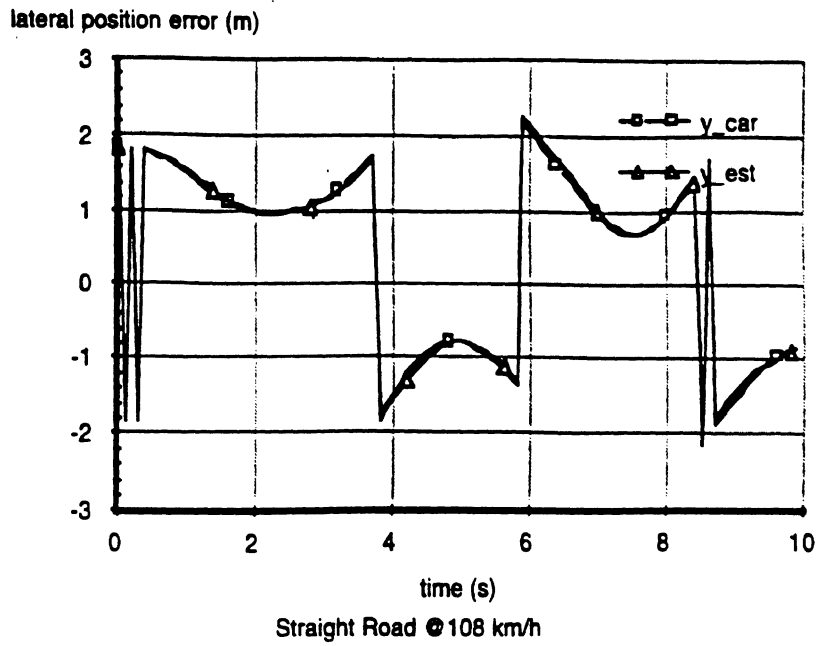


Figure 3.5.6 Estimation of the lateral vehicle position.

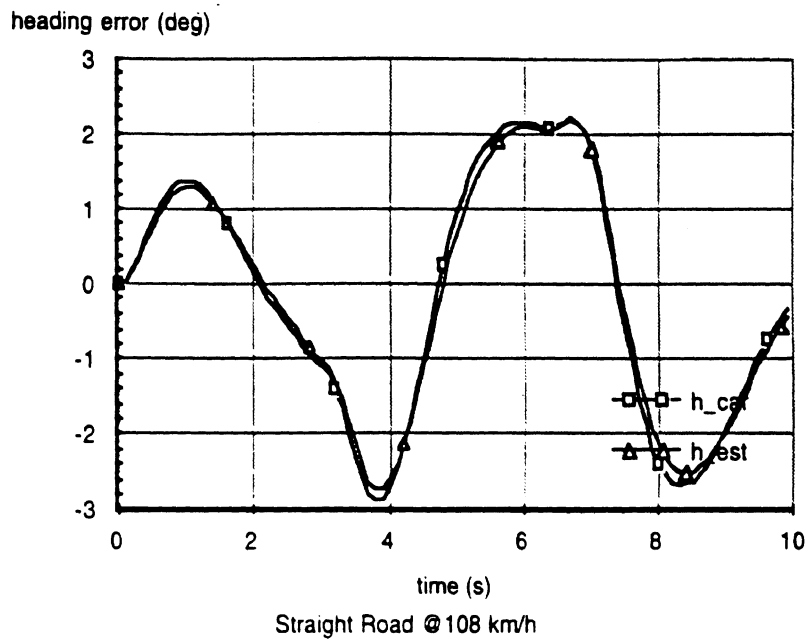


Figure 3.5.7 Estimation of the vehicle heading angle.

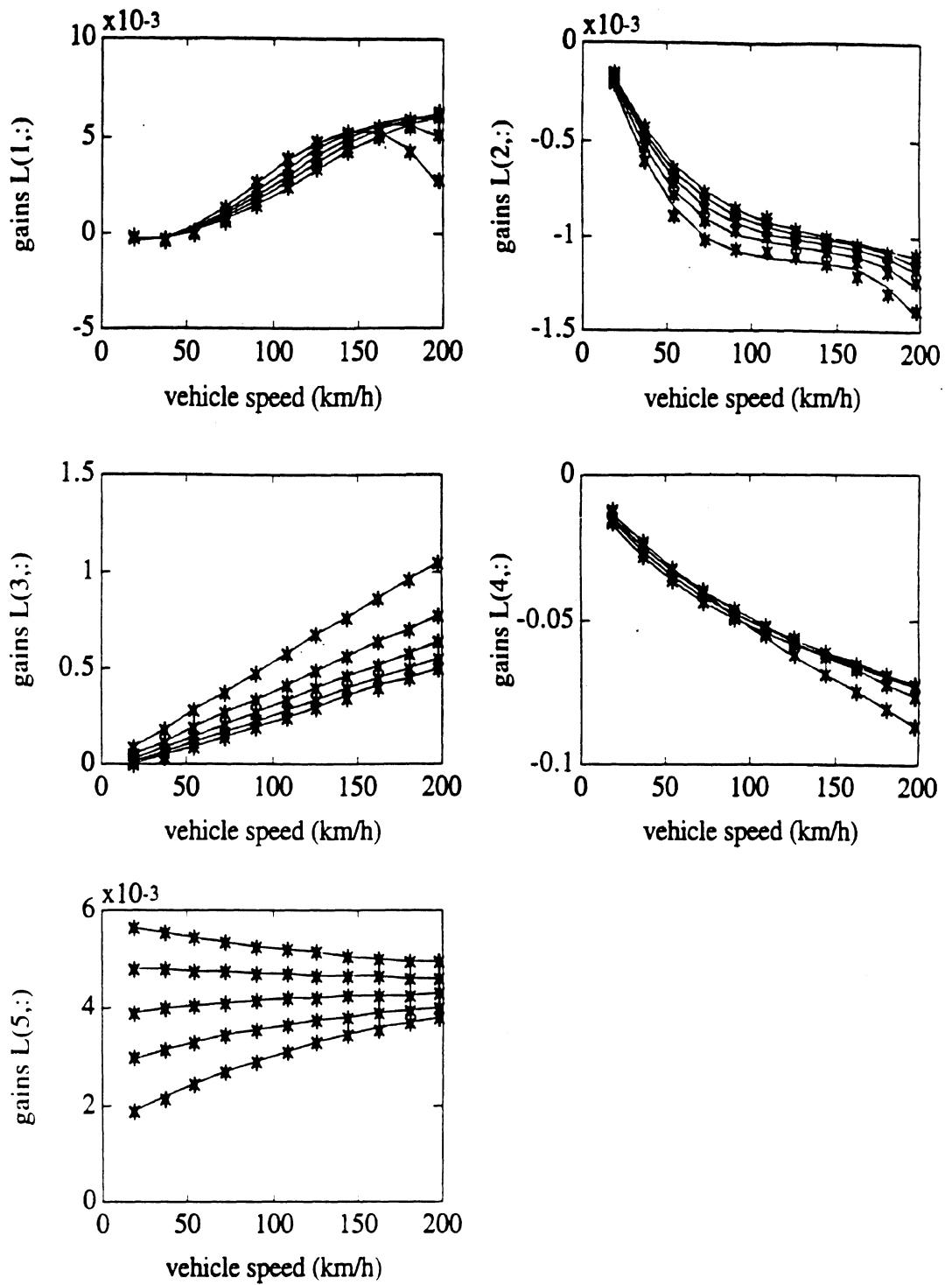


Figure 3.5.8 Kalman filter feedback gains as a function of vehicle speed.

Figure 3.5.9 illustrates the final lay-out of the Kalman filter. It has 3 inputs: the front wheel steer angle of the vehicle, the applied brake pressure on a rear wheel and the lane marker data from the LMS. The front wheel steer angle and lane marker coordinates can be measured directly. The applied brake pressure comes from the brake controller as will be discussed in the next section. A first-order system with time constant τ has been augmented to the command brake pressure input. It describes the dynamics of the brake system and improves the state estimation when brake-steer control is applied.

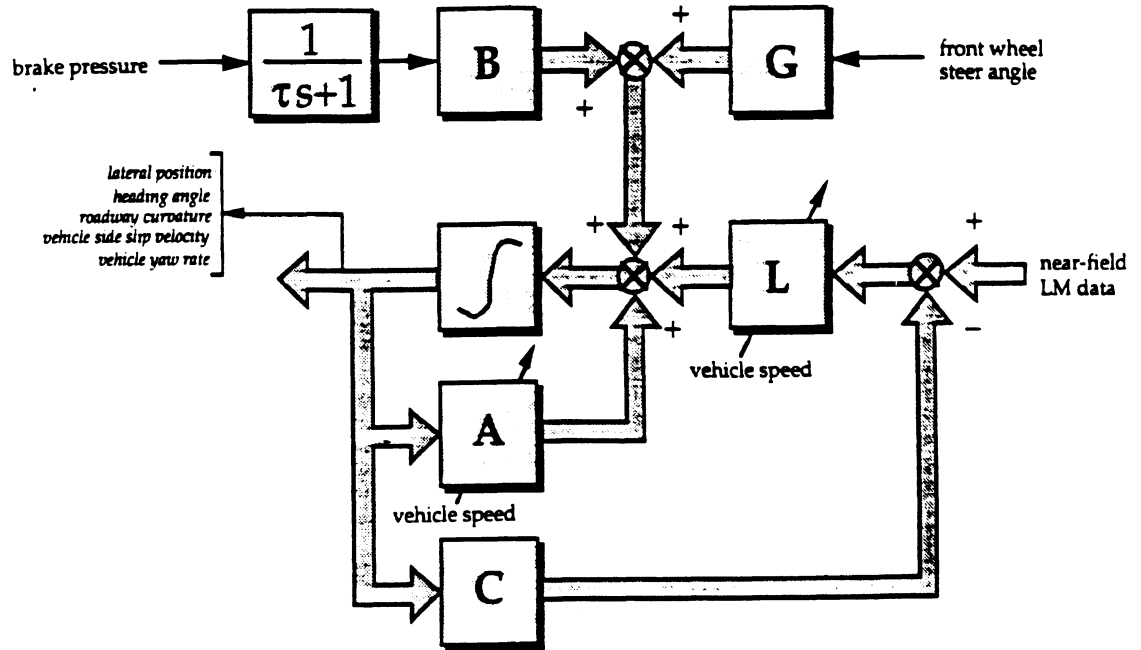


Figure 3.5.9 Kalman filter lay-out.

3.5.3 Brake-Steer Controller

The brake-steer controller is based on an LQ (linear quadratic) state feedback [2] plus a feed-forward design. The states estimated with the near-range Kalman filter described above are now used to close the loop to provide an active intervention function. The intention is to steer the vehicle by applying the brakes of a single rear wheel.

The front wheel steer input can be regarded as a disturbance while closing the loop during a brake-steer intervention. Depending on the direction of the steer angle it might help or counteract brake-steer control. Since the front wheel steer angle is measured directly it can be rejected by a feed-forward controller. The magnitude of the front-wheel steer disturbance depends on the roadway geometry. On a straight road without wind or road-surface disturbances, any steer angle unequal to zero must be rejected. While driving through a curve the front-wheel steer angle depends on the radius of the curve and the vehicle's handling characteristics (under/oversteer). Since the LMS in combination with the above described Kalman filter is able to determine the curvature of the roadway it is possible to estimate the kinematically required front-wheel steer angle δ_{kin} if the under/oversteer behavior (and thus vehicle speed) is known. From equation (3.5.6) the rear-wheel brake pressure required to reject a yaw motion can be determined quite easily and is given by

$$P_b = -\left(\frac{4C_{F\alpha 1}a_v}{T_{w2}K_{b2}}\right) \cdot (\delta_{fw} - \delta_{kin}) \quad (3.5.14)$$

The kinematically required front-wheel steer angle as a function of the vehicle speed U and estimated roadway curvature $\hat{\kappa}$ can be determined from the steady-state solution of equations (3.5.5) and (3.5.6)

$$\delta_{kin} = \left((a_v + b_v) - \frac{(C_{F\alpha 1}a_v - C_{F\alpha 2}b_v)m_v U^2}{2C_{F\alpha 1}C_{F\alpha 2}(a_v + b_v)} \right) \hat{\kappa} \quad (3.5.15)$$

The first part of equation (3.5.15) describes the relation between the wheelbase of the vehicle and the radius of a curve ($1/\hat{\kappa}$) and the second part contains the correction based on the understeer coefficient for the steady-state handling characteristics of the vehicle.

The starting-point for the control-system design based on LQ control is the state-space representation of the vehicle-roadway model given by

$$\underline{\dot{x}} = \mathbf{A} \cdot \underline{x} + \mathbf{B} \cdot P_b + \mathbf{G} \cdot \delta_{fw} \quad (3.5.16)$$

with

$$\underline{x} = \left[v \quad r \quad y_e \quad \Psi_e \quad \int y_e \right]^T$$

As can be seen, the curvature has been eliminated from the roadway model because this state is uncontrollable (meaning that the brake pressure cannot influence the curvature κ of the roadway geometry). However, it does affect the relative lateral position y_e and heading angle ψ_e . Therefore the roadway geometry model can be simplified to

$$\dot{y}_e = -v - U\psi_e \quad (3.5.17)$$

$$\dot{\psi}_e = -r \quad (3.5.18)$$

The integral of the lateral position has been added in order to eliminate static position offsets due to unknown disturbances. The vehicle dynamics equations remain unchanged from equations (3.5.5) and (3.5.6).

The object of LQ control is to specify an input brake pressure P_b which steers the vehicle to a specified target state in such a way that, during the process, a defined quadratic cost function J is minimized. Minimization of the performance criterion yields an optimal feedback law compromising control effort (brake pressure) and control quality.

The aim of the brake-steer controller is to redirect the vehicle such that it remains on the road if the driver fails to take over control. The parameters to assess the quality of the controller are the lateral position, with respect to the road edge, the heading angle, and the integral of the lateral position error. Furthermore the applied brake pressure is limited and should be included in the performance index (otherwise LQ control will fail). Therefore the following performance index J will be used

$$J = \int_0^{\infty} \left(q_y y_e^2 + q_\psi \psi_e^2 + q_i \left(\int y_e \right)^2 + q_p P_b^2 \right) dt \quad (3.5.19)$$

with q_y , q_ψ , q_i , and q_p being weighting factors. Many different control laws can be derived by changing the weighting constants. It is up to the user to make a proper selection of the factors.

With the performance index and state-space equations available it is now quite straightforward to design the feedback controller that minimizes J at given weighting factor. The brake pressure P_b is a linear combination of the states by means of state feedback matrix K according to

$$P_b = -K \underline{x} - \left(\frac{4C_{F\alpha 1} a_v}{T_{w2} K_{b2}} \right) \cdot (\delta_{fw} - \delta_{kin}) \quad (3.5.20)$$

with

$$K = \begin{bmatrix} k_v & k_r & k_y & k_\psi & k_{\int y} \end{bmatrix} \quad (3.5.21)$$

Note that the feed-forward part is not related to the state-feedback design and can therefore just be added to the total brake pressure. Matrix K can be determined by solving the matrix Riccati equation (LQR function in Matlab).

Depending on the choice of the weighting factors the response might be too sluggish, have too much overshoot, or exhibit a static offset. The control system designer may tune the closed-loop behavior by adjusting the weighting constant. For example if the response is too sluggish it means that the contribution of the y_e part in quadratic index J is too small. The response can be made faster by increasing weighting factor q_y and recalculating the feedback gains.

The closed-loop performance has been determined by a simulated step position error input of 1 meter and a front wheel step-steer input of 0.5 degree while the vehicle was driving on a straight road. The first simulated condition characterizes the quality of the feedback part, while the second tests the quality of the feed-forward part in combination with the feedback. Figure 3.5.10 and 3.5.11 show the results for different values of the weighting factors.

All simulations were carried out with the 14 DOF vehicle model with non linear tires based on the Magic Formulae (combined slip). As can be seen from the figures, a trade-off exists between a fast recovery of the lateral position of the vehicle without too much overshoot and the rejection of disturbances. The more integral action the feedback structure has (large value of q_i), the larger the overshoot will be. However, more integral action results in a better disturbance rejection. The following combination of weighting factors gave a reasonable good closed-loop performance:

$$q_y = 20, q_\psi = 0, q_i = 5, q_p = 1e-12$$

The integral action is necessary in order to be able to cope with model errors, side wind, road superelevations, and faulty driver inputs.

The real brake-steer controller in the prototype CAPC vehicle is subjected to time delays:

- 100 ms from the lane marker imaging system (10 Hz)
- 100 ms from the brake-steer controller (10 Hz)

These time delays might affect the closed-loop performance significantly depending on the magnitude (and thus closed-loop eigenvalues) of the feedback gains. Figure 3.5.12 and 3.5.13 illustrate the effect of 100 and 200 ms time delay between brake-steer controller input (estimated states from the Kalman filter) and output (brake pressure).

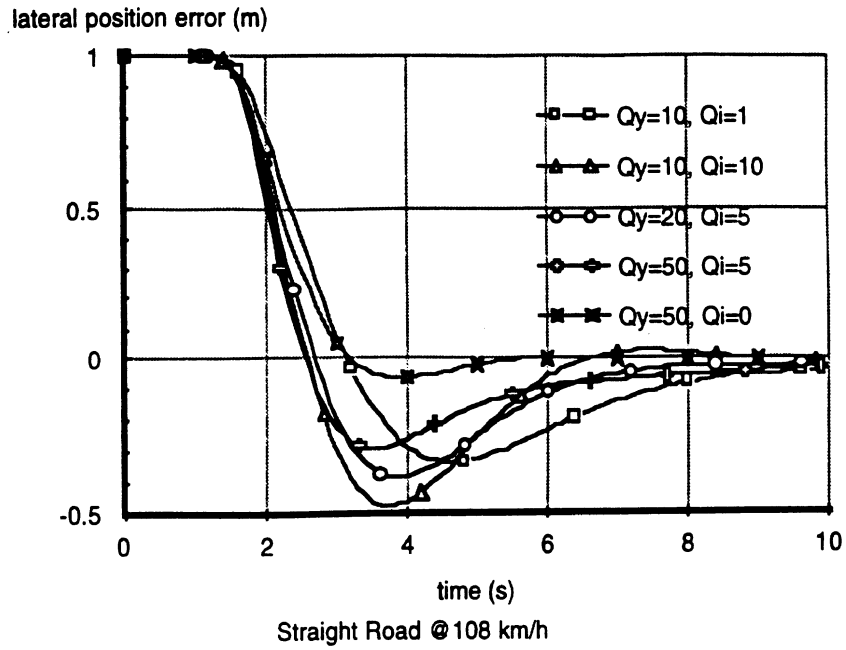


Figure 3.5.10 Position step input of 1 meter at $t = 1$ sec

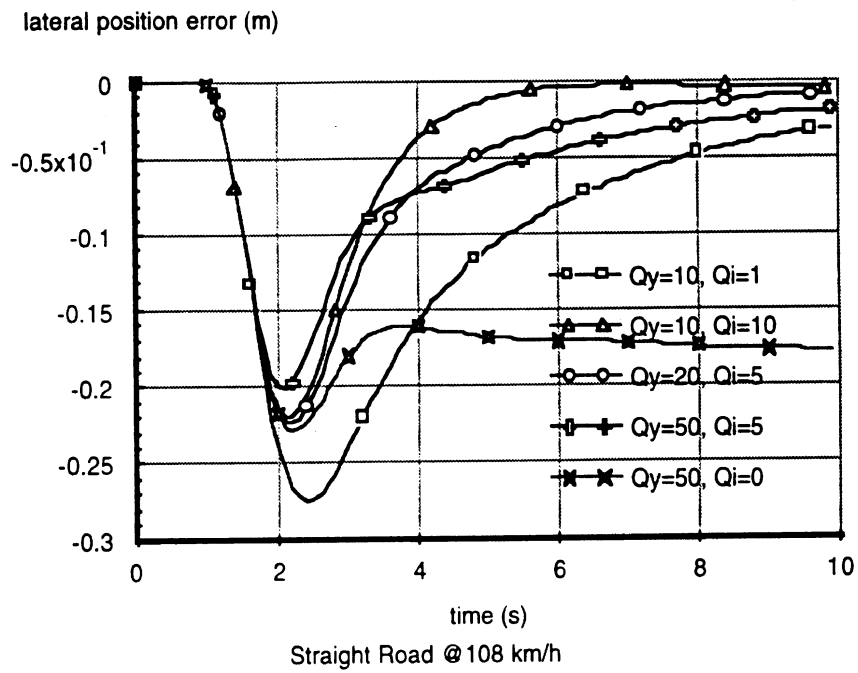


Figure 3.5.11 Steer step input of 0.5 deg at $t = 1$ sec.

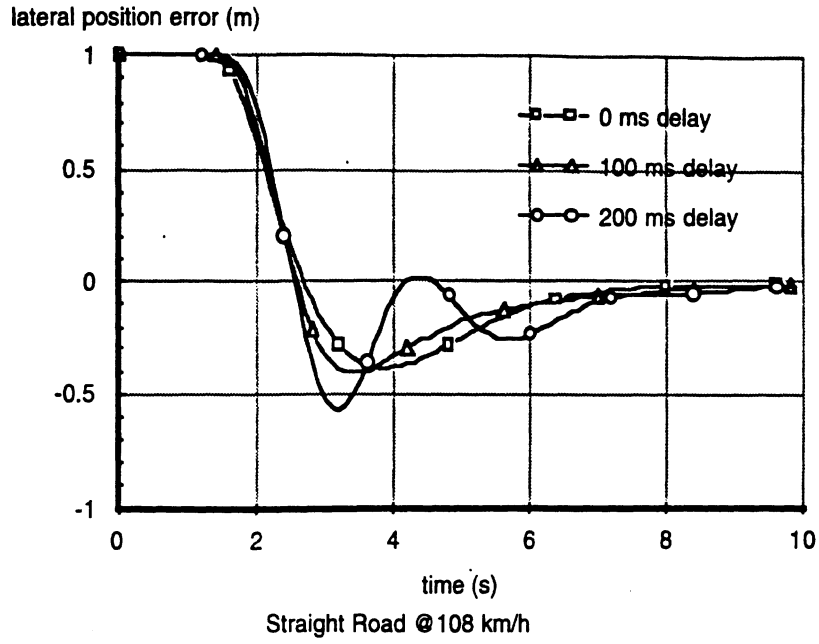


Figure 3.5.12 Position step input of 1 meter at $t = 1$ sec.

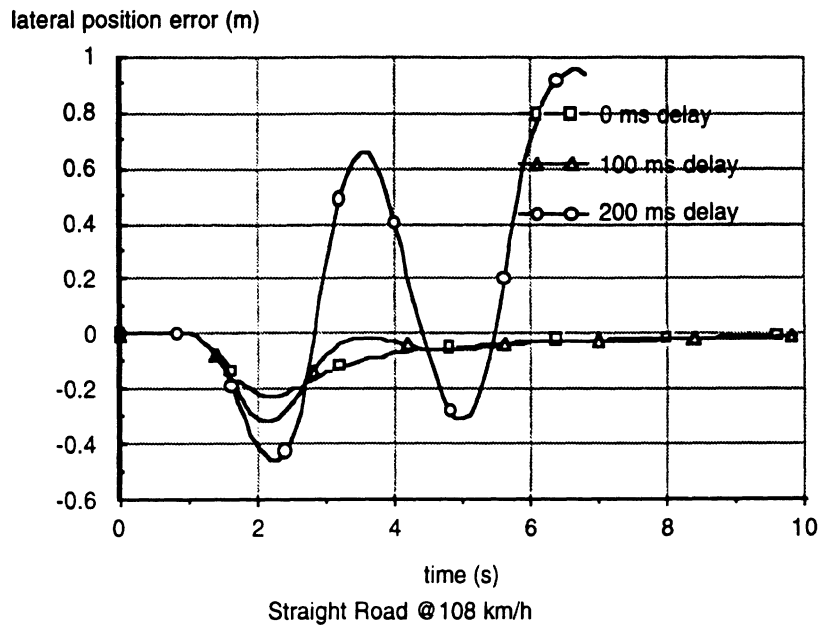


Figure 3.5.13 Steer step input of 0.5 deg at $t = 1$ sec.

As can be seen from the figures, time delays are fatal for a good brake-steer controller design. Less resonant behavior can be obtained with more conservative LQ gains, but performance degrades significantly too. Therefore it seems to be necessary to predictor the vehicle/roadway states 200 ms ahead such that they match the applied brake pressure.

The simplest way to predict these states t_{del} sec ahead is to use the derivatives from the Kalman filter states. The new predicted states that should be used for the brake-steer controller are then given by

$$\begin{aligned} \hat{y}_e^* &= \hat{y}_e + \hat{\dot{y}}_e \cdot t_{del} \\ \int \hat{y}_e^* &= \int \hat{y}_e + \hat{y}_e \cdot t_{del} \\ \hat{v}_e^* &= \hat{v}_e + \hat{\dot{v}}_e \cdot t_{del} \\ \hat{r}_e^* &= \hat{r}_e + \hat{\dot{r}}_e \cdot t_{del} \end{aligned} \tag{3.5.22}$$

Figure 3.5.14 and 3.5.15 show the results of including a prediction step between the Kalman filter output and the brake-steer controller input. The responses of the system with prediction are much better damped and the disturbance rejection response is now stable. Without the prediction step the system will get unstable in case of a disturbance due to the 200 ms time delay.

As with the Kalman filter, the feedback gains of the brake-steer controller are vehicle speed dependent. Figure 3.5.16 presents the gains as a function of speed of travel. The gains have been calculated in the range from 5 to 55 m/s and a 3rd order polynomial fit has been used in the simulation code to determine the gains.

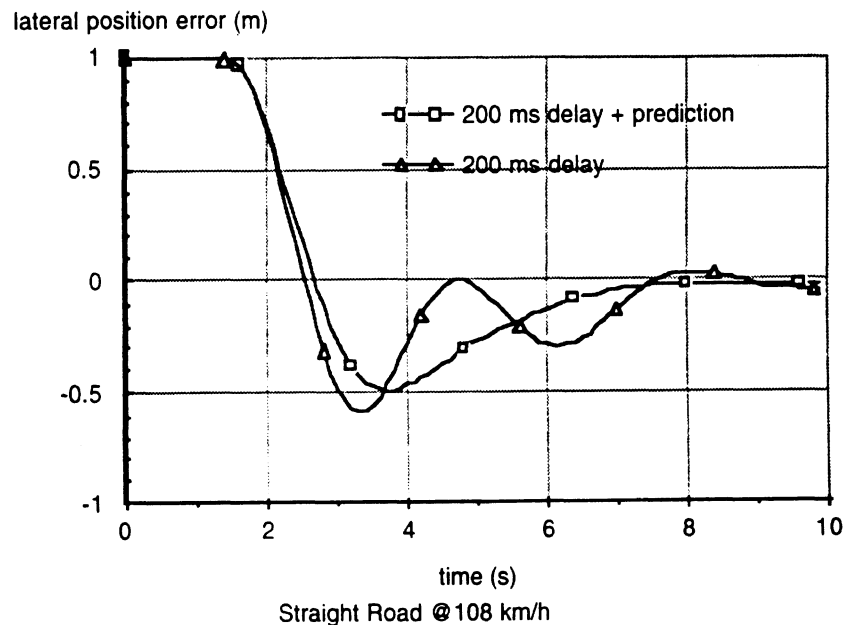


Figure 3.5.14 Position step input of 1 meter at $t = 1$ sec

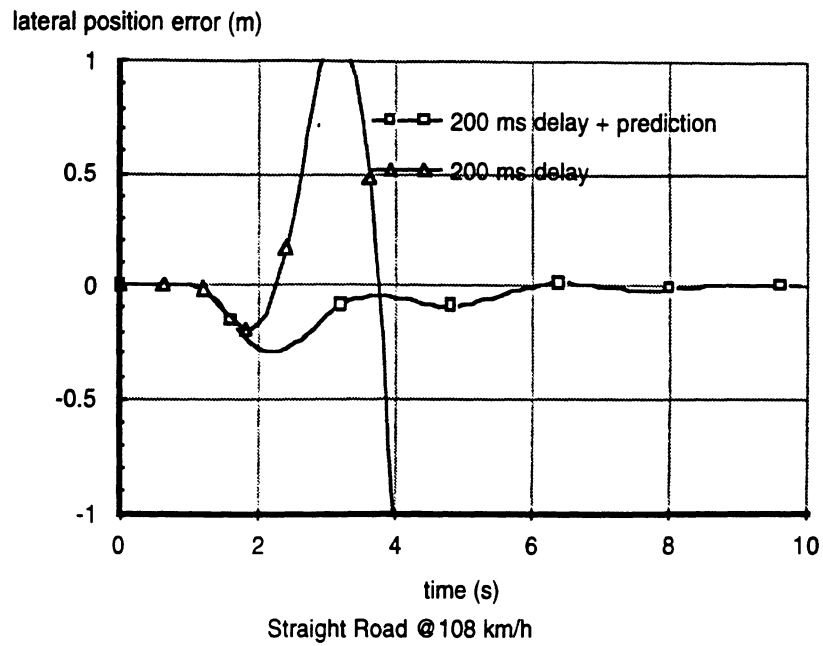


Figure 3.5.15 Steer step input of 0.5 deg at $t = 1$ sec.

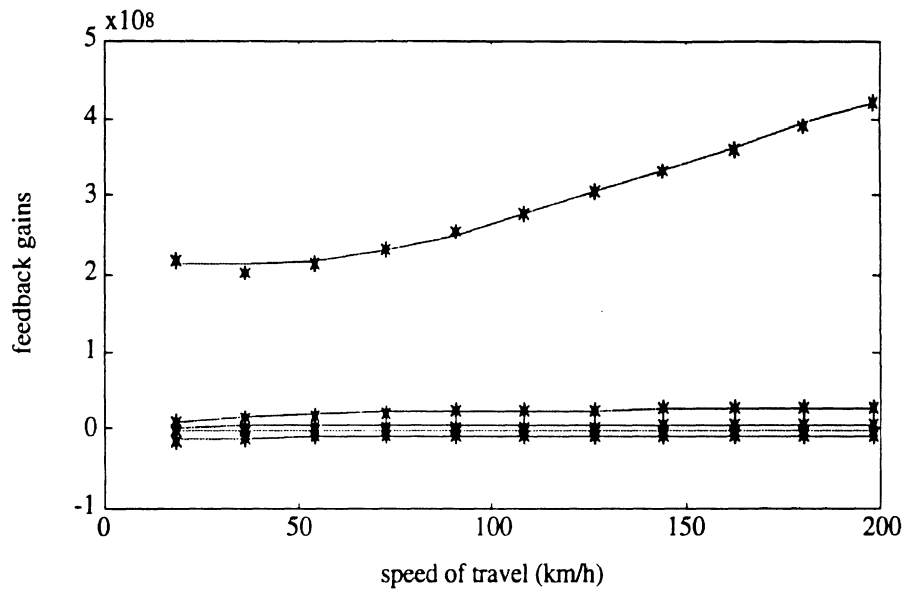


Figure 3.5.16 Brake-steer feedback gains as a function of vehicle speed.

Finally Figure 3.5.17 shows a schematic of the entire brake-steer controller. Both feed-forward and feed-back loops are clearly identifiable.

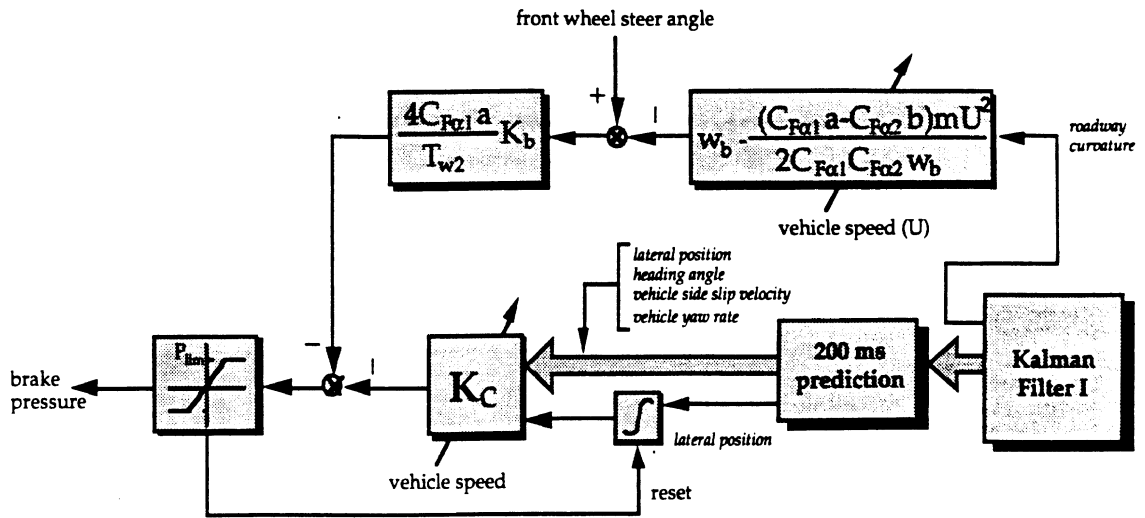


Figure 3.5.17 Schematic of the brake-steer controller.

3.6 Design of the Decision Module

The decision module has the task of determining when to initiate a warning and/or intervention response. Because the differential braking intervention controller is not yet implemented on the CAPC prototype vehicle, there is currently only an audible buzzer to indicate when an intervention would occur, if the differential braking system were completed. The CAPC simulation tool does include the active intervention feature. The design of warning and intervention logic in both the vehicle and the simulation have been based on the presence of closed loop intervention. The decision rules for activating both the warning and intervention buzzers in the vehicle is done based both on extensive simulation work and on field experience with the CAPC vehicle. Here we describe the decision rules, the considerations made and work performed to arrive at these rules, and the types of road departure scenarios in which the system is expected to function well.

The concept of the CAPC system includes making decisions based on both the time-to-lane crossing (TLC) and on the driver state. However, work on the driver-state assessment (Section 3.4) has not proceeded sufficiently to include a driver-state-assessment module onboard, and so decisions are based only on TLC. The use of both metrics is desirable because the TLC is a scalar measure of the vehicle-roadway tracking margin assuming that steering inputs will not change; the driver state indicates the alertness of the driver which, in turn, can be considered as the likelihood that the steering input will change to improve lane keeping.

TLC is updated at 10 Hz, and the decision to warn, intervene, or do nothing is also updated at 10 Hz, immediately following the computation of TLC. In essence, decisions to warn or intervene are made by comparing the TLC to two threshold values, TLC_w and TLC_l :

If $TLC > TLC_w$, then do nothing.
If $TLC \leq TLC_w$, then warn.
If $TLC \leq TLC_l$, then intervene and issue a warning.

The values of the two parameters used in the prototype are $TLC_w = 2.0$ sec, and $TLC_l = 1.0$ sec. These values were chosen by a combination of simulation and experience in the field using the prototype vehicle. Simulation indicates that this set of thresholds will be effective for mild and moderate road departure scenarios. Limitation in the moderate authority of differential braking, of course, means that there will be severe departure scenarios for which even active intervention cannot prevent road departure. Empirical field experience with several drivers indicates that the system provides a good "feel" -- i.e., warnings and intervention indicators are issued at reasonable times. The rest of this section describes how these values were determined, as well as insights discovered.

There is tight coupling between the decision logic parameters and the high-level system goals. We chose these parameter values because we wish to meet the following system goals, listed in order of precedence:

1. Warn only when the driver cannot perform normal lane-keeping with normal driving authority.
2. Intervene only when:
 - (a) the driver cannot keep the vehicle on the roadway even using an aggressive level of steering authority
 - (b) the driver has had time to react to a warning
3. Allow driver ultimate authority of lateral motion during intervention.
4. Once started, a successful intervention is one in which:
 - (a) intervention does not begin before one wheel leaves the lane
 - (b) intervention (assuming no driver reaction) keeps one wheel on the pavement of the travel lane at all times.

These goals determine the feel of the system. For example, notice that the system is not designed to keep the vehicle on the road at all costs. To do that results in many false alarms, since the TLC thresholds for intervention would need to be quite large to handle severe road departure scenarios. Instead, the approach taken is to provide a backup to the driver which allows the driver as much leeway as possible. This minimizes false alarms and driver annoyance while still providing effective safety assistance for departure scenarios which would be induced by drowsiness or inattention.

Note that the placement of goal 2(a) above goal 2(b) places higher priority on timely intervention than on allowing the driver enough time to react to a warning before the active safety system intervenes. We note, also, that the use of brake-steer for active intervention is assumed to satisfy the third goal. Certainly other sets of goals can be selected, and other goals would simply result in different thresholds.

Simulations were used to choose initial values for the TLC thresholds: $TLC_w = 1.4$ sec, $TLC_l = 0.8$ sec. These were later adjusted in the field to $TLC_w = 1.8$ sec to allow the driver more time to react, and finally, $TLC_w = 2.0$ and $TLC_l = 1.0$ sec after additional drivers reported their preferences.

Additional decision logic is included to improve robustness: (1) The TLC must fall below the threshold for three consecutive samples before any new action is taken, and (2) no action is begun unless there are four points currently being reported by the LMS on the right side of the lane. The full logic used is shown in Table 3.6.1.

We briefly describe how the initial simulation-based TLC thresholds were chosen. Numerical simulations are run to choose TLC thresholds that meet the desired criteria over the largest range of initial conditions. Simulations include two basic steering scenarios conducted in three different road curvature settings. Steering includes both a so-called pulse steer input and a step steer input, both applied at the steering wheel. The pulse input

is used to generate a motion in which the vehicle travels along its longitudinal axis with zero steer angle and no curvature in its path. This represents a gentle drift off the road. The step steer represents a driver who has, for instance, fallen asleep, and the vehicle travels with curvature in its path (relative to the ideal path). Departures are simulated in three roadway scenarios: in a straight-away; in a constant radius curve; in a Euler spiral transition from constant curvature to (or from) a straight-away. Speeds of 40 to 120 kph (25 to 75 mph) are used. We assume that the driver either reacts appropriately -- as a step input of the steering wheel -- or the driver does not react, and the steering wheel is kept fixed.

Figure 3.6.1 shows the results of a simulation in which the vehicle is traveling at 90 kph down a straight away when a step input of 4 deg is applied at the steering wheel (about 0.25 deg at the front wheel). The vehicle curves slightly toward the right edge of the road, as shown in the second subplot of the figure. Note that the CG distance is measured with respect to either the left or right side lane, depending on the vehicle direction and proximity -- hence the sudden jumps in the subplot. Soon the TLC falls below the warning threshold. In this simulation the "driver" takes no action despite the warning, and soon an intervention occurs. The brake-steer system brings the CG back to the right-side lane marker, as desired, overcoming both the off-road heading angle and the original 4 deg offset in steering. The plot shows that the CG travels up to about 0.7m past the edge, so that the left wheels barely remain on the lane. This simulation is considered successful because intervention does not begin until a wheel has left the lane, and because at least one wheel remains on the lane. We note that the vehicle is traveling toward the edge at about 1.4 m/s when the intervention begins, which is a rather steep heading angle (about 3 degree).

A second simulation is run in which the same initial step input of steer is applied, but now the driver "reacts" to the warning. This is modeled with a step steer applied 0.8 sec after the warning reaches the driver. (The simulation program contains the system delays that are part of the prototype computing environment.) This corrective steer is 20 deg at the steering wheel -- corresponding to about 0.25G at 90 kph. Figure 3.6.2 presents the results of this simulation, which shows that the driver's action is sufficient both to avoid triggering the intervention and to keep the vehicle in the lane.

Finally, a third simulation is run in which, again, the step input is applied. This time, however, the vehicle's curve toward the right edge is taken to be part of the driver's normal lane-keeping behavior, and a corrective steer of 10 deg at the wheel (about 0.13 G at 90 kph) is applied just before the warning would go off. This simulation's results are shown in Figure 3.6.3. The vehicle CG is still in the middle of the lane when the warning is about to sound -- but the steering indicates that if held constant, the vehicle CG will pass over the road edge in two seconds at a speed of about 1.6 m/s, which is a moderately severe departure.

Extensive simulation work was performed to choose the TLC thresholds, and to evaluate the performance that might be expected. The decision rules and parameters chosen in simulation were found to provide a good warning and intervention functionality which was

expected to have a good feel, by designing specifically to avoid false alarms in the large majority of typical driving situations, and by allowing time for the driver to react to the warning. Experience in the vehicle itself confirmed this -- the thresholds were changed only slightly.

We find that the important variables in the success of the warning and intervention decision system are vehicle speed; angle of vehicle departure; difference in curvature between the roadway and the vehicle's path; front wheel steer angle; and of course, the authority and quality of the intervention actuation system.

The system, of course, has a finite set of situations in which both the system function and the system feel are delivered successfully. Clearly for very severe departure situations, the TLC will trigger warnings and interventions while the vehicle is still well within the lane. Additionally, the current system configuration, including the limited authority of brake-steer intervention, cannot prevent the vehicle from completely leaving the lane if departure angles are greater than approximately 3.5 deg (at 90 kph) in straight aways, and about 2.0 deg in sharp curves (radius of curvature 400m). There are several ways to expand the set of recoverable scenarios, including braking with the front wheels, improving the brake-steer controller design, and using additional variables in the decision logic (such as predicted angle of departure). The last option would allow us to sidestep the natural tradeoff in TLC threshold design between (a) avoiding false alarms and early warnings in mild scenarios and (b) providing adequate warning and time for intervention in more severe scenarios. A summary of the limits of performance of the system with the decision rule and parameters described here is presented in Table 3.6.2.

Warning on if:

estimated $TLC \leq \overline{TLC}_w$ for [3] consecutive samples

and

warning has been on for less than [10] seconds

and

a new warning is not begun unless the previous warning has been off for at least [1] second, and unless the LMS has reported at least [4] points on the right side of the lane.

or if:

intervention is active

Intervention begins when:

estimated $TLC \leq \overline{TLC}_i$ for [3] consecutive samples **

and

a new intervention is not begun unless the previous intervention has been off for at least [1] second.

Intervention and warning continue until: TLC rises above the respective thresholds, or until the action has continued for [10] seconds.

No warning or interventions allowed when: vehicle speed is below [30 kph] or above [120 kph].

Threshold values:

$$\overline{TLC}_w = 2.0 \text{ sec}$$

$$\overline{TLC}_i = 1.0 \text{ sec}$$

Table 3.6.1 Warning and Intervention Rules in the CAPC Prototype

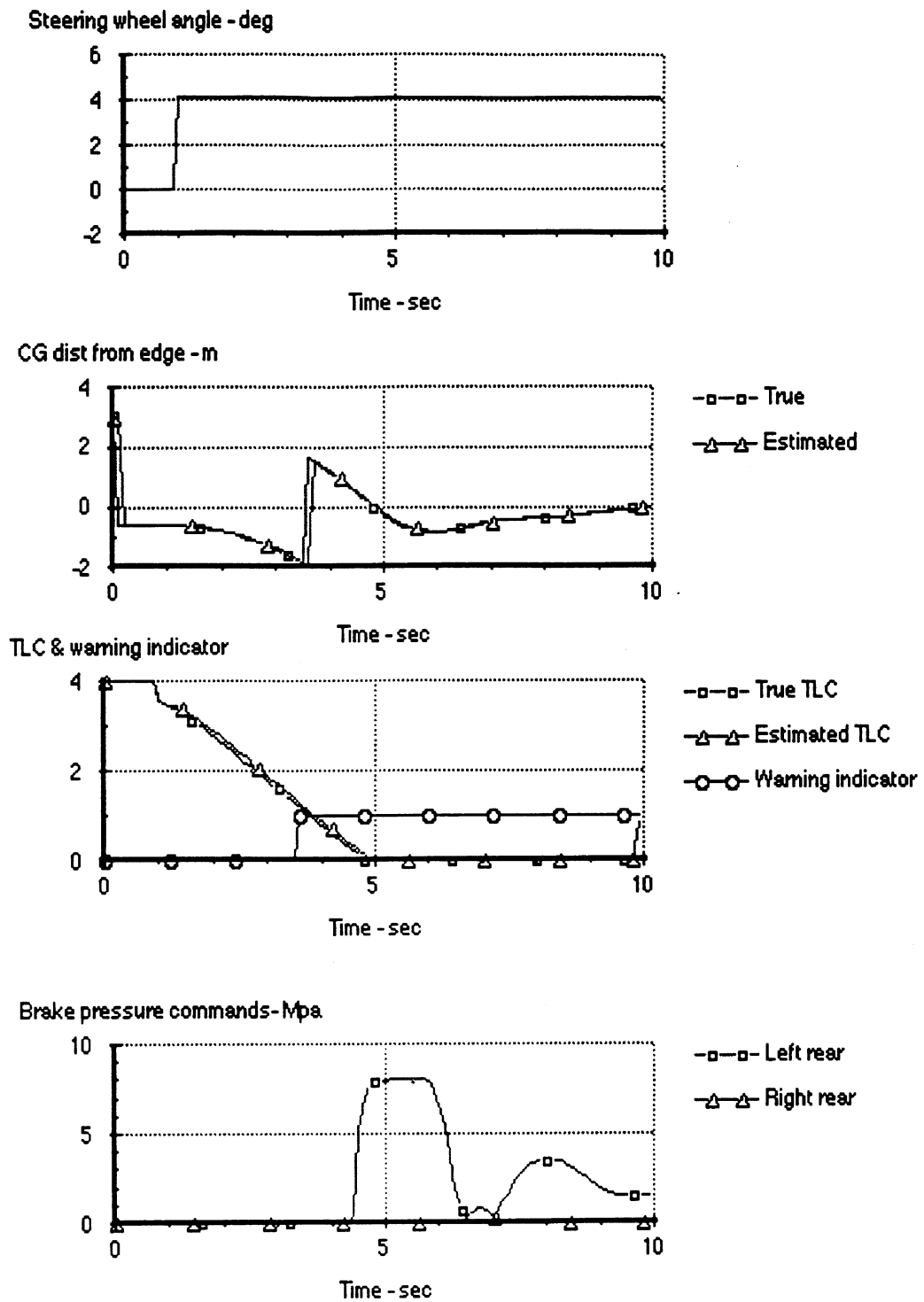


Figure 3.6.1:
Simulation: Warning and intervention with no corrective steering by driver
(90 kph on straight road with Step steer input of 4 deg at steering wheel)

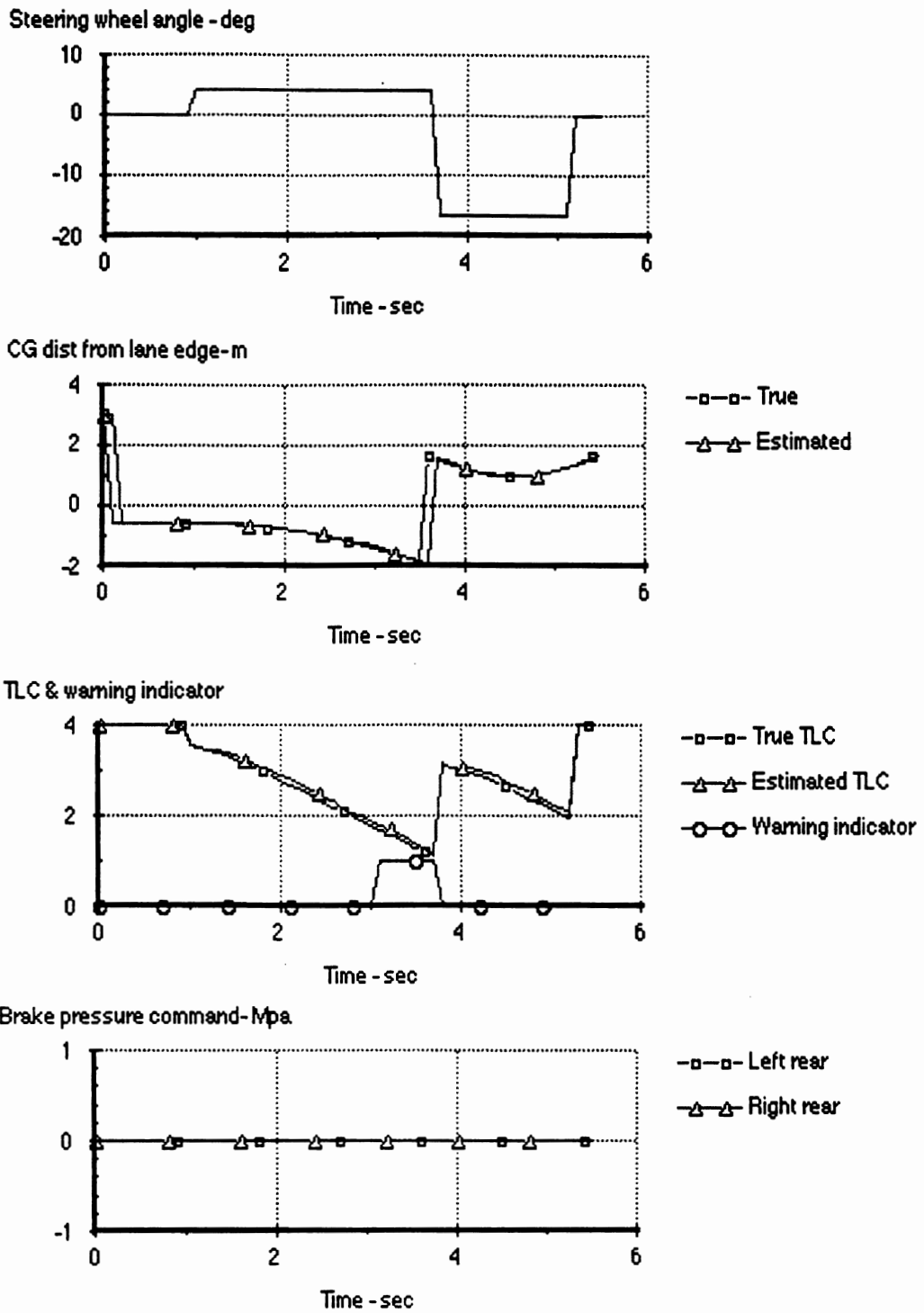


Figure 3.6.2:
Simulation: Driver reacts to a warning (0.25G), and intervention is avoided

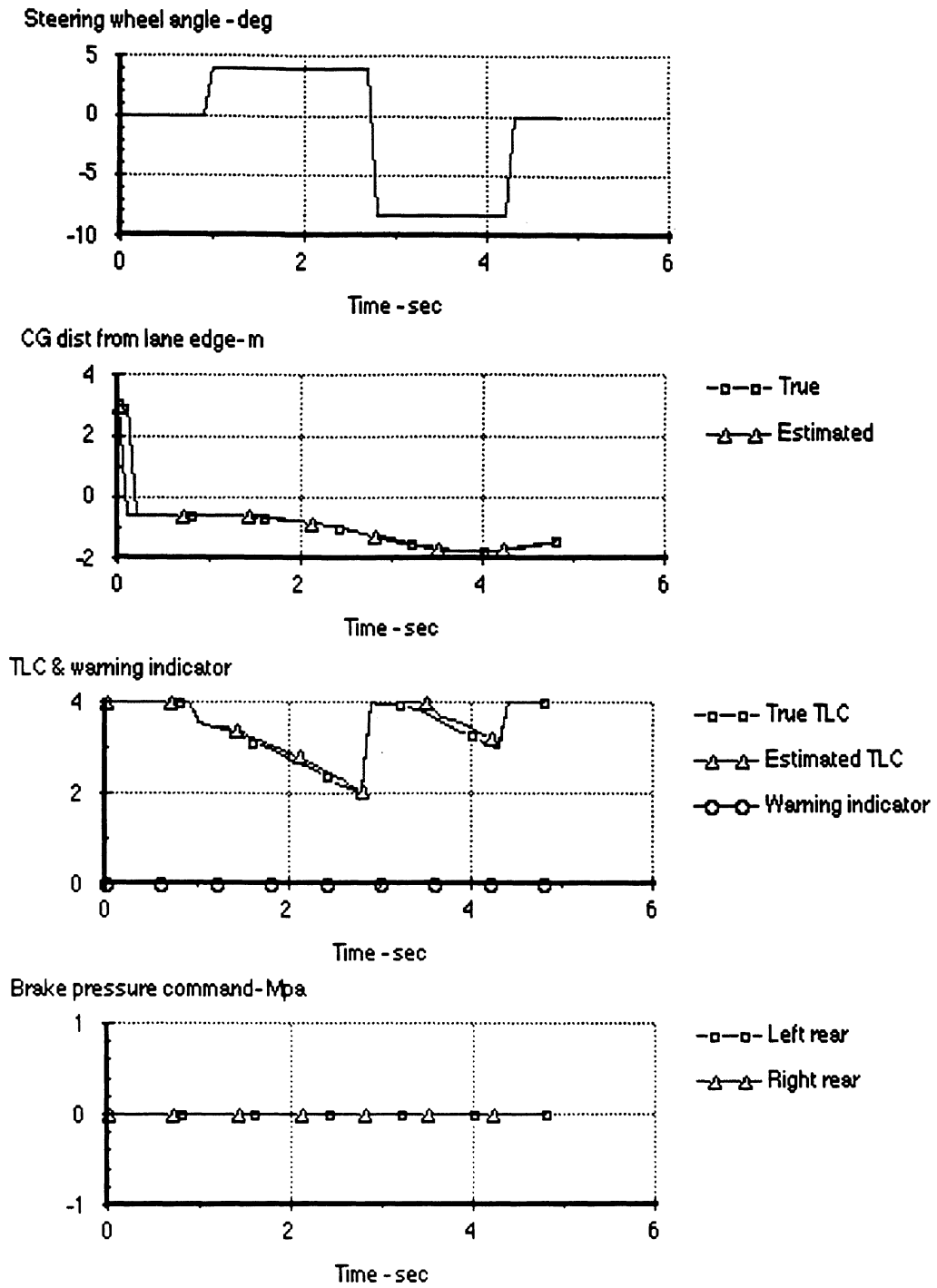


Figure 3.6.3:

Simulation: Warning is not issued when driver lane-keeps with normal authority

Normal weaving within lane:

- False alarms may occur in transitions on tight test track curves.

Aggressive weaving within lane:

- False alarms of warning likely to occur. Unwanted "interventions" should be rare.
- False alarms are induced especially by large steer angles.

Mild or moderate departures (< 2 deg departure angles and small acceleration of vehicle off roadway)

- Warnings will feel fine except in curves, where they may feel late.
- Most interventions are "successful" in straight-aways, once started-- starts after one wheel on shoulder, and keeps one wheel on roadway at all times.
- Interventions in curves are less successful than in straight-aways, sometimes allowing all wheels to leave road. Especially true at higher speeds.

Severe departures: (includes transitions, large departure angles, accelerating departures)

- Warnings may feel too early in straight-aways and too late in curves.
- Interventions in straight-aways begin before one wheel is on the shoulder, and are successful up to 3 deg departure angle at 90 kph.
- Interventions in curves are rarely successful at highway speeds -- the vehicle leaves the roadway and "ABS" limits the authority.

Table 3.6.2: Simulation: Limits of performance

3.7 Design of a System Simulation Tool

The different subsystems of a lane-departure avoidance system as described above have been combined on a simulation level in the CAPC simulation tool. The simulation tool is a modular concept written in C-language and runs on Macintosh computers with floating point processors. The main objective of the tool is to combine the subsystems into a road departure avoidance system and to study the performance and interaction among the different modules. The seven major parts of the CAPC simulation tool are the vehicle, the lane-marker recording and processing, the estimation of the future trajectory of the vehicle, the time-to-lane crossing calculation, the brake-steer controller, the driver-status assessment and the CAPC decision-making module. Appendix A is the User's Manual for the CAPC simulation tool. A more detailed document (the Reference Manual for the CAPC Simulation Tool) describes the models in the simulation, and is available upon request from the UMTRI Engineering Research Division.

The simulation tool has played a significant role as the CAPC system has grown to its final (hardware) design. Before implementation onboard the prototype vehicle, control strategies and state estimation methods have been implemented in the simulation tool to verify their efficiency and stability. The final controller and the CAPC simulation code have both been written in C-language, and the commonality of the code has simplified and shortened the implementation phase significantly.

The CAPC simulation tool offers the possibility to study the interactions among the various lane-keeping subsystems. For example, the influence of external disturbances like road roughnesses and wind on the determination of the perceived roadway geometry has been studied using the CAPC simulation tool. Since the vision system is mounted rigidly on the sprung mass of the vehicle, the motion of the vehicle will affect the estimates of the lane marker locations due to heave, roll, and pitch motions. The simulation tool was also used to explore the need for image stabilization as a counter measure to this problem.

3.7.1 Simulation Architecture

The core of the CAPC simulation is the vehicle model. The CAPC simulation tools offer a variety of vehicle models, each designed for a specific application. The most extensive model contains 14 degrees of freedom (DOFs). The sprung mass is able to move in 3 directions (longitudinal, lateral, and vertical) and to rotate about three axes (roll, pitch, and yaw). Each wheel suspension has one DOF with respect to the vehicle body and the rotation of a wheel also accounts for one DOF. This 14 DOF model contains all the major DOFs necessary for describing the motions of the vehicle for the CAPC application.

Two simplified models based on 7 and 8 DOF vehicle models are also available. The models are simplified with respect to the wheel suspensions. The 7 DOF model doesn't contain suspensions (and can therefore only describe the longitudinal, lateral, and yaw motion of the vehicle) and the 8 DOF model adds the roll motion of the sprung mass to the

7 DOF model. The simplified models are more computationally efficient but less accurate than the 14 DOF model.

It is important to consider that the tire plays a crucial role in vehicle dynamics. It accomplishes essentially three basic functions: (1) support the vehicle weight, cushioning road irregularities, (2) develop lateral forces for cornering, (3) develop longitudinal forces for accelerating and braking. In treating tire effects, several different tire models have been made available in the CAPC simulation tool. The most extensive is based on an empirical model known as the Magic Formula tire model. It can describe the tire slip force for a large range of load and slip quantities. Furthermore, it offers the possibility to treat the case of combined slip (cornering and braking or accelerating at the same time). The less extensive models are all based on the Magic Formula but contain certain simplification in order to speed up the simulation time.

The CAPC simulation vehicle can be operated on various roadway types with different geometries and road roughnesses. An oval test track, a straight road, a winding road, and a skid pad have been pre-programmed as options within the CAPC simulation tool. The geometry can be extended with grades and superelevations. Furthermore, several data sets of different artificially generated road roughness profiles are available. The road friction coefficient can be changed too.

Most of the maneuvers done with the simulation vehicle can be executed under the control of a driver model. Two driver models are available in the CAPC simulation tool. The simple driver model is characterized by a preview model that looks at a single point in front of the vehicle. The other option is a more elaborate optimal preview model that minimizes the tracking error at several points in front of the vehicle.

The CAPC vehicle was initially equipped with two vision systems: one for the near-range and one for the far-range area. Both cameras are modeled in the CAPC simulation tool. The image of the camera can be determined by applying several coordinate transformations to the known roadway geometry in front of the vehicle assuming a flat earth model. Hardware specifications describing the camera's CCD chip size and resolution have been modeled as well. Furthermore, the position of the cameras on the vehicle, the orientation and the update frequency can be selected over a wide range.

A summary of the subsystems available in the CAPC simulation tool follows:

◆ **Vehicle Models:**

- 7 DOF flat vehicle model: longitudinal, lateral, yaw and 4 wheel rotational DOFs
- 8 DOF yaw/roll model: 7 DOF model + roll DOF
- 14 DOF full vehicle model: 6 DOFs for the vehicle body, 1 DOF for each axle and 4 wheel rotational DOFs

◆ **Tire Models:**

- Steady-state (different road friction coefficients possible):
 - * Cornering stiffness as a function of vertical tire load
 - * Magic Formula (pure slip, combined slip)
- Transient: first-order relaxation system (tire load dependent)

- ◆ **Antilock Brake System Model:**
 - First-order lag system with brake pressure saturation
- ◆ **Driver Model:**
 - Simple preview model with driver limitations (time lag)
 - UMTRI's optimal preview driver model
- ◆ **Cruise-Control** (set, resume, accelerate)
- ◆ **Far-range Camera** determining the future roadway geometry
- ◆ **Near-range Camera** determining the heading angle and lateral deviation
- ◆ **Path Prediction** based on:
 - 2 DOF linear flat vehicle model (lateral and yaw DOF)
 - * Linear tires, fixed cornering stiffnesses
 - * Non-linear tires, cornering stiffnesses as a function of vertical tire load
 - 3 DOF linear yaw/roll vehicle model (lateral, roll and yaw DOF)
 - * Linear tires, fixed cornering stiffnesses
 - * Non-linear tires, cornering stiffnesses as a function of vertical tire load
- ◆ **Lane Margin Calculation:** time-to-lane crossing
- ◆ **Driver Status** assessment based on vehicle states and steering wheel activation
- ◆ **Brake-Steer Controller** based on LQ feedback of the lateral deviation, heading angle, side slip velocity and yaw rate
- ◆ **CAPC Supervisory Controller** based on driver status and time-to-lane crossing
- ◆ **Roll & Pitch angle estimation** based on measured suspension deflections
- ◆ **Road Roughnesses** collected from empirical road models
- ◆ **Roadway Geometry:** straight lines, curves w/lw/o superelevations and/or slopes
- ◆ **Wind Disturbances:** constant wind, crosswind gust, random crosswind
- ◆ **Real-Time Driving Simulator:**
 - 8 DOF yaw/roll model including wheelspin DOF, Magic Formula tires, cruise control, path-prediction, lane margin calculation, warnings and intervention

3.7.2 Simulation User Interface

The CAPC simulation is menu driven and is supported by various animations. The user is able to modify all important model parameters using pull-down menus and dialogs. A complete user's manual can be found in the appendix of this report. The user's manual gives a short overview of all the options that can be selected in the graphical user interface of the tool.

The simulations are supported by graphical and numerical outputs. The graphical animation shows the vehicle from a top view including the roadway geometry, predicted future trajectory and perceived roadway geometry. The scenery of the roadway as recorded by both vision systems is also displayed during the animation. Figure 3.7.1 shows a screen display of the animation.

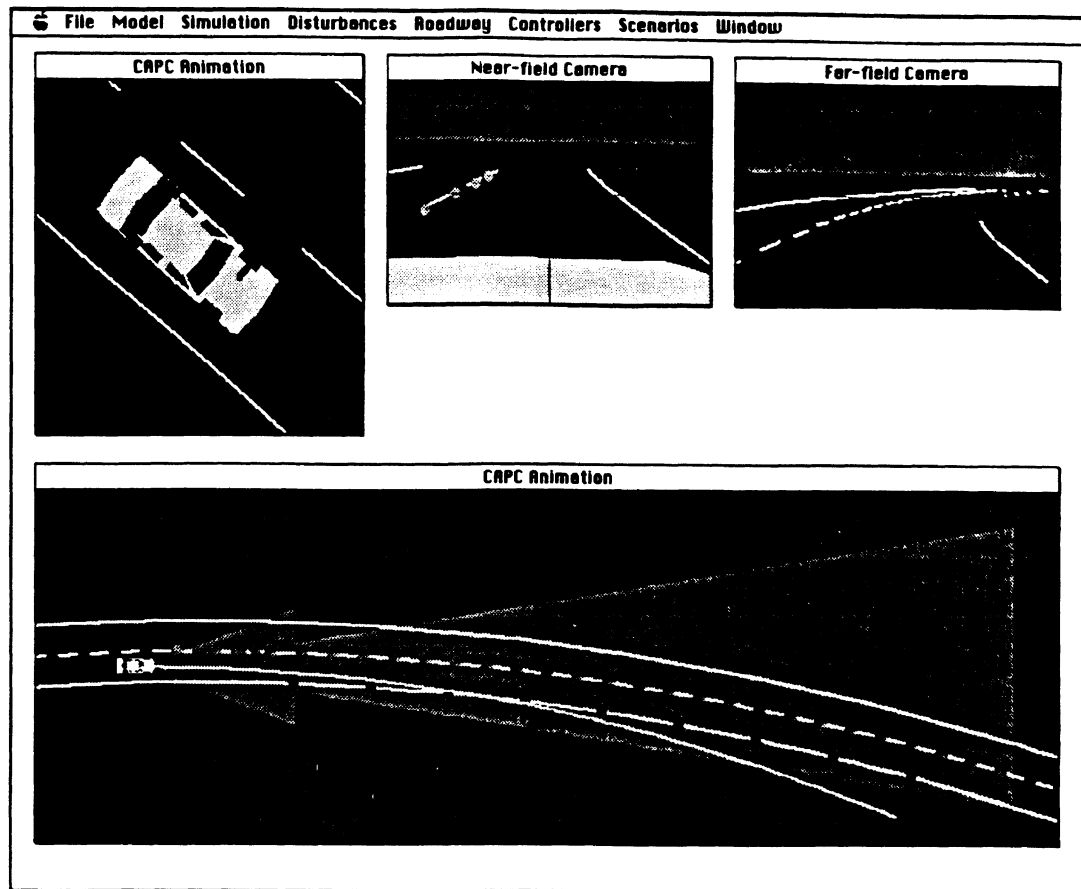


Figure 3.7.1. CAPC graphical simulation output.

The numerical output contains all important states of the vehicle, roadway geometry, disturbances, vision system outputs, and CAPC control in- and outputs. The data is stored in an ERD file format and can be displayed by a separate public-domain engineering plotter (EP) available from UMTRI. It is also possible to look at the simulation in progress (real-time) from the point of view of the driver. Figure 3.7.2 gives an impression of this driving simulator mode. This mode is based on the same modules as the simulation mode. However, most of the modules are simplified in order to let the simulation run in real-time. Some items have been disabled (such as the 14 DOF vehicle model) because of the computational burden of these modules.



Figure 3.7.2. CAPC driving simulator output.

3.7.3 Vehicle Model Evaluation

One should never attach value to simulation results if the models that were used to generate these results were not evaluated and compared with the real world. The most complex module with the largest amount of (unknown) parameters is the vehicle dynamics model. After the completion of the prototype vehicle, the simulation tool and the real world have been compared in order to tune the model (by altering its parameters) such that model and prototype car match. The CAPC prototype vehicle is installed with numerous sensors whose outputs can be compared with simulated signals. The following sensor signals are available:

- vehicle speed at each wheel (4 ABS wheelspin sensors)
- front wheel steer angle (displacement LVDT on the rack)
- steering wheel angle (LVDT on the steering wheel shaft)
- yaw rate (optical sensor)
- lateral acceleration (accelerometer)
- suspension deflection at each wheel (LVDT)

The post-processing mode of the CAPC simulation tool allows the user to port a measured ERD data file from the prototype car directly into the simulation. This mode has been designed especially to evaluate the vehicle dynamics models within the CAPC tool. The measured front wheel steer angle from the rack has been ported to the simulation tool

(multiplied by the steering system gear ratio) while making the steering elasticity of the vehicle model infinite such that measured and simulated steer inputs match. Later on the front-wheel steering elasticity has been derived by comparing the front-wheel steer angle with the steering angle measured at the wheel. The vehicle speed has been derived by averaging the four measured wheelspin speeds. This derived speed has been used as a set-point speed for the cruise control module in the simulation. Having both steer input and vehicle speed the driver model can be by-passed in the simulation tool and measured data from the vehicle can directly be compared with simulated signals from the CAPC tool.

Not all parameters needed to be adjusted. The geometry of the vehicle and suspensions could be derived from design drawings provided by Ford. The main unknown parameters were the vehicle's mass, moments of inertia and effective tire cornering stiffnesses. The mass of the prototype vehicle including equipment and driver + passenger on the right rear seat has been determined by four scales. From the four measured quantities the total front and rear axle load can be derived. Although the measured wheel loads showed a slight asymmetric pattern, the vehicle was modeled as symmetric with identical left and right wheel loads, so that the CG is located at the middle of the car.

The moments of inertia and tire-cornering stiffnesses have not been measured in the laboratory. A different approach has been used to identify these parameters. A least square optimization technique has been used to fit measured vehicle responses with equivalent simulated data. Several handling related tests were carried out with the prototype vehicle and the measured vehicle speed, front wheel steer input, steering wheel input, yaw rate and lateral acceleration were recorded with the MacDAS data-acquisition package. After that the measured front-wheel steer input and vehicle speed were ported into a simulated vehicle model and identical maneuvers were carried out with these real driver inputs.

The vehicle model used for the parameter identification was a simple 2 DOF vehicle with linear steady-state tire characteristics (based on a fixed cornering stiffness) and a transient tire model extension (based on a fixed relaxation length). In MATLAB the unknown cornering stiffnesses and yaw moment of inertia have been optimized using the FMINS function. The object function to be minimized is the sum of the quadratic errors between measured and simulated lateral acceleration and yaw rate. It is given by

$$J = \sum_0^{n_e} \left(q_1 (A_y^{meas} - A_y^{sim})^2 + q_2 (r^{meas} - r^{sim})^2 \right) \quad (3.7.1)$$

Weighting factors q_1 and q_2 might be chosen freely by the user. Since the magnitude of the yaw rate (in deg/s) is about the same as the magnitude of the lateral acceleration (m/s^2), coefficients q_1 and q_2 have both been chosen equal to unity. The optimization in MATLAB is such that the 2 DOF vehicle model is simulated for the total duration of the measured steer/speed input. The quadratic object function (3.7.1) is then computed and, in the LMINS procedure, the three parameters (front and rear tire cornering stiffness, yaw moment of inertia) are adjusted such that J gets smaller. The optimization is iterative and thus the vehicle model is simulated over and over until a minimum of J has been found.

The maneuver chosen to optimize the vehicle parameters was a moderately severe lane change ($A_y < 0.4 \text{ g}$). Higher levels of lateral acceleration should be avoided because these kinds of maneuvers cannot be described well enough with the simple tire model based on cornering stiffnesses. Table 3.7.1 shows the parameters of interest before and after optimization. The old inertia values were estimates based on the size of the vehicle [1].

variable	unit	OLD	NEW
front axle load	kg	1,086	1,091
rear axle load	kg	723	723
front tire cornering stiffness	N/rad	70,760	63,760
rear tire cornering stiffness	N/rad	59,850	66,430
steering elasticity	N/rad	50,000	20,000
yaw moment of inertia	kgm ²	3,562	3,960
roll moment of inertia	kgm ²	580	464
front suspension damping	Ns/m	1,295	2,590
rear suspension damping	Ns/m	1,032	2,063

Table 3.7.1 Vehicle model parameters.

As can be seen the effective front-tire cornering stiffness has been reduced significantly during the optimization process. This is mainly due to the steering elasticity and elasticity in the front suspension due to bushings compliances. By knowing the new effective cornering stiffness values, the parameters (a_3 and a_4) of the Magic Formula (MF) have been changed such that the effective cornering stiffness valid for the static front and rear tire load match the values as presented in table 3.7.1.

Since the vehicle model comes with the steering elasticity built-in, the cornering stiffness of the front tire $C_{F\alpha 1}$ must be derived from the effective cornering stiffness $C_{F\alpha 1}^{eff}$ according to table 3.7.1 using the geometry of the front suspension in combination with the known steering elasticity. Its value is given by

$$C_{F\alpha 1} = 0.5 \cdot \frac{2 \cdot C_{F\alpha 1}^{eff}}{1 - 2 \cdot C_{F\alpha 1}^{eff} \frac{t_m + t_p}{c_{st}}} = 84,943 \text{ N / rad} \quad (3.7.2)$$

with

t_m the mechanical trail of the front suspension (0.02 m)

t_p the pneumatic trail of a front tire for small slip angles (0.0191m)

c_{st} the steering elasticity (2.0e4 Nm/rad)

This value is considerably higher than the effective cornering stiffness and it is also higher than the value for the rear tires.

The roll moment of inertia and suspension damping coefficients have been tuned using frequency response functions and this analysis will be described hereafter. Figure 3.7.3 shows a comparison of the measured prototype vehicle data with simulation results. Two

vehicle models were simulated: the simple 2 DOF model with linear tires and the complex 14 DOF model with non-linear tire characteristics (MF). The type of maneuver can be characterized as a series of lane changes with increasing severity at constant forward speed.

The simulation results match quite well with the measurement data. Even the simple 2 DOF model does a surprisingly good job when the severity increases. However, once higher levels of lateral acceleration are reached the simple, cornering-stiffness tire model is not sufficient because it neglects the influence of the load transfer on the tire slip forces and thus the sum of simulating both tire side forces at an axle is higher than in reality. Here the 14 DOF model with the Magic Formula tire model is doing a much better job.

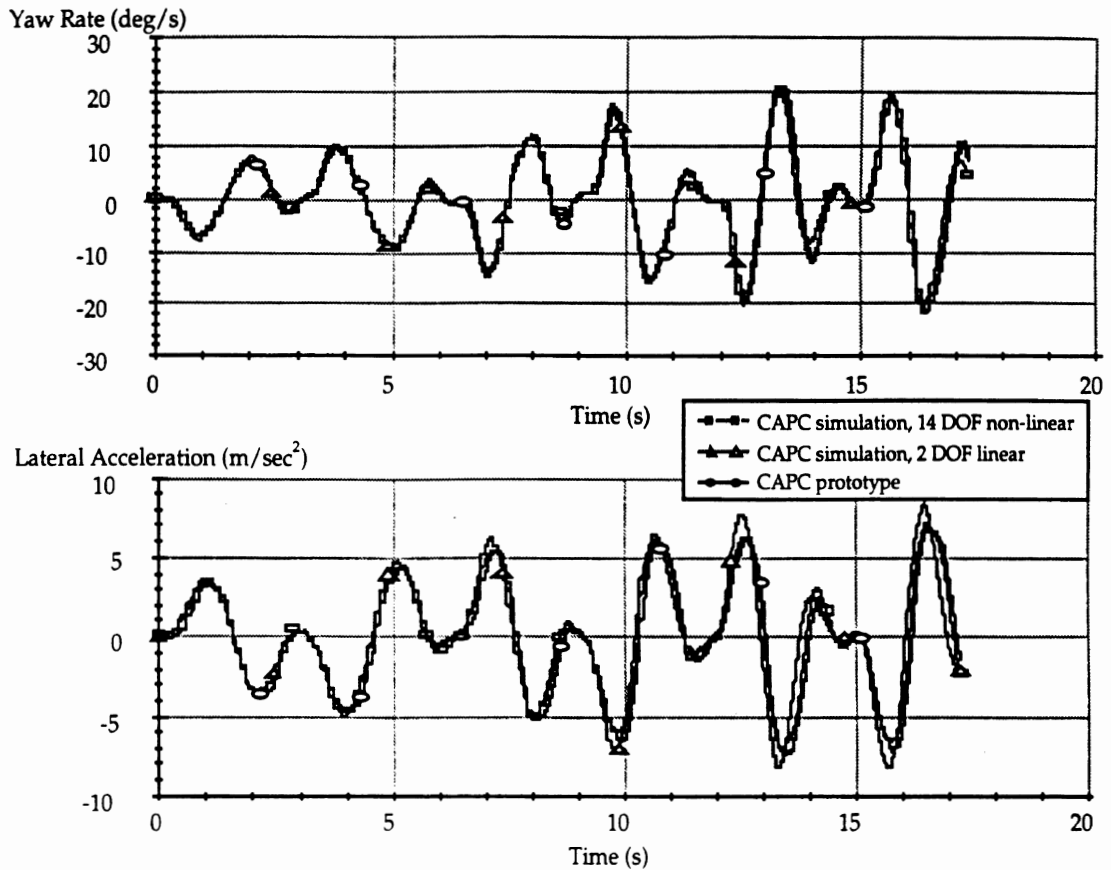


Figure 3.7.3. Lane changes at 121 km/h (75 mph).

Frequency response functions (FRFs) are also valuable for tuning vehicle models. In particular the FRF of the lateral acceleration and yaw rate over the steer input can be of use in the model evaluation phase. A random steering maneuver at a constant vehicle speed has been shown to be the best type of maneuver, when applied correctly, because the measured data contains many frequencies of interest. The total duration of the measurement sequence was limited to about 17 sec. This sequence has been split into four windows each 4.25 seconds long. Fast Fourier transformation techniques have been used to calculate power/cross spectral densities, transfer functions and coherencies. For the vehicle models it has been chosen to use the linear representation of the model (linearized set of equations of motion) to calculate frequency response functions in an analytical way. In that case no time

domain simulations are required. Figure 3.7.4 and 3.7.5 present two important FRFs. The solid lines correspond to the measured response and the dashed one with the 2 DOF linear model. The dash-dot line is valid for a 10 DOF linearized model. This latter vehicle model is equivalent to the 14 DOF model, however, without the wheelspin DOF.

For the yaw rate FRF not much difference can be seen between the two simulation models and prototype car. This is not surprising because both models behave almost identically in the yaw degree of freedom as a rigid mass with two lateral dampers (tires). For the lateral acceleration FRF there is a substantial difference above 1.5 Hz. The main reason being that the accelerometer in the car was not exactly attached in the CG of the vehicle. It was installed 0.25 m behind and 0.17 m below the vehicle's CG. Therefore the measured lateral acceleration consist of a combination of true lateral acceleration and yaw and roll acceleration. The FRFs for both vehicle models were corrected for this offset. Since the 2 DOF model doesn't include the roll degree of freedom a difference between measured and analytically determined response for the lateral acceleration FRF is present.

The roll moment of inertia and suspension damping have been altered in order to match the measured response with the 10 DOF model FRF. With the initial set of parameters the roll motion related resonance peak near 2 Hz was not well damped and thus the damping has been increased. The location of the roll natural frequency has been used to tune the roll moment of inertia while keeping the suspension and antiroll bar stiffnesses constant. After some trial and error both the measured FRF and the 10 DOF FRF did agree very well.

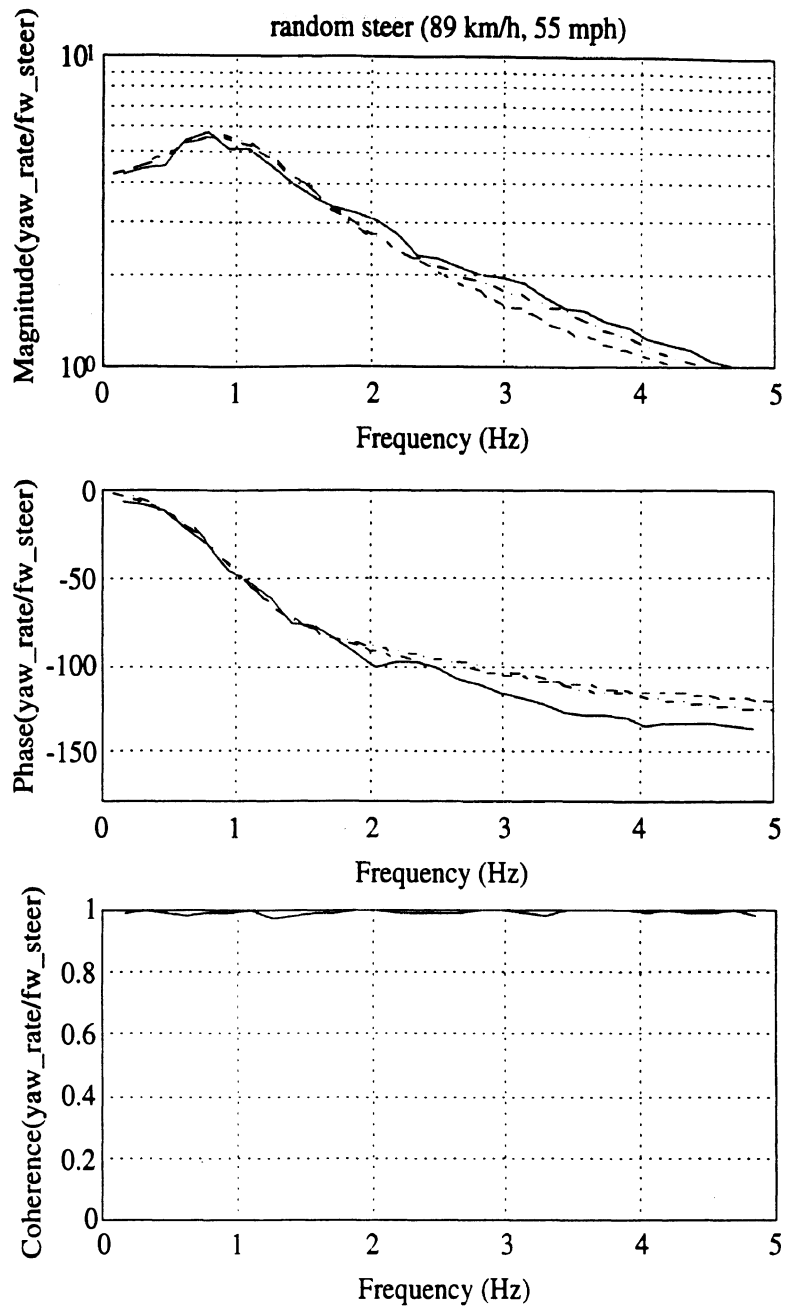


Figure 3.7.4. Frequency response functions (yaw rate vs. steer input).

- = measured with the CAPC prototype vehicle
- - - = 2 DOF linear vehicle model
- . - . = 10 DOF linearized vehicle model

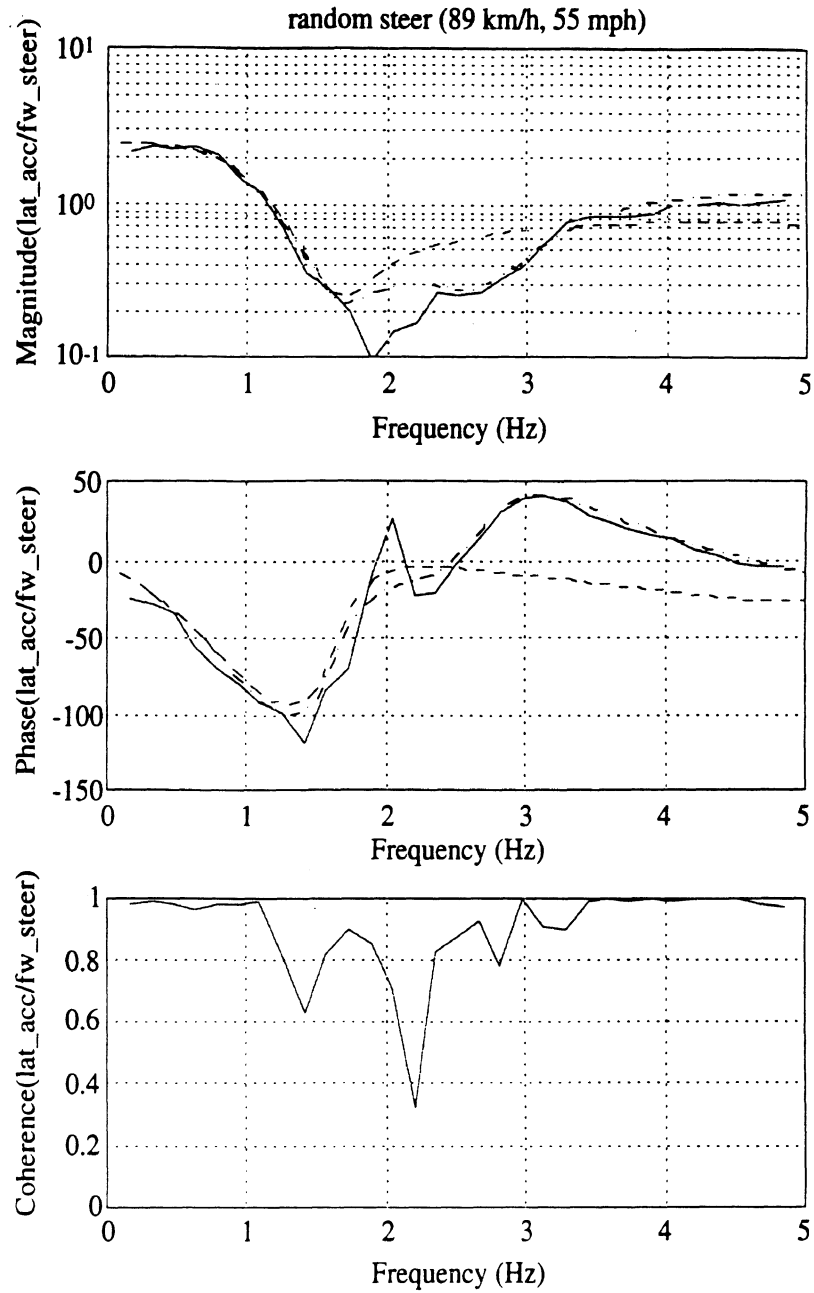


Figure 3.7.5. Frequency response functions (lateral acceleration vs. steer input).

- = measured with the CAPC prototype vehicle
- - - = 2 DOF linear vehicle model
- . - . = 10 DOF linearized vehicle model

4.0 Implementation of CAPC prototype

The CAPC prototype hosts a number of processors, controllers, and specialized hardware to implement the function. Transducers, signal processing, data collection and storage are handled via UMTRI's MacDAS system. This provides a basic framework in which experimental controllers can be developed. The MacDAS system and the CAPC controller algorithm reside on a Macintosh Quadra computer. Lane sensing, analog signal processing, and brake pressure servo control are handled by other processors. A block diagram of the CAPC prototype is shown below. Separable components and high level interfaces allow integration of other data sources.

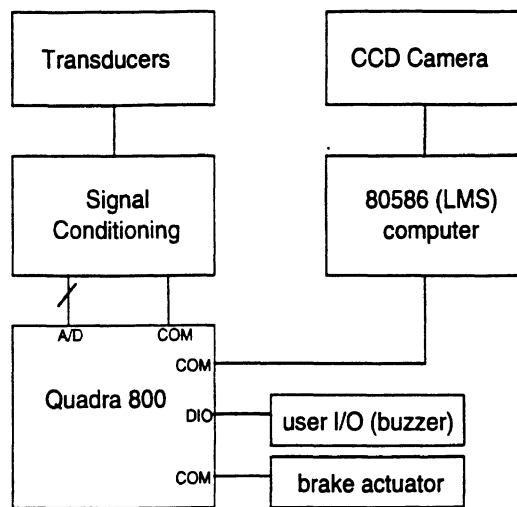


Figure 4.0.1: CAPC Prototype block diagram

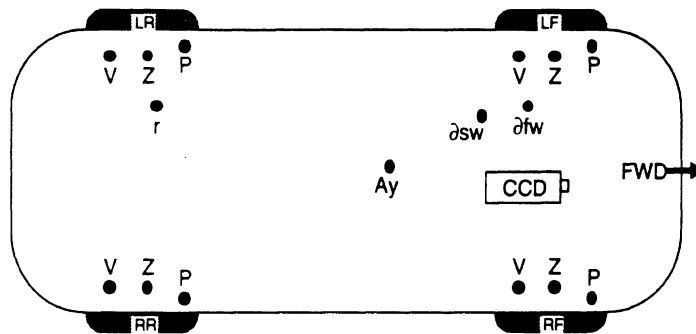
4.1 Prototype System Configuration

Brake Actuator

The hydraulic brake system of the Taurus SHO has been modified by Ford Motor Company. The system currently provides for independent operation of each of the four wheel brakes on the SHO. Each brake caliper is equipped with a pressure sensor such that a brake pressure loop can be closed around each wheel. The control mechanism is intended to be a build-dump-hold pressure servo using a modest amount of hysteresis about the desired pressure point. The CAPC function will determine the desired amount of pressure to be applied and which wheel it will be applied to. When the brake actuator receives this information via an RS-232 link, it operates the valves and closes the pressure loop, executing a digital control loop at several hundred Hz. Schematics are included in Section 8.4 for the designs of brake-steer controller interface wiring, pressure transducer signal conditioning, power supply design, and communications circuitry to the BlueEarth Micro-445 microcontroller chosen to close the pressure servo loop. These circuits were designed but not constructed on the CAPC prototype during the project's Phase I.

Transducers

The vehicle contains a suite of transducers for measuring or estimating vehicle motion including roll, pitch, velocity, lateral acceleration, and yaw rates. Also sensed are driver inputs and roadway geometry. A table of transducers is shown along with a few specifications. The vehicle drawing indicates the transducers' locations .



V - forward velocity	freq-to-voltage converters, 0-45m/sec
Z - suspension defl.	LVDT signal cond., $\pm 60\text{mm}$ fr, $\pm 120\text{mm}$ rear
∂sw - steer wheel angle	string pot around shaft, ± 90 deg, no zero
∂fw - front wheel angle	LVDT on rack, ± 6 deg (at front wheel)
Ay - lateral acceleration	servo-type accelerometer, $\pm 10\text{deg/sec}$
r - yaw rate	optical gyro - ± 100 deg/sec, no drift
also	
P - brake caliper pressure	installed by Ford, 3000psi max

Figure 4.1.1: Transducers installed in CAPC prototype

Dots in figure 4.1-1 indicate nominal transducer location. Note that the instrumentation at all four wheels is identical. The Linear-Variable Differential Transformer (LVDT) indicated as "Z" for suspension measurements (hence pitch and roll estimates) are installed such that the motion between the sprung mass and the unsprung mass is transduced (i.e., the change in length of a suspension strut). Pitch and roll are then computed knowing the vehicle's wheelbase and track width dimensions. In this way we can deduce the gross motions of the sprung mass, except for contributions due to tire deflection. The LVDT for front-wheel steer angle (∂_{fw}) transduces the linear motion of the steering rack. The string pot for driver's steering-wheel input (∂_{sw}) is attached to an aluminum grooved barrel, which is clamped around the shaft between the tilt mechanism and the universal joint at the firewall. Vehicle velocity (V) is transduced using the OEM magnetic pickups provided with the ABS system, and a frequency-to-voltage conversion. (See Section 8.4 for a schematic of the frequency-to-voltage conversion circuit.) The accelerometer and yaw rate transducers are commercial products and follow a typical installation. The CCD camera is part of the LMS system. Photos of the LVDT and CCD camera installations are shown in Figures 4.1.2 and 4.1.3.

Calibration of steering-wheel angle (in terms of both front wheel angle and rack displacement) and suspension deflection was performed on UMTRI's Suspension Measurement Facility, shown in Figure 4.1.4. In this facility, the vehicle's suspension was exercised in a known manner and the outputs of the transducers calibrated to the motion. The suspension LVDTs were calibrated to indicate the purely vertical change of the spindle at the lateral center of the tire. The front-wheel steer angle was calibrated to indicate the angle of the front tire plane (including static compliances at three different load conditions). The driver steering-wheel input was calibrated with reference to a known rotary transducer. Yaw rate and lateral acceleration transducers were calibrated prior to installation in the vehicle.



Figure 4.1.2: Suspension LVDT and steer LVDT installed in CAPC prototype



Figure 4.1.3: CCD camera installed in CAPC prototype

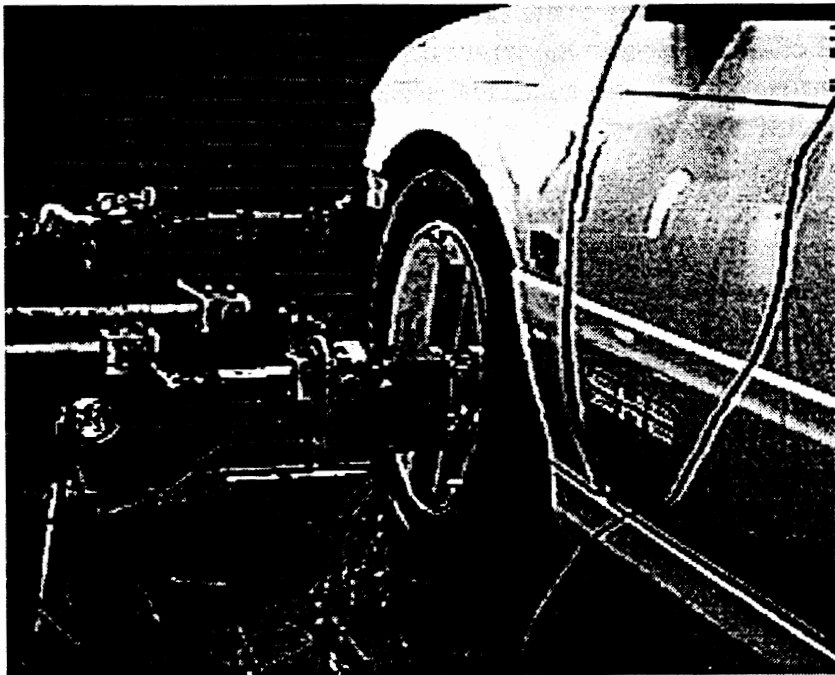


Figure 4.1.4: UMTRI's Suspension Measurement Facility

Signal Conditioning

All analog transducers shown above receive signal conditioning from customized circuitry. The LVDTs and velocity pulsers use special signal processing stages to generate the proportional output. A block diagram of this circuitry plus channel gain, offset, and filtering is shown below in Figure 4.1.5. See Section 8.4 for detailed schematics of the LVDT and velocity signal processing designs.

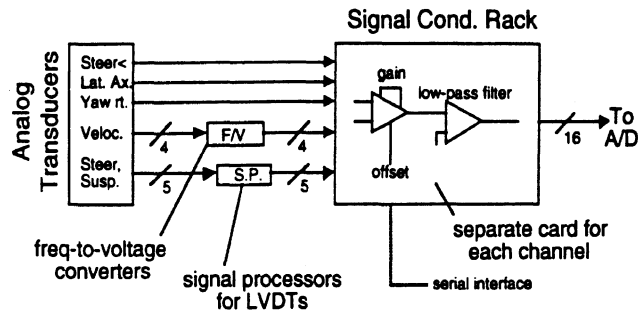


Figure 4.1.5: Signal conditioning block diagram

The analog gain and filter stage is housed in a single unit and controlled via RS232 serial communications to the Quadra host computer. The analog rack has automatic gain and offset measurement and zero adjustment capability, and can be fully calibrated automatically via the MacDAS system. The CAPC code initializes the rack to the desired configuration at startup.

Computers

There are many processors in the CAPC system. The primary computers are the Macintosh Quadra 800 executing the CAPC function computations and data acquisition, and the 80586 implementing the lane-mark sensor system. Ancillary processors are a microcontroller in the analog rack, one for the brake servo pressure loop, and processors in the imaging camera and on computer I/O cards. A photo of the primary computer installation is shown in Figure 4.1.6.

The Quadra 800 hosts MacDAS, UMTRI's standardized, test-platform, control code. The system is an infrastructure for controllers and data acquisition, providing the control system with analog and digital I/O, serial I/O, graphics, real-time loop management, user interfaces, and standard ERD file I/O. The MacDAS system also controls the analog rack and shields the control algorithms from hardware upgrades.

The Brake Servo design hosts a microcontroller (a BlueEarth Micro-445) for closing the build-dump-hold pressure loop. This micro contains analog and digital I/O and dual RS232 serial ports. The pressure commands from CAPC are transmitted via serial line to the controller. The controller also generates a serial output stream of pressure values which can be collected using a laptop or other computer.

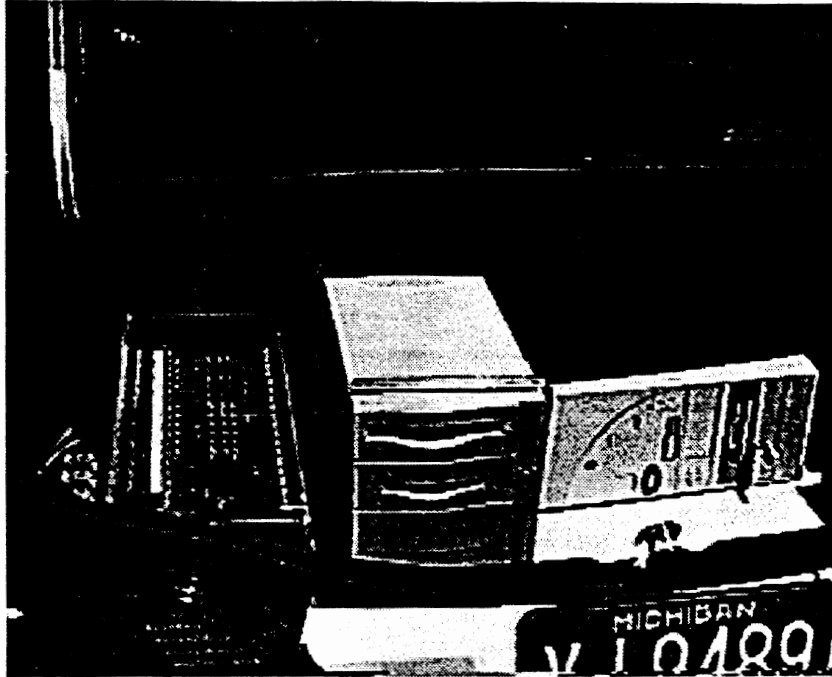


Figure 4.1.6: Signal conditioning rack and computers located in the CAPC prototype vehicle's trunk.

Data Communications

Communications between modules in the CAPC system are largely RS232 serial, including MacDAS with the analog rack, CAPC with the LMS system, and also CAPC with the Brake Servo processor. The communication protocol developed between the CAPC function and the LMS system is outlined in detail in Section 8.3.2 and Appendix B, while the basic interaction is illustrated in Figure 4.1.7.

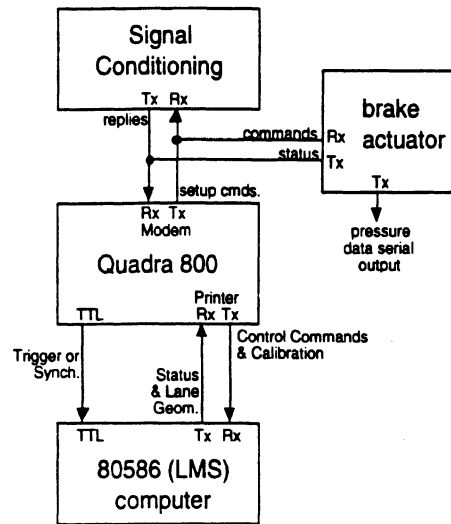


Figure 4.1.7: Communications block diagram

RS232 communications convey information to and from the Quadra computer (hence the CAPC function and MacDAS infrastructure) to the signal conditioning rack, the brake controller, and the LMS computer. Commands are typically sent from the Quadra and replies or status information are returned. The communications to the LMS system consist of operational commands outlined in Section 8.3.2. The LMS system will also return lane geometry via this connection. The analog rack receives control for gain, filtering, and offset adjustments, and provides status information. This is typically done at startup, and is never performed during the actual running of the CAPC function. When the system is in operation, all communications are ignored by the analog rack (it is placed in a stand-by mode) but are utilized by the brake servo controller. The servo can also report its status to the CAPC system. Note that neither MacDAS nor CAPC collect the brake servo's pressure data over this line, so an additional transmit line is provided for hooking up a separate processor (such as a laptop) with a serial port for collecting data. This data is simply a stream of values indicating the pressures at each wheel transducer, reported at 20Hz.

4.2 Physical Installation in Taurus SHO

Installation of the MacDAS system and the CAPC function was accomplished using UMTRI's shop facilities. Wiring for transducers and controllers was installed in the vehicle's wiring channels wherever possible. CCD camera signals and power were routed through the roof frame system, under the headliner. Also present are duplicate CCD signal, power, and extra twisted-pair cables to support a second camera; other sensors are also possible. Section 8.4 includes schematics for the analog transducer cabling.

The LVDTs for suspension deflection measurements were mounted using customized bracketry attached to the vehicle's suspension components and a rigid member of the frame or body. The steering rack LVDT body was also mounted on a customized bracket and the rod-end attached to the rack's lateral gear. Vehicle velocity is generated by picking up pulse signals from OEM supplied ABS wheel speed sensors.

The ABS system has been disabled since it will not function properly with the modified braking system. The ABS control unit itself was also removed from the OEM housing, and its housing and connector were used to interface the brake servo controller described above. This connector also provides convenient interface to the velocity pulses, ABS relay power, and brake switch signals formerly used by the ABS controller. All brake controller signals and power have been routed to a space behind the vehicle's stereo system, where the servo controller would be installed.

The accelerometer was located near the C.G. in the lowest section of the front seat arm rest. The yaw rate transducer was conveniently located in the trunk with the primary computers and signal conditioning. Also in the trunk is the power distribution system consisting of a 1000W square-wave inverter whose input is tied directly to the battery of the vehicle. The vehicle's charging system will maintain sufficient battery charge as long as the engine is running above idle - i.e. while driving. Cooling for the equipment in the trunk was necessary. A simple solution was the removal of speakers in the rear deck and installation of a fan to move cooler cabin air through the trunk.

Great care has been taken to eliminate all ground loops in the test platform and to assure maximum noise rejection. All system installations were designed to receive a single ground reference at the inverter. Mountings to the vehicle are made through insulated mechanisms such as wood and plastic to avoid contact between instrument cases and the vehicle chassis. Many transducers are isolated based on their technology while others required insulated mountings. At present there is a single loop which exists between the CCD camera and the vehicle chassis where the camera is attached to the roof frame. This connection can be easily broken by inserting a delrin (or similar material) barrier between the camera and mount. Since there is at present no voltage drop across this loop and hence no current flow, the loop has not caused any problems. Future modifications or additions to the vehicle will likely warrant isolating the camera and recalibrating the sensing system.

5.0 Testing of the CAPC Prototype

5.1 Test Plan and Rationale

The CAPC vehicle prototype testing has consisted of three basic activities:

- 1) calibration and sensor noise quantification
- 2) identification of vehicle model parameters
- 3) verifying qualitatively that the TLC and decision algorithms behave as expected

Testing has confirmed qualitatively that the prototype provides the basic functionality of a road-departure, warning-and-intervention system. There has not yet been the opportunity to conduct thorough quantitative testing of the performance of either the subsystems or the system. Such quantitative testing will require "ground truth" knowledge -- that is, external measures of the vehicle's position in the lane, as well as a time-tagging to provide a means to compare the TLC that is computed on-line to the actual truth. This initial phase of the project has been primarily a design and implementation phase, and the testing completed to date provides evidence that the system is performing as intended.

In the next sections we review test results that address the items above.

5.2 Test Results

5.2.1 Experimental data

Sensor calibrations

Sections 3.2 and 4 described static calibrations of the LMS and the analog systems, respectively.

Analog transducer noise stationary vehicle

Analog transducers are filtered with a 10Hz lowpass filter before digitization. There is no digital filtering. Figure 5.2.1 shows traces of the analog transducer signals while the car is stationary, the engine is revved at 3000rpm, and all vehicle systems are running off the onboard power. This gives an indication of electrical noise on each of the transducers. The 3 sigma values for each measurement signal (for electrical noise only) are approximately as shown in Table 5.2.1. Figure 5.2.2 shows similar traces collected while the vehicle was being driven at 98 kph (62 mph). The signal variations now include disturbances of the roadway environment.

steering wheel angle	0.4 deg
front wheel steer angle	0.03 deg
suspension deflection	0.4mm
yaw rate	0.15 deg/s
vehicle speed	-na-

Table 5.2.1 Electrical noise values for transduced signals (3 sigma, estimated)

Vehicle dynamics modeling parameter estimation

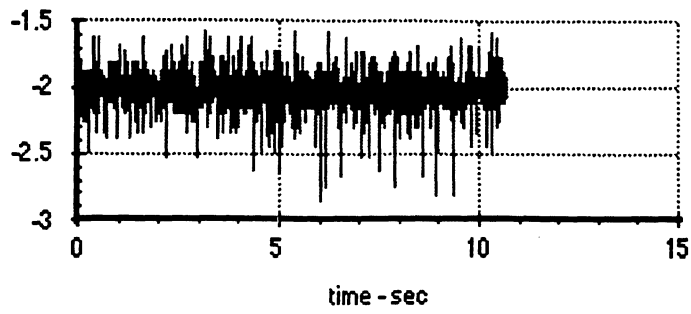
Section 3.7.3 describes the testing performed to compute values for vehicle yaw inertia, cornering stiffness at the front and rear tires, and suspension damping values. The yaw

inertia and cornering stiffness are especially important because several sets of gains in the TLC and brake-steer algorithms depend on these parameters.

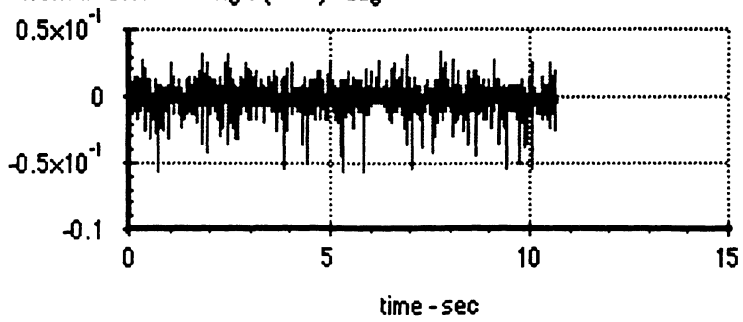
TLC while steering sinusoidally

Figure 5.2.3 shows traces of signals collected while driving on a straightaway at highway speed and while gently steering with a sinusoid-like motion, so that the vehicle traveled back and forth in the lane. The figure shows how for relatively large steer motions, the estimates of lateral position are still consistent. The TLC drops below four seconds several times, and as steer amplitude is increased, warnings and intervention buzzers are triggered. We point out the agreement in this data between the two Kalman filters estimates, and especially that the yaw estimate coming from the near-range Kalman filter matches the direct output of the yaw rate sensor. The curvature estimates are very nearly zero.

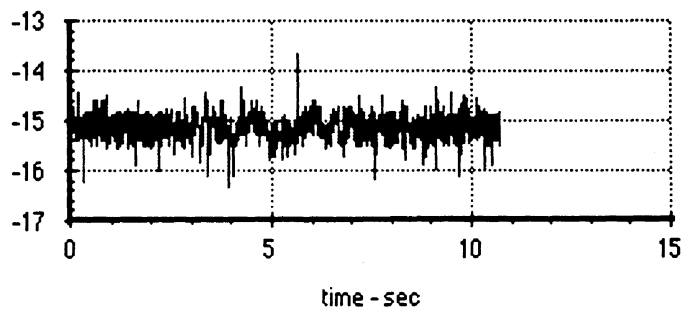
steering wheel steer angle - deg



front wheel steer angle (rack) - deg



suspension defl., right front - mm



yaw rate gyro - deg/s

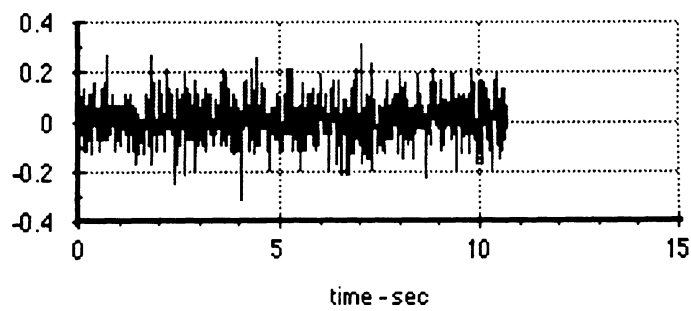


Figure 5.2.1 Analog transducers -- stationary vehicle

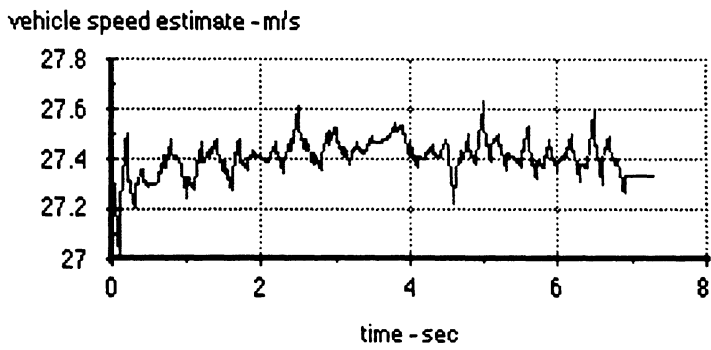
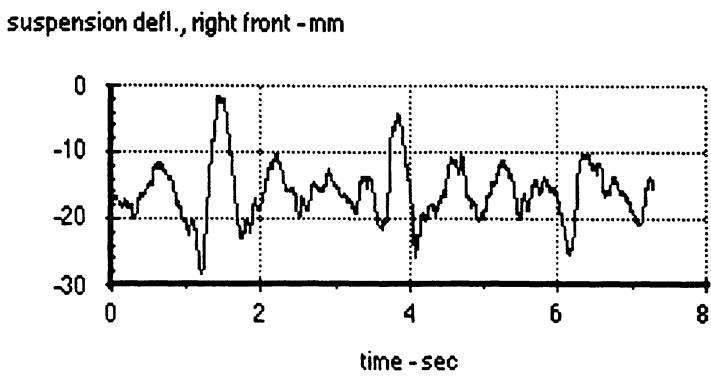
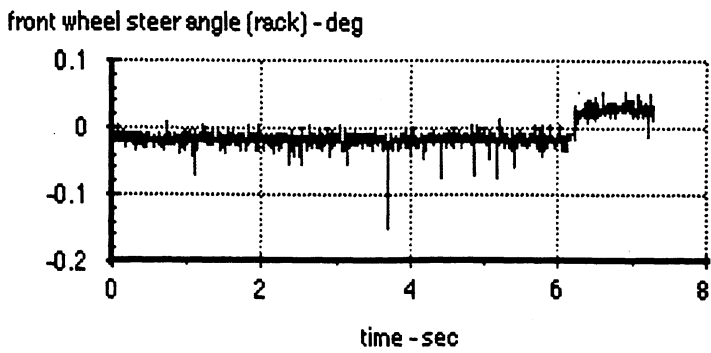
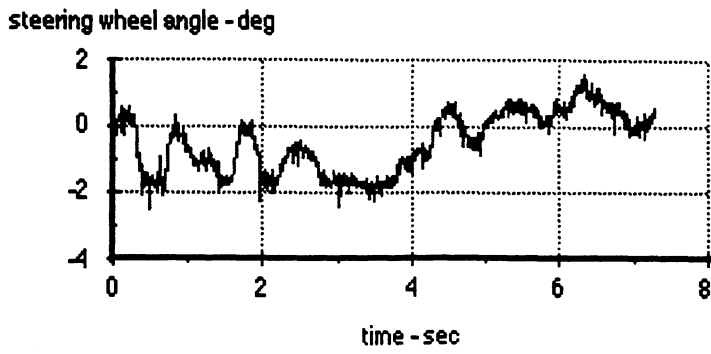
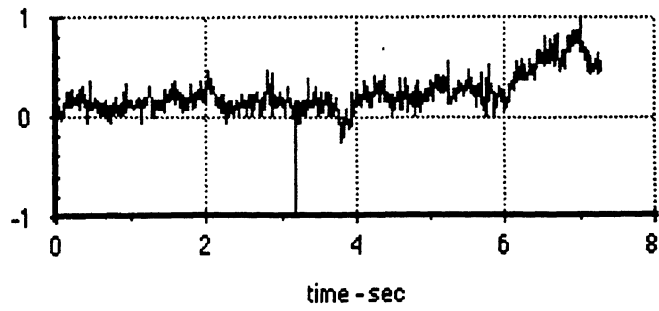
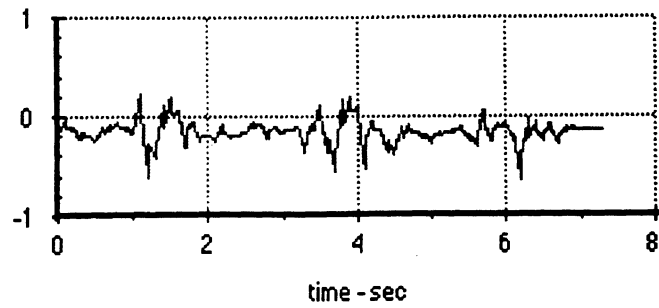


Figure 5.2.2 (page 1): Analog transducer signals, 98 kph

yaw rate gyro - deg/s



estimated pitch - deg



estimated roll - deg

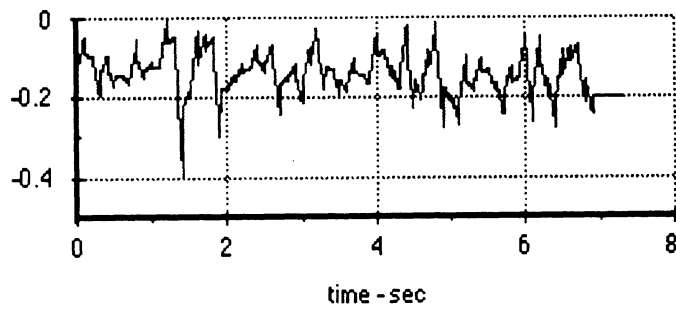


Figure 5.2.2 (page 2): Analog transducer signals, 98 kph

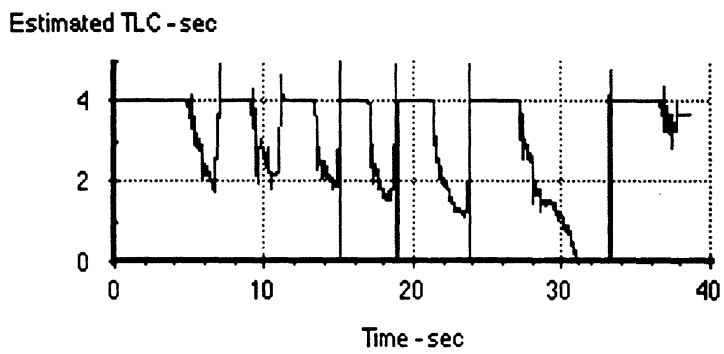
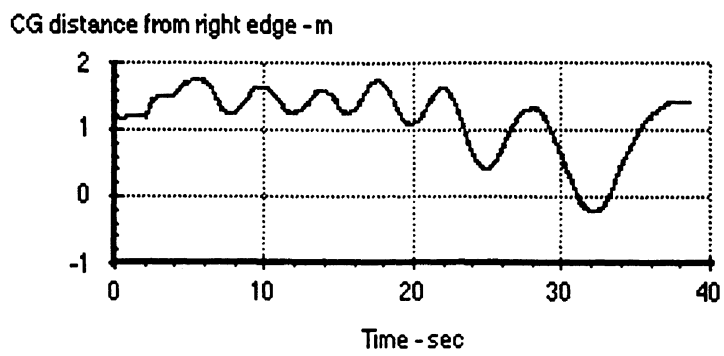
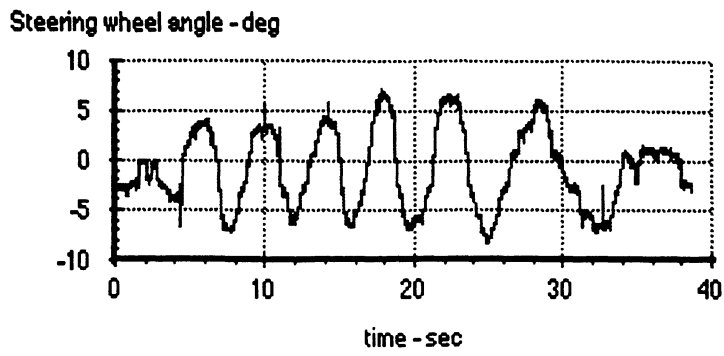


Figure 5.2.3 (page 1): Sinusoidal steering to trigger warnings and intervention

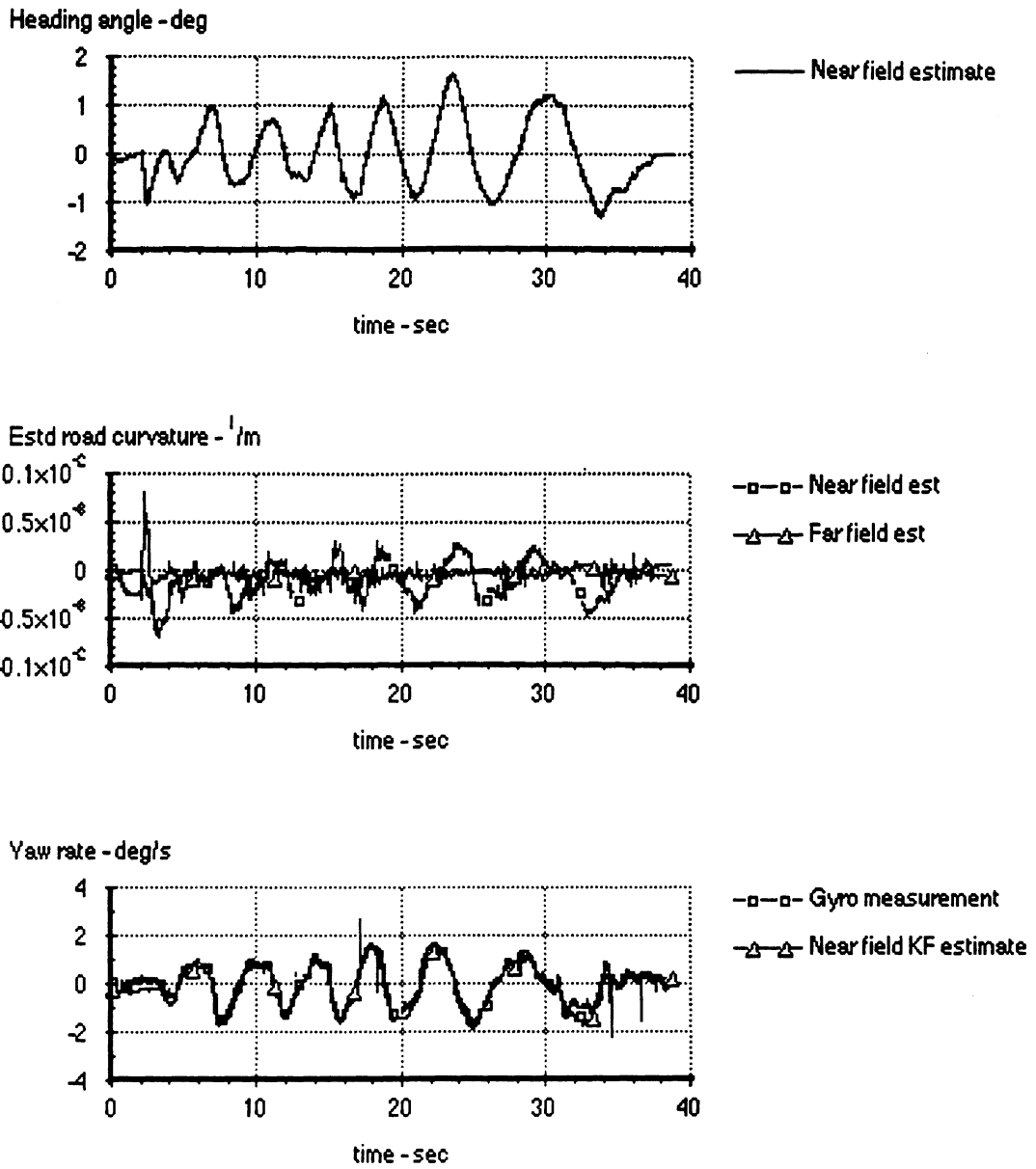


Figure 5.2.3 (page 2): Sinusoidal steering to trigger warnings and intervention

TLC and the warning & intervention function during road departures:

Numerous tests were conducted to see how the warning and intervention signals "felt." Figure 5.2.4 below is an example of these tests, in which the vehicle is steered into a path of departure. The estimate of the CG's position shows that the CG crosses at a time of 13 seconds, whereupon the TLC is also approximately zero. Before that time, at approximately 10.8 sec, the warning was triggered, and a short time later, as the TLC dropped below one second, the intervention signal goes high. We note that there is 0.2 sec delay in the TLC subsystem, and so the CG will cross the lane marker ideally when TLC has computed 0.2 sec.

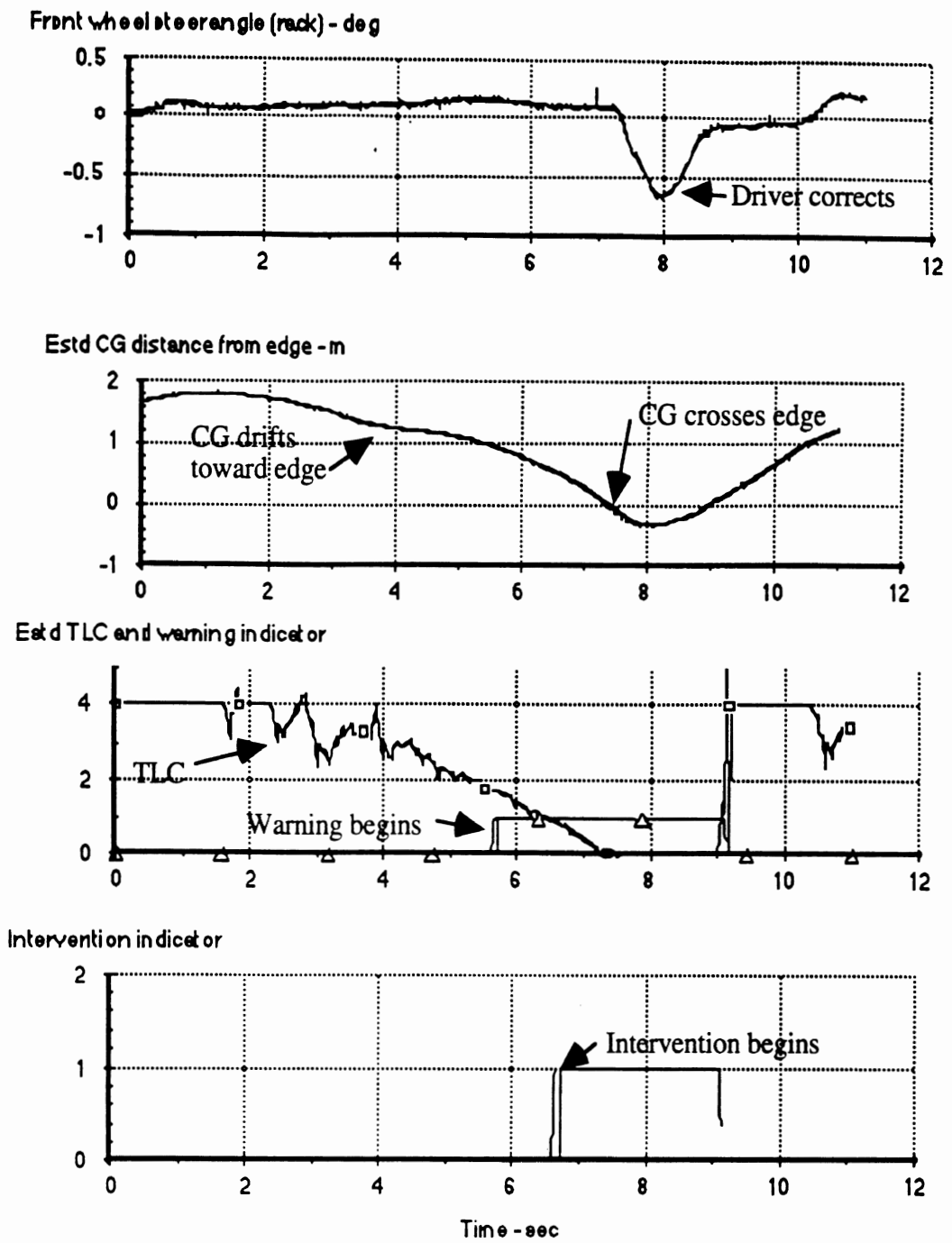


Figure 5.2.4 : Warning and intervention indicators as vehicle drifts out of lane

5.2.2 Performance constraints during testing

As noted earlier, the CAPC road-departure prevention system is designed to prevent unintended road departures due to driver inattention. [Daytime, dry-weather test track-like conditions] are assumed. When the system is operated under other conditions -- such as on urban freeways, under difficult lighting conditions, or for severe departure conditions -- the performance can degrade. Below are items seen during testing which constrain the circumstances under which CAPC works as intended. Please note that most of these are well outside the bounds of the design conditions.

System constraints:

For best performance the system must be operated on pavement that is in reasonably good shape. Potholes or rough pavement jar the vehicle, resulting in spikes in analog transducer measurements, which in turn create erroneous jumps in the estimated TLC. The result is an increase in false warnings.

We find, as predicted in simulation, that for the current decision rules there is a tradeoff between minimizing false warnings and succeeding in delivering a timely warning soon enough during severe road departure scenarios. If the TLC thresholds are too high, the driver experiences false alarms during the course of normal lane-keeping excursions, and the system is more susceptible to false warnings induced by outliers from the sensors. If the TLC thresholds are too low, warnings are not issued until the vehicle has traveled further toward the road edge. In some departure scenarios, there would not be enough warning for the driver to react and recover the vehicle before all four wheels would leave the lane.

At very low speeds the accuracy of the vehicle dynamics models used in TLC computation degrades. Operation is now limited to speeds greater than 10 m/sec. At very high speeds, the Kalman filters used in the TLC computation are not well tuned; operation is limited to speeds below 50 m/s -- i.e., obviously covering the range of all legal speeds in the U.S.

Road superelevation should not be greater than six degrees. For superelevations greater than about three degrees, special tuning of the Kalman filters used in TLC computation is required. Virtually all U.S. freeway mileage (in the through lanes of travel) involve superelevation values below three degrees.

Road radii of curvatures should be 300 meters or more for a vehicle traveling at highway speeds. (This covers the Ford Michigan Proving Grounds and all U.S. interstate highways.) Greater curvatures can be accommodated with further tuning of the Kalman filters and with redesign of the LMS camera parameters, so that an adequate field of view is obtained.

Decision tuning: If the shoulder is quite narrow, then a system designed for wide shoulders may allow the vehicle to leave the roadway. Conversely, if the system is designed for very narrow shoulders, false alarms will increase and the warning and intervention will feel a bit early to the driver.

Sufficient LMS data are required to make decisions to issue warnings or to indicate when an intervention would occur. No new warning or intervention decision is indicated in the

current CAPC logic unless the LMS has reported four points from the right-side lane marker for five or more consecutive frames.

LMS constraints:

The CAPC system issues warnings and indicates when interventions would occur *only* for right-side departures from the current lane. This is done because the LMS does not report sufficient data from dashed lines for reliable TLC computation, and because the LMS currently does not decide whether a lane marker is dashed or solid. We note that the CAPC simulation tool (Section 3.7) includes logic to compute TLC only to solid lines (road edges), and that the LMS system currently has the ability to communicate this information. Accordingly, this constraint will be removed with additional development.

The LMS initializes its operation assuming that the vehicle is traveling roughly down the middle of a lane.

To avoid false alarms, the system is disabled manually at exit and entrance ramps and other areas where the rightside lane marker is interrupted.

Twelve-foot lanes are assumed, and the near-range search windows are steered together. Therefore, when operating in lanes of significantly different width, the CAPC function can be affected if the LMS locks onto the left-side marker. This is true because the right-side search windows are then centered twelve feet to the right of the left marker, and this leads to missing the marker entirely. The LMS contains a constant parameter for lane width, which can be changed by an engineer.

When operating with traffic in, or near, the lane of travel, false alarms can be triggered by the LMS locking onto either the vehicle ahead or its shadow. This is because the roadway geometry estimation assumes that all LMS data comes from lane markers. Obviously the roadway geometry estimates become erroneous when nonmarker objects are captured.

While passing beneath overpasses the LMS often loses track of where the lane marks are located, and the search logic reinitializes itself.

Poor lighting conditions can also lead to false alarms or misses. In general, the LMS is tolerant of illumination conditions that satisfy the original design intent, but for other conditions the following observations have been made during field testing:

Testing during times of significant ambient lighting variation (e.g., cumulus clouds) can lead to loss of lane marker tracking. The camera exposure time is automatically set when CAPC operation is begun, however, as lighting changes due to clouds, shadows, or a change in the travel direction, the exposure may be inappropriate, and image quality may become too poor to locate lane markers.

Operation while traveling east within a few hours after sunrise, or while traveling west before sunset, does not provide adequate lighting conditions.

Nighttime operation is not allowed with the current setup. The difficulty is that the required long exposure time conflicts with the existing framegrabber software.

During rainfall, the system is usually able to find the lane markers, but more outliers are reported by the LMS to CAPC because of puddles, reflections, and

decreased contrast between the roadway and the lane markers. These effects linger until the roadway is largely dry.

The system operates with an increased number of false warnings in very tight curves, such as cloverleaf transitions. The LMS only tracks to 20 or 30m range because the search window logic in the LMS is designed for test tracks or freeway geometries, and does not expect the tight radii. False warnings occur because the limited downrange sights, along with the superelevation, lead the TLC Kalman filter to believe the road to be straighter than it is.

6.0 Conclusions and Recommendations

The first phase of the CAPC project has yielded the somewhat simplified, but operating, prototype of a road-departure warning system. Given the interim nature of this achievement, we have chosen to cite various conclusions in reference to each of the objectives for the overall work. As such, the conclusions constitute a sort of status summary of the progress toward a more complete system. Likewise, the recommendations that accord with the current status are organized as subtasks of a follow-on phase of the project.

Conclusions

- 1) The first objective was to monitor the location and orientation of painted road edges and lane delineators, so as to characterize the layout of the road ahead of the host vehicle. By way of conclusion, the LMS package satisfies this objective under more or less ideal roadway conditions. As such, the LMS serves to enable the study of the lane-keeping/road-departure process in a real vehicle, at highway speeds, on real roads, and even in real traffic. Although a precise ground truth tool is not yet available for assessing the locational accuracy of the LMS subsystem, it appears that the LMS achieves an estimate of the immediate lateral position of the vehicle that is within a few inches of its true location relative to the road edge. Pitch and roll corrections have been seen as a necessary element for obtaining suitable road-finding performance in the far-field, where the assessment of road curvature is important.
- 2) The second objective was to predict the path of the vehicle in near-future time. The on-board transducers of the vehicle's motion response and the steer-input angle appear to support an effective prediction of the vehicle's path. Although the system makes a static extrapolation from the current motion variables as if no future changes in input will occur, test observations suggest that a *dynamic forecast* (where, for example, a future change in the steer input would be anticipated based upon the immediate past history) is unlikely to add value. This expectation is based upon the observation that favorable TLC thresholds for warning tend to be relatively short—i.e., less than 2 seconds into the future. Thus, the impact of the static nature of our path projection is rather small, given that the application scenario is that of a drowsy or otherwise impaired driver whose most likely steering waveform is flat.
- 3) The third objective was to develop a continuous prediction of the future time, TLC, at which the predicted path of the vehicle crosses beyond the outermost travel lane of the roadway ahead. Perhaps the largest single effort was devoted to the processing that yields the TLC measure. The current system provides both TLC projections and their uncertainty. A future stage of development will employ the uncertainty data in the decision making process, using a fuzzy logic controller.
- 4) The fourth objective of monitoring the driver's road-keeping behavior and effecting an assessment of his/her road-following performance is not fully realized within the current

project. Extensive work on the Ford Driving Simulator has yielded measurable differences in driver control quality associated with attention level. These differences should be assessable from data that can be collected on-board the vehicle. It remains to develop the process for characterizing driver state from real-time measurements and implementing this processor within the CAPC host computer.

- 5) The current CAPC prototype does satisfy the fifth objective of providing an audible warning by which to alert the driver to a pending road-edge departure. The warning criteria are fully adjustable using a pull-down menu that is available on the engineer's console inside the vehicle.
- 6) Physical implementation of a control intervention, the sixth objective, is not quite complete. The CAPC prototype does, however, incorporate all of the mechanical hardware for controlling brake pressure at the two rear wheels. Also, the "power electronics" module by which control solenoids would be activated has been designed. The intervention controller software has been developed and exercised extensively using the CAPC simulation tool. Simulation results show that the intervention controller will be able to redirect the vehicle with a level of control authority that is equivalent to one-degree of steer angle applied at the front wheels. The prototype system currently computes the basis for an intervention and signals the driver by a pulsed warning tone to indicate the point at which a physical redirection of the vehicle would have commenced, if intervention hardware were active.
- 7) The seventh objective was to implement a robust decisionmaking function by which the occurrence of false alarms and misses could be properly managed. While the current prototype does implement a number of features which do aid in managing decision errors, the complete decisionmaking module awaits the next phase.
- 8) Since only a warning system is implemented, currently, the prototype system does satisfy the eighth objective—namely, that of retaining a level of control authority for the driver which exceeds that of the provided system at all times. It will remain for full-scale testing of the system, with its intervention system activated, to determine whether the brake-steer feature does indeed leave sufficient control authority for the driver's safe recovery from a near-departure event.

Recommendations

The achievements represented in the fully integrated prototype vehicle suggest that a basically sound approach has been developed for road-departure warning and intervention. Or, more precisely, the functionality of the current prototype provides an effective platform for examining the related issues of intervention mechanics, sensor robustness, and human interaction with the function. We recommend that the first phase of work be followed with an effort that utilizes the platform and expands the base of knowledge for implementing real road-departure prevention products. The following tasks are recommended:

- 1) *A Systems Requirements Document* should be developed as an explicit statement of the CAPC capability to be prototyped in a next phase.
- 2) *Priority Improvements in System Performance* should be undertaken, based upon a listing of candidate enhancements that has emerged from the first phase. Such enhancements should include:
 - a) spot-refinements of the LMS package so as to improve its robustness while still retaining the original architecture and physical elements of this subsystem
 - b) enhancements to the software for determining TLC
 - c) implementation of the brake-steer intervention subsystem
 - d) addition of a driver state-assessment algorithm (with associated new on-board measurements, as needed)
 - e) addition of a fuzzy-logic module for effecting decision making based at least upon TLC, the uncertainty levels in TLC, the vehicle's roadway departure angle, and the driver's state
 - f) provision of the sensing and high level logic needed to extend the CAPC function to left-side road departures, as well as right-side
- 3) *Development of a Ground Truth Tool* should be undertaken as a means of quantitative evaluation of the CAPC prototype's location-finding performance. The Ground Truth Tool would yield data on the precise location of the prototype vehicle, at 10 Hz. The same tool can be implemented to premap a selected test area such as the Ford Proving Grounds, thereby permitting subsequent differential calculations that yield the reference values for vehicle location relative to the road edge lines. Differential GPS is an obvious candidate for this implementation.
- 4) *Evaluation of Systems Improvements* would be effected by means of computer simulation, use of the UM and Ford driving simulators, and by full scale testing of the system on proving grounds (especially when the intervention function is enabled) and on the public highway. The Ground Truth Tool should be employed for obtaining data showing the levels of system performance in quantified, statistically meaningful terms.

7.0 References

References for Section 3.4:

- [1]. MacAdam, C. C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving." *IEEE Transactions on Systems, Man and Cybernetics*. 1981; 11(6); p. 393-399.
- [2]. McRuer, D.; Weir, D. H. "Theory of Manual Vehicular Control." *Ergonomics*. 1969; 12(4); p. 599-633.
- [3]. Modjtahedzadeh, A.; Hess, R. "A Model of Driver Steering Control Behavior for Use in Assessing Vehicle Handling Qualities." *J. of Dynamic Systems, Measurement and Control*. 1993 Sep; 115
- [4]. Knipling, R.; Wierwille, W. "U.S. IVHS Research: Vehicle-Based Drowsy Driver Detection". *Vigilance and Transport Conference*. INRETS/USTRB; 1993 Dec.
- [5]. Thorpe, C.; Hebert, M.; Kanade, T.; Shafer, C. "The New Generation System for the CMU Navlab". *Vision-based Vehicle Guidance*. New York, NY: Springer-Verlag; 1992: 30-82. (Masaki, I. ed. Springer Series in Perception Engineering).

References for Section 3.5

- [1] Kalman, R.E., Bucy, R.S., "New Results in Linear Filtering and Prediction Theory," *J. Basic Eng.*, Trans. ASME, Ser.D. pp.95-108.
- [2] Kwakernaak, H., Sivan, R., *Linear Optimal Control Systems*, Wiley-Interscience, 1972.

References for Section 3.7

- [1] Riede, P.M., Leffert, R.L., Cobb, W.A., *Typical Vehicle Parameters for Dynamics Studies Revised for the 1980's*. SAE 840561.

8.0 Prototype Operating Instructions & Software Descriptions

8.1 Safety Assessment Report

The System

From the viewpoint of hazard analysis, the CAPC prototype constitutes a conventional passenger car driven by a human operator on a paved roadway, with the companion assistance of an instrumentation system that couples electronically with the brake system. The analysis, below, is based upon the assumption that hazards that are associated with any of the conventional aspects of this commercially available passenger car require no special consideration here. Accordingly, the entire scope of any special CAPC hazards pertain to the system's electronic connection with the rear-wheel brakes. Insofar as this feature has not been implemented in the current CAPC prototype, we flatly declare that no special safety issues are posed by this system. In anticipation of the brake-steer implementation, however, this safety discussion has been prepared.

In particular, it is recognized that inadvertent actuation of the electronically controlled brakes could pose a possible loss of control of the vehicle, especially in the case in which both rear wheels would be applied to the full-torque condition at the same time as another malfunction caused the antilock system to fail, thus causing a spinout. A lesser, but perhaps still alarming disturbance would ensue if only a single brake was inadvertently applied to the full-torque condition, thus inducing a temporarily curved path until the driver responded with recovery steering or foot-pedal braking.

In order to circumvent either of these modes of failure when CAPC's full functionality is not being tested, the future extension of the prototype system will provide a master "Brake Shut-Off" switch on the dashboard, with a red "Brakes Hot" alert lamp. The switch is wired directly into the circuit branch that applies control signals to the electronic brakes. By this arrangement, the state of enablement of the electronic control circuits is continuously displayed to the driver and the circuits can be disabled whenever the test driver chooses. The "Brake Shut-Off" switch is a palm-operated button that latches open when pressed, thus providing a panic-button functionality.

An additional feature of the control-circuit design will provide that any application of the brake pedal by the test driver, thereby closing the brake lamp switch, causes the entire electronic brake actuation circuit to be disabled. Thus, whenever the CAPC prototype is being subjected to "full-function" testing, with electronic brakes enabled, any inadvertent brake actuation due to an electronic anomaly can be readily overridden by means of the natural, manual braking reaction of the test driver.

Aside from brake system issues, the CAPC prototype is equipped with a benign instrumentation package that is transparent to the driver and which presents nothing unusual to the driving environment. The instrument system is powered by a 110 VAC inverter located in the trunk of the vehicle. No possibility of electrical shock to the driver, from this system, exists.

Given the provision of an electronic brake disabling switch, the means for minimizing hazards during our test program are procedural, as outlined below.

All Testing

Under all conditions of CAPC testing, use of the prototype vehicle will be under the supervision of the CAPC project director or a designated test engineer. No other use of the vehicle will be permitted. The driver of the vehicle will either be a UM or Ford employee authorized by the test engineer or another individual permitted to drive during demonstration testing, with the test engineer accompanying.

When the future CAPC prototype is operated on a public highway, the electronic brake system will always be disabled. Thus, no full-function testing of CAPC on public highways will be done. This provision is discussed under "On-Highway Testing," below.

Full-function testing of CAPC, with the electronic brake system active, will only be done on private proving grounds facilities under conditions which give satisfactory margin for recovery by the test driver, in the event that inadvertent actuation of the brakes should occur. This provision is discussed under "Proving Grounds Testing." below.

On-Highway Testing

Whenever the CAPC prototype vehicle is to be driven on the public roadway, either for the purpose of collecting image data or simply driving to and from a private proving grounds area, the "Brake Shut-Off" switch will be engaged such that electronic brakes are disabled. In this mode of operation, the brake system behavior is identical to that of the unmodified, production version of Ford's Taurus SHO. Accordingly, hazards pertaining to inadvertent brake application in the public highway environment will be eliminated by a procedural admonition that electronics brakes never be enabled on a public roadway.

Proving Grounds Testing

Full-function testing of CAPC, on private proving grounds facilities, will require that the electronic brake system be active. Accordingly, the following procedural steps will be taken to ensure that no significant test hazards are posed by the possibility of inadvertent (or simply unanticipated) actuation of the electronic brake system:

- 1) The "Brake Shut-Off" switch will be engaged (i.e., system disabled) whenever the test vehicle is being operated in any possibly hazardous proximity to other test vehicles (such as in ingressing or egressing from the test track for full-function testing.)
- 2) The vehicle will only be operated in the full-function mode in test lanes which give at least one-lane clearance from guardrails or other fixed objects. At Dana and Ford Proving Grounds, the outermost lanes will not be used for CAPC testing as they lie adjacent to guardrails. At TACOM's track, we will not operate immediately adjacent to the east retaining wall.
- 3) The test driver will be instructed that a simple tapping of the brake pedal will disable the electronic braking function, thus arresting any inadvertent or unanticipated actuation.
- 4) Full-function testing will begin at lower speeds, which afford a simpler context for manual recovery from automatic braking interventions. Only after the driver has gained experience with the intended brake interventions at low speeds will speed be elevated to the level of highway operations.

8.2 CAPC Prototype Vehicle System Operating Instructions

Basic Operation

The CAPC Prototype can function basically turn-key except for a few simple steps. The basic procedure is outlined below in this section.

Step 1. Power up systems

- With vehicle parked *and running*, turn on inverter and AC power strip - NEVER run the inverter with the vehicle off. All systems but the Quadra will come on line.
- Press the power-on button on the Quadra computer keyboard. The MacDAS software will automatically load and begin initialization.

Step 2. Run Controller

- select the "Controller..." option from the bottom of the "MIO-16" menu. The data window will appear and signals should be displayed in the plots.

Step 3. Zero References

- Zero references for steer, yaw, and acceleration MUST be grabbed at the start using the ⌘-Z key combination. One MUST be driving straight down a level road, holding the steering wheel as straight as possible. It is recommended that new zeros be grabbed every hour or when driving conditions change significantly.

Step 4. Go / Kill

- The CAPC function can be started and stopped using the "Go/Kill" button. The system should only be started ("Go") when centered in a lane on a straight section of roadway so that the LMS can find the lane markers. The function may be killed at any time with the "Kill" button.

Step 5. Finished - with the vehicle STILL RUNNING,

- Close the Controller window (via the "Done" button) and Quit MacDAS (via the "Quit" selection under the "File" menu). Select "Shut Down" from the "Special" menu under the Finder.
- Power-down the AC power strip.
- Turn off the inverter.

Additional steps for Data Collection

The CAPC Prototype can collect a limited amount of data during running. If it is desirable to collect analog and controller data, the system's calibration should be updated prior to beginning operation. The following steps should be inserted in the above list where applicable.

Step 1a. Calibrate Transducers

- Place the vehicle on a level roadway and steer exactly straight ahead, with the nominal load in place (passengers, etc.). Sit **VERY** still for the calibration.
- Select 'Calibrate Cards...' from the "**MIO-16**" menu.
- In the next dialog, select the 'Cal' button.
- In the last dialog, the calibration will take place and the results are displayed on the screen. The channel printed in italics is the one currently being calibrated. Do **NOT** move during calibration. Allow **ALL** channels to be calibrated.
- When the last channel is finished, select the 'Done' button at the lower right corner of the calibration window.

This calibration will remain valid until another setup file is loaded or the application is terminated. Calibration of the LMS system should only be done by qualified personnel.

Step 4a. Go / Kill / Save

- In order to save data, the 'Kill' button must be selected **BEFORE** 95 percent of the buffer is filled (via the Ok, 0% indicator), otherwise a buffer-overflow error will occur. Alternatively, the 'Autosave' box can be checked.
- Once Killed, the data for that run can be saved by selecting the 'Save' button. Selecting 'Go' again will clear the data spooler and restart data acquisition; data will not be saved for the previous run.
- Data will be saved in the 'DataFiles' folder as an ERD file with the name indicated in the file display (0001:0), and the file number automatically incremented.

8.3 CAPC Software Systems Manual

8.3.1 CAPC Prototype Vehicle Integrated User/Programmer's Manual

Introduction

The CAPC prototype functionality is defined by the many subsystems and their interactions. The primary control logic of CAPC resides within the control loop of UMTRI's MacDAS data acquisition system. This software is an infrastructure for controllers and data acquisition, providing the control system with analog and digital I/O, serial I/O, graphics, real-time loop management, user interfaces, and standard ERD file I/O. The MacDAS system also controls the analog rack and shields the control algorithms from hardware upgrades.

The software provides a variety of menu options for working with the analog rack and transducers, automated calibrations, digital I/O and analog outputs. [Also are options for sampling and channel setups.] The controller-specific portion of the system (in this case the controller is CAPC) provides a few simple menus and user interaction dialogs.

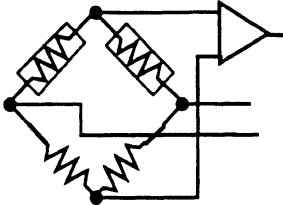
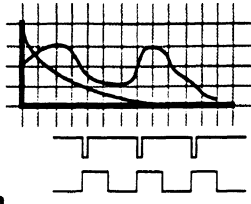
The basic program layout is shown below:

- ◆ **Data Files:**
 - read & write ERD data files
 - initialize analog and digital sections; internal gain & offset tables
 - auto-sequencing of data file numbers for easy data collection
- ◆ **Hardware:**
 - complete channel setup control including gain, offset, filter frequent
 - automated calibration system for input channels
 - analog and digital I/O exercise capability
 - automated calibration capability for analog outputs
 - specialized rack interaction commands and dialogs
- ◆ **Data Acquisition:**
 - automated data collection and storage
 - variety of real-time graphics displays
 - derived channel capability (linear combinations of other channels)
- ◆ **Controller:**
 - real-time interface with MacDAS for data transfers
 - hooks into MacDAS loop timing for time-dependent controllers
 - hooks into MacDAS control loop for state-machines
 - controllers can modify the data acquisition process in real-time
 - internal data generated by controllers is collected and saved in the ERD file

During program startup and initialization the following screen will appear:

Macintosh Data Acquisition & Control System

Mac
DAS 2.0

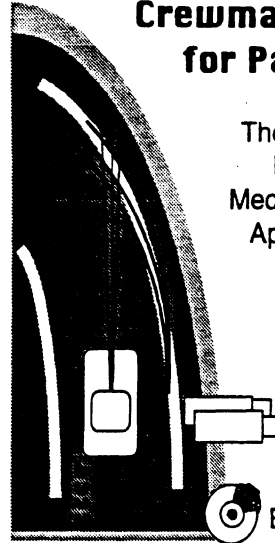
whoblame
G.E. Johnson
K.A. O'Malley
M.S. Campbell

**The University of
Michigan Transportation
Research Institute**

- C.A.P.C. Controller - Crewman's Associate for Path Control

The U of M Transportation
Research Institute &
Mechanical Engineering and
Applied Mechanics Dept.

whoblame
P.J.Th. Venhovens
G.E. Johnson
D.L. Leblanc



Vision system by E.R.I.M.
Brake-Steer by U.M., Ford

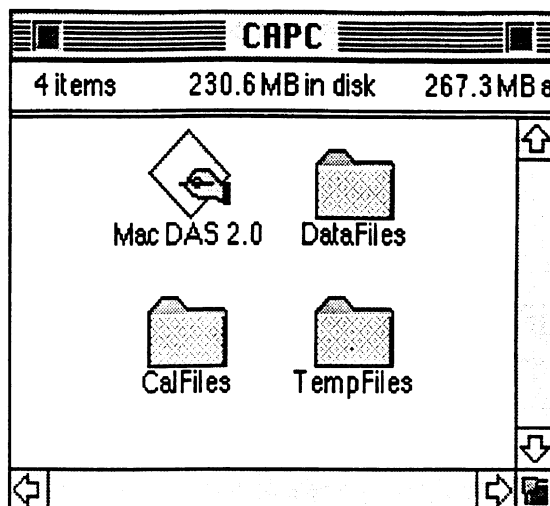
Copyright © 1993-1995 The Regents of The University of Michigan;
parts by National Instr., GW Instr., Symantec, and Apple Computer 🍏

Figure 8.3.1.1. CAPC / MacDAS start-up screen.

The CAPC prototype software begins execution by loading setup files and initializing all hardware. This includes locating and initializing the analog rack, the LMS sensor, and the brake controller. Also a default channel setup for CAPC is loaded and the program placed in the standard Macintosh menu loop. In this mode the software is interactive, allowing calibrations, I/O exercising, and system setup editing. The two other distinct modes of operation are either *Data Collection* or *Controller*. In both modes the system switches to a real-time graphical display and user interface. Specific data channels are displayed on the monitor as well as a variety of status information. The user can start and stop the operations and save collected data. The distinction between the two modes is that *Data Collection* simply collects the data specified in the channel setup, while *Controller* not only collects data but makes calls to the controller loop and stores any internally generated data. Details on these and other modes follow in the menu descriptions below.

Software Description

The PROGRAM folders



There are four items in the CAPC folder required for execution of the prototype. The application MacDAS 2.0 contains the compiled MacDAS code and the CAPC code. The folder "DataFiles" is the default location to store all data files generated by MacDAS. This also contains the default setup file named '0000'. The folder CalFiles contains data files generated during calibration, and the folder "TempFiles" contains temporary dump files generated by the data collection system. These last two need not be of concern for the casual user.

The File menu

File	
Save Channel Setup...	⌘S
New Channel Setup...	⌘N
Current File Number...	
Quit	

This menu provides for loading and saving primary setup files. It also allows for the changing of data file numbers and termination of the program:

- *Save Channel Setup*

The current channel setup (names, units, gains, offsets, calibrations, etc.) may be saved. ALL data in the setup is saved into an ERD header file, including disabled channel information. Note that the file '0000' is loaded at startup;

saving a new setup file called '0000' to replace the old one will provide for a new default setup at startup.

- *New Channel Setup*

The current channel setup may be overridden by another ERD file. This allows a previous data file to be loaded and used as the setup for current tests.

- *Current File Number*

This will allow the user to change the current ERD file number. During data collection, data files are numbered sequentially like 0014, 0015, etc. This option sets the start number for the next set of data files. Valid values range from 1 to 9999.

- *Quit*

This will terminate the application, freeing memory and serial ports.

The *Edit* menu

Edit	
Undo	⌘Z
Cut	⌘C
Copy	⌘B
Paste	⌘B
Clear	⌘B
Preferences...	

This menu is currently unused but included for completeness.

The MIO-16 menu

MIO-16	
Edit Chan. Setup... ⌘E	
Sampling Setup...	
Calibrate Cards...	
Calibrate D/As ▶	D/A Channel 0
	D/A Channel 1
	D/A Channel 2
	D/A Channel 3
Analog Inputs...	
Analog Outputs...	
Digital I/O...	
Amplifier Cards ▶	Set Offset D/As...
Record New Zeros	Set Cal Relay...
Restore Analog	Shunt Cal Relays
Init. Blue Earth ⌘I	
Collect Data... ⌘D	Control Line 1
Controller... ⌘R	Control Line 2
	Filter Freq's...

This menu is the primary interface to the MacDAS system. It is termed 'MIO-16' since that is the name of the multi-function I/O board installed for MacDAS. Each item is discussed below.

- *Edit Chan. Setup*

The current channel setup can be modified, including names, gains, offsets, calibrations, etc. Details of channel editing and parameters can be found in the *MacDAS Technical Reference*. Modification is *not* recommended for the casual user.

- *Sampling Setup*

The sampling rate, controller rate, buffer size, and miscellaneous parameters can be modified here. The impact of changing each parameter is described in detail in the *MacDAS Technical Reference*. Modification is *not* recommended for the casual user.

- *Calibrate Cards*

The cards in the analog rack require periodic calibration and adjustment. When the system starts up the default cal information is loaded from '0000'. This function will recalibrate the analog cards and automatically rezero drifted transducers or circuitry. This function also provides a basic "health-check" on amplifiers and transducers. This operation should be performed whenever data is to be collected or if a transducer performs poorly.

- *Calibrate D/As*

This function is not utilized in CAPC.

- *Analog Inputs*

This function allows the user to view a range of analog input channel values. The raw data on the inputs is displayed as A/D units, voltage, and engineering units.

- *Analog Outputs*

This function is not utilized in CAPC.

- *Digital I/O*

This allows the exercising of the digital outputs and monitoring of inputs. For example the warning buzzer and the intervention alert can be energized independently from this control panel.

- *Amplifier Cards*

This function can be found in the *MacDAS Technical Reference*.

- *Record New Zeros*

This function will measure the current voltage on each channel and record the values as a new zero for each channel.

- *Restore analog*

This function can be found in the *MacDAS Technical Reference*.

- *Init .BlueEarth*

This function can be found in the *MacDAS Technical Reference*.

- *Data Collection*

This allows the user to collect data on the channels specified in the setup files (default '0000'). Data is saved in ERD files using the sequenced file number as a name. Details on this function can be found below under 'Operation' and in the *MacDAS Technical Reference*.

- *Controller*

This allows the user to execute the embedded control function (CAPC) while collecting data. The controller is called as part of the timing loop, and data is shared between MacDAS and CAPC. Internal data generated by CAPC is saved by the MacDAS system as part of the normal data stream. Data is saved in ERD files using the sequenced file number as a name. Details on this function can be found below under *Operational Notes* below and in the *MacDAS Technical Reference*.

The CAPC menu

CAPC
LMS Tools Brake Tools
Brake Enabled Reset
TLC Thresholds ...

This menu is dedicated to controller-specific functions.

- *LMS Tools*

This provides a valuable interface to the LMS system for initialization, debugging, development, and demonstration. It provides access to all defined communication packets, system calibration, and will exercise the LMS system in real-time while displaying returned lane geometry. It can also send user-specified pitch, roll, and velocity values to the LMS system. Details of this tool can be found under *Operational Notes* below.

- *Brake Tools*

Function not currently implemented.

- *Brake Enabled*

Function not currently implemented.

- *Reset*

Function not currently implemented.

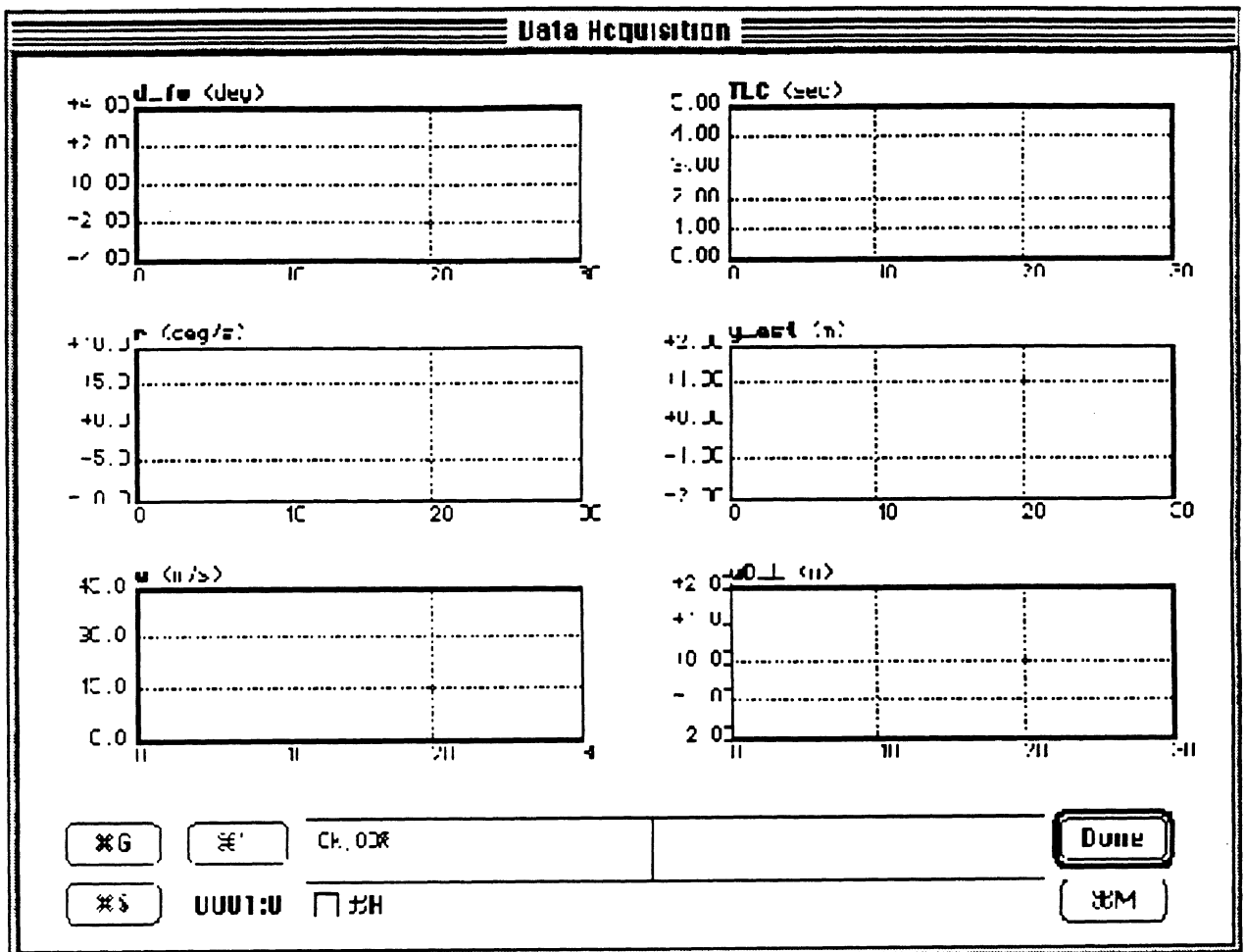
- *TLC Thresholds*

Three parameters that determine when the warning and intervention buzzers are activated can be changed through this selection. A pop-up dialog box appears; details can be found below under *Operational Notes* .

Operational Notes

The Collect Data & Controller window

Below is the window that appears during data collection or controller exercising. Six plotting windows are indicated showing a variety of signals. The amplitude of each parameter is plotted on the vertical axis against time on the horizontal axis.



This window shows a number of specific functions, each of which is outlined below.

- *Go / Kill*

This is the primary control button for all system users. When selecting "Go", the hardware starts collecting data into a circular buffer. Also, the controller will receive a 'start' command and will be called in a timely manner thereafter. The button name will change to 'Kill'; this is to stop the controller and data collection and hardware clocks. Note that when in the control mode, pressing

'Go' causes the LMS system to receive a START packet and begin looking for the lane lines in the center of the image. The vehicle should be centered on a straight section of road at this time. 'Kill' can be issued at any time and will also send a STOP packet to the LMS system.

- *Pause / Cont.*

This function simply halts all clocks and data acquisition cycles. This is intended for system level diagnostics - data channel phasing will be corrupted by use of this option. This is intended primarily for state-machine operation and is not recommended for use in the CAPC function.

- *Save*

This will save the data in the spooled buffers as an ERD file. This save may take several seconds, depending on the size of the data file. All controller activity is stopped during a save and the controller must be restarted after completion.

- *Autosave*

When checked, the system will automatically save the data buffer when it reaches 95 percent full. Note that the controller is automatically stopped when saving data.

- *Done*

This will leave the data acquisition or control window.

- *Matrix*

This is not implemented in the CAPC prototype

- *0001:0*

This indicates the current file number that the presently acquired data will be saved under. The :0 is an indication of spools made to the hard disk, not utilized by the CAPC controller.

- *Ok, 0%*

This shows an estimate of the amount of data that has been collected (in kbytes) and the amount of buffer space being consumed (in %).

- *hidden keys*


There are a number of functions that were not given buttons on the screen, rather they can be accessed via hidden key strokes. The table below outlines each function and its appropriate command-key combination.

Esc	Abort	Abort test controller & I/O
⌘C	Clear	Clear spool files (0001:N -> 0001:0)
⌘Z	Zero	Record zeros for steer, yaw rate, Ay
⌘I	Init	Initialize collection to startup state
⌘H	Reset	Reset circular buffer ptrs. (N%-> 0%)
⌘T	Thresh.	CAPC Threshold Pop-up Menu

- *CAPC Threshold Menu*

The CAPC function uses three thresholds in deciding when to issue warnings or to indicate when interventions would take place. The following menu can only be accessed when the controller is not actively running (i.e., by quitting the *Data Acquisition & Controller* menu). The menu is accessed by choosing "TLC Thresholds ..." from the CAPC menu. The user adjusts the TLC values at which warnings and "interventions" are triggered, as well as shifting the lateral location of the lane markers. Default values are embedded in the control code.

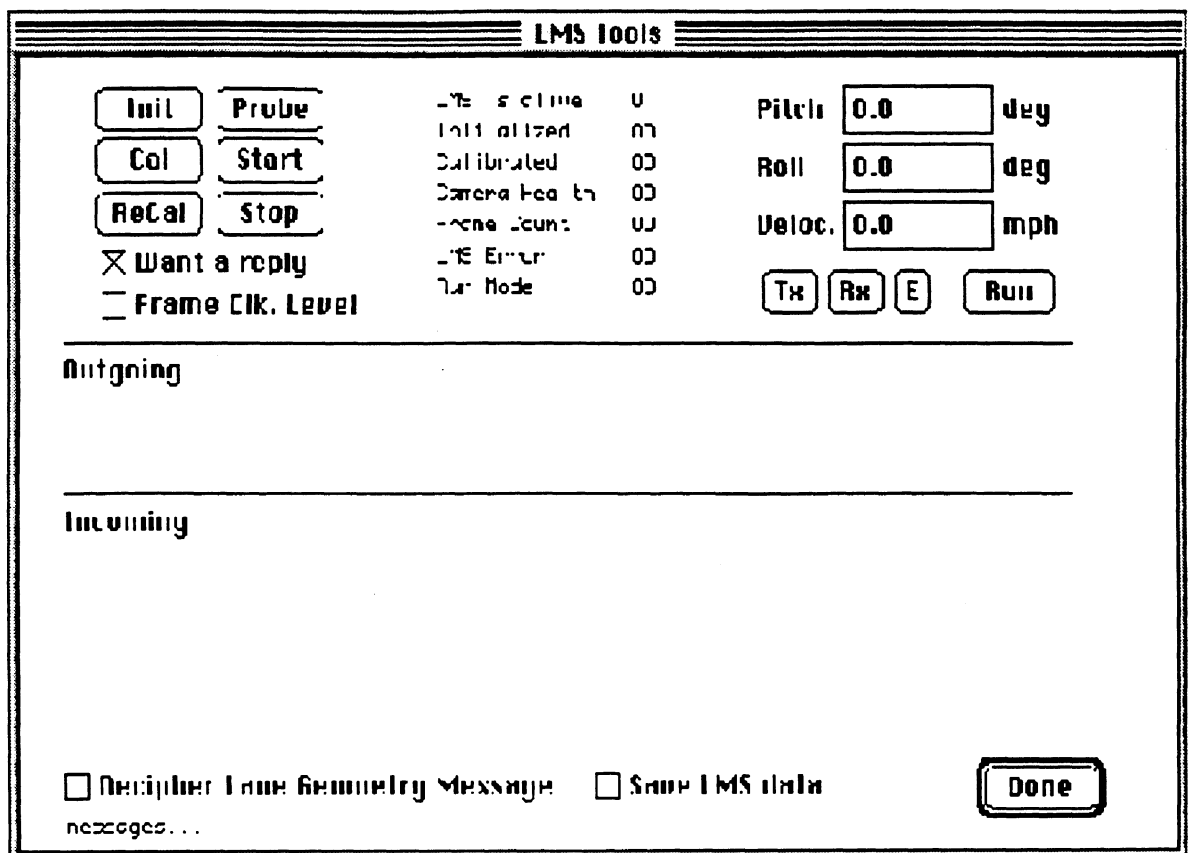
TLC warning threshold	<input type="text" value="2.0"/>	sec
TLC intervention threshold	<input type="text" value="1.0"/>	sec
Extra lateral offset	<input type="text" value="0.0"/>	m
<input type="button" value="Default"/>		<input type="button" value="OK"/>



The *LMS Tools* window

Below is the window for interfacing with the LMS system in a development stage. The tool is provided to allow high-level access to packet transmissions but provide low-level information on data streams. It also contains a controller simulation which can exercise the LMS system in real time and display and collect returned lane geometry.

The top third is dedicated to user controls, LMS status display, and parameter adjustment capabilities. The middle section (Outgoing) displays the actual data in hexadecimal that was transmitted to the LMS system. The lower section (Incoming) displays replies from the LMS system as either packets (in hexadecimal) or in lane-geometry engineering units (x,y coordinate pairs for data points as well as lane descriptors).



This window shows a number of specific functions, each of which is outlined below.

- *Init, Probe, Cal, ReCal, Start, Stop*

Selecting any one of these buttons will send the specified packet to the LMS system. Any replies may be displayed on the Incoming section.

- *Tx, Rx, E, Run*

These provide serial port control functions. 'Tx' will transmit a single pitch/roll/veloc packet to the LMS system. 'Rx' will check the Quadra's serial

buffer to see if it has received any lane geometry. 'E' will empty the Quadra's buffer (allows for setting the buffer to a known state). The 'Run' button will begin a nominal 10Hz control loop. There is no CAPC function being executed, but analog channels are being sampled, the LMS system is being clocked, and lane geometry is being processed and displayed on the screen. This is primarily a test of the communication systems and data validation. Note that holding down the Option key while pressing Run will cause the system to run at about half speed, or 5Hz (useful for debugging & testing communications).

- *Pitch, Roll, Veloc*

These edit boxes allow the user to enter values for parameters to be sent to the LMS system using the 'Tx' button. When in 'Run' mode, these boxes display the actual transducer values which are being sent to the LMS system.

- *Done*

This will leave the LMS Tools window.

8.3.2. Software Flowcharts and System Interface Requirements

MacDAS / CAPC Controller

The CAPC prototype control code is executed under UMTRI's MacDAS real-time data acquisition and control system. Note in the following diagram that the CAPC function has been broken into two halves. The first half is responsible for interacting with the transducers and generating data for the controller to use. The second half deals with the LMS data arriving (see timing diagram following the flow chart) and the execution of the actual CAPC control code.

The MacDAS system functions in essentially two distinct modes. The first is a simple display mode which is the default when 'Controller...' is launched. This mode simply displays the values of the analog channels and any other information programmed into the display routine. This is *Not* a data collection mode, nor is the controller being executed. When 'Go' is selected, the real-time loop management begins by starting internal clocks and timers. The CAPC function itself will also be told that it is expected to begin functioning, allowing for initial states to be set.

Data is sampled during this phase at will, without any interest in controlled timing. This allows for smooth display of information and also allows the Macintosh to 'borrow' the processor for other applications which may be running under the Finder. In the data acquisition or run mode, the timing of data is strictly controlled via hardware interrupts to achieve the desired sample rate; all other processes are suspended. When data is required for display or control, the most recently collected samples are copied from the acquisition buffer into the controller's variable space.

During the running mode, the CAPC function is called at a fixed sample rate. In the case of CAPC the controller executes at 10Hz. Also during the run mode, the LMS system receives Transister-Transister Logic (TTL) signals for triggering the camera and vehicle state data packets. The TTL signals are synchronized to the analog data stream and the CAPC function calls.

Real-time loop management for the controller is achieved using hardware timers and a polling approach, rather than an interrupt-based system as the analog sampling does. This increases the overhead but allows for the controller to overrun its allotted time limit without causing system failures. If the control function or other sequence times out (such as waiting for lane geometry) the loop manager is able to resynchronize itself without interfering with the control loop timing (i.e., time overruns will *not* accumulate).

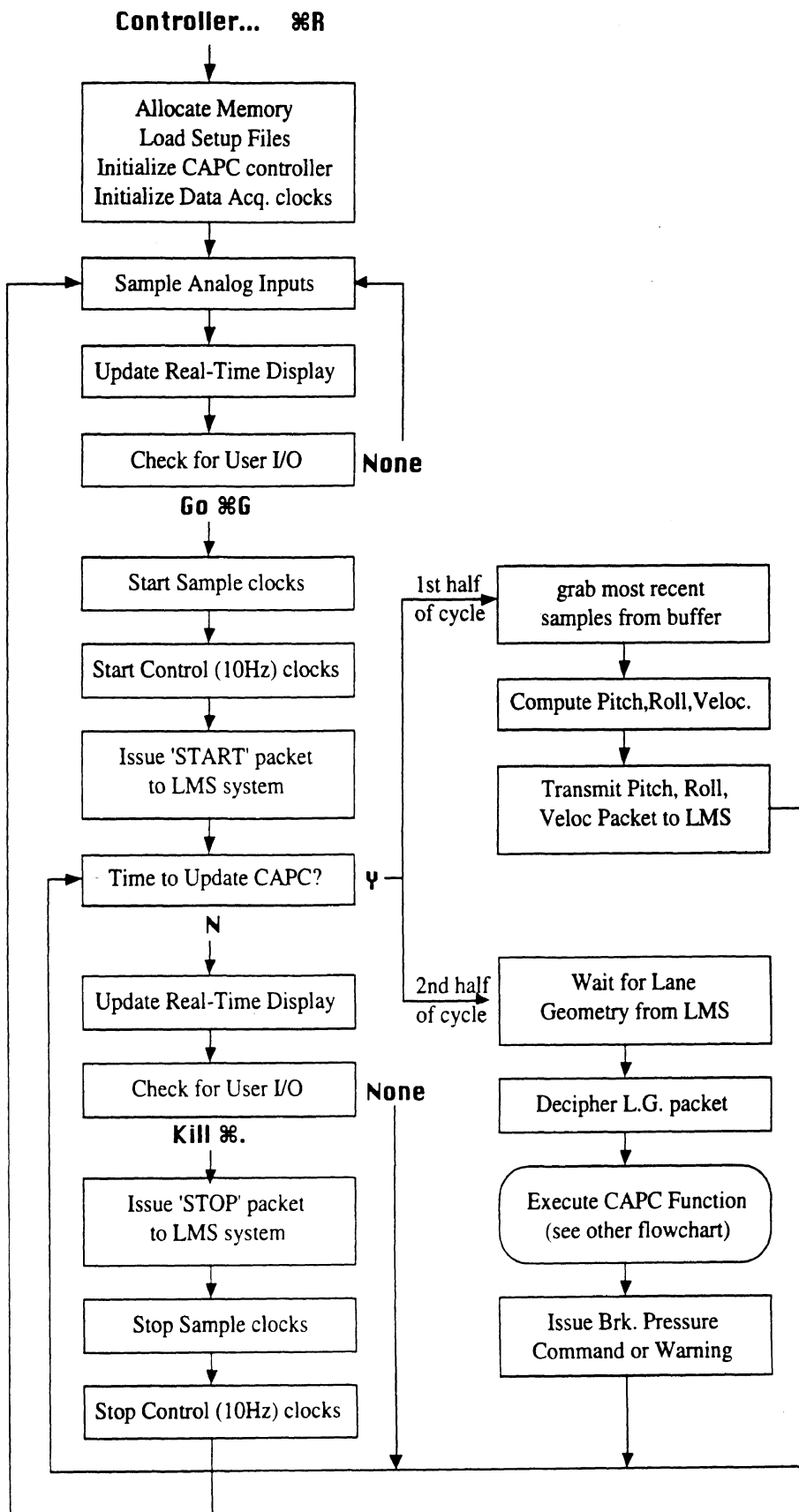


Figure 8.3.2.1. MacDAS / CAPC flow diagram

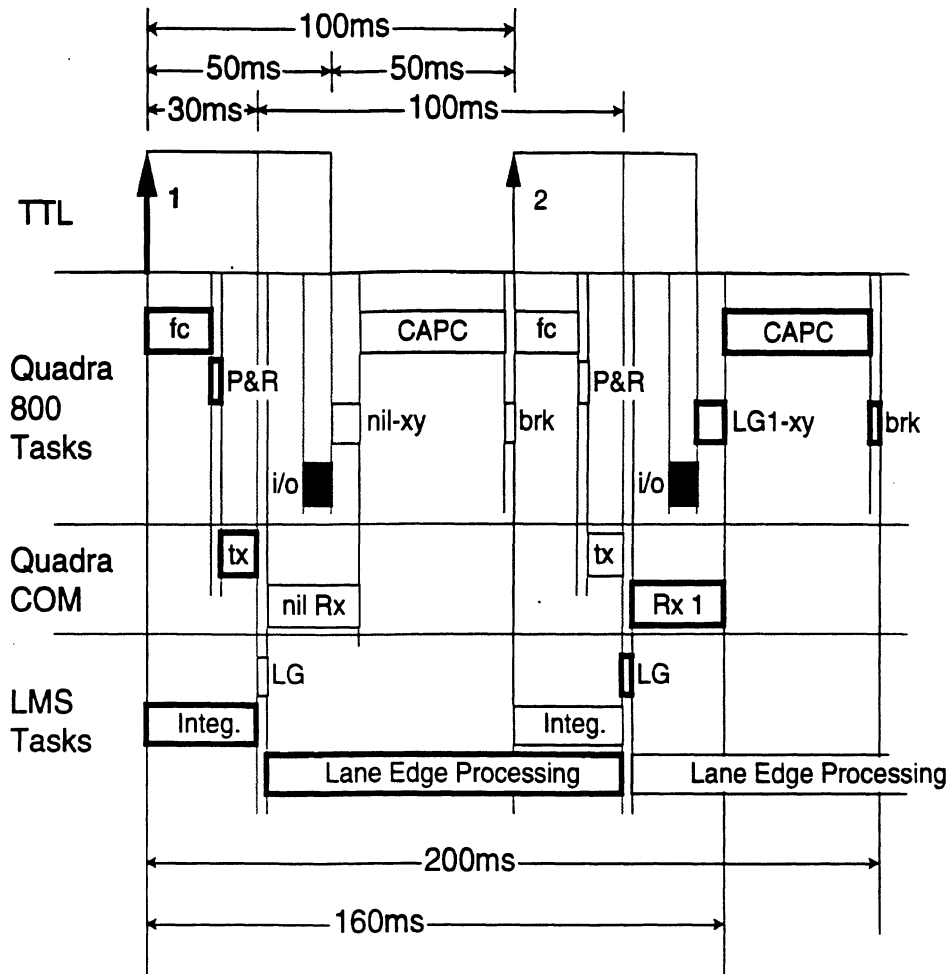


Figure 8.3.2.2 MacDAS / CAPC / LMS timing diagram

The above timing indicates the split in cycle halves for the controller. Each 50ms interval represents half of the allotted 100ms interval for the 10Hz control cycle. In the first half, data is sampled from the buffer (fc), pitch and roll is computed (P&R) and transmitted to the LMS system (tx). Also at the start of the first half the LMS system begins integration of the video image (integ), receives the pitch and transmits the lane geometry that was computed (LG) in the previous cycle.

The blocks outlined in bold lines indicate the overall path to complete one entire cycle of the CAPC function, including communication and computations on all platforms.

CAPC Function Flowchart

Figure 8.3.2.3 shows the flow of the CAPC function itself. The CAPC function is begun by hitting the 'Go' button in the *Data Acquisition & Controller* window, as described above. CAPC initializes itself as follows: first CAPC waits for LMS data to arrive, then checks whether the vehicle speed is within the allowed bounds. If it is, then an initial least squares fit is used to estimate roadway geometry from the first image, and this is used as initial conditions for the Kalman filters.

Once initialization takes place, the system goes into a 10 Hz loop which lasts until either the 'Kill' or 'Save' button are pressed in the *Controller* window. The loop consists of: vehicle path prediction; near-range Kalman filtering; far-range Kalman filtering; decision-making; and possibly activating warning and/or intervention devices. Variables are passed to the MacDas function for storage. Details of the code itself follow in Section 8.3.3.

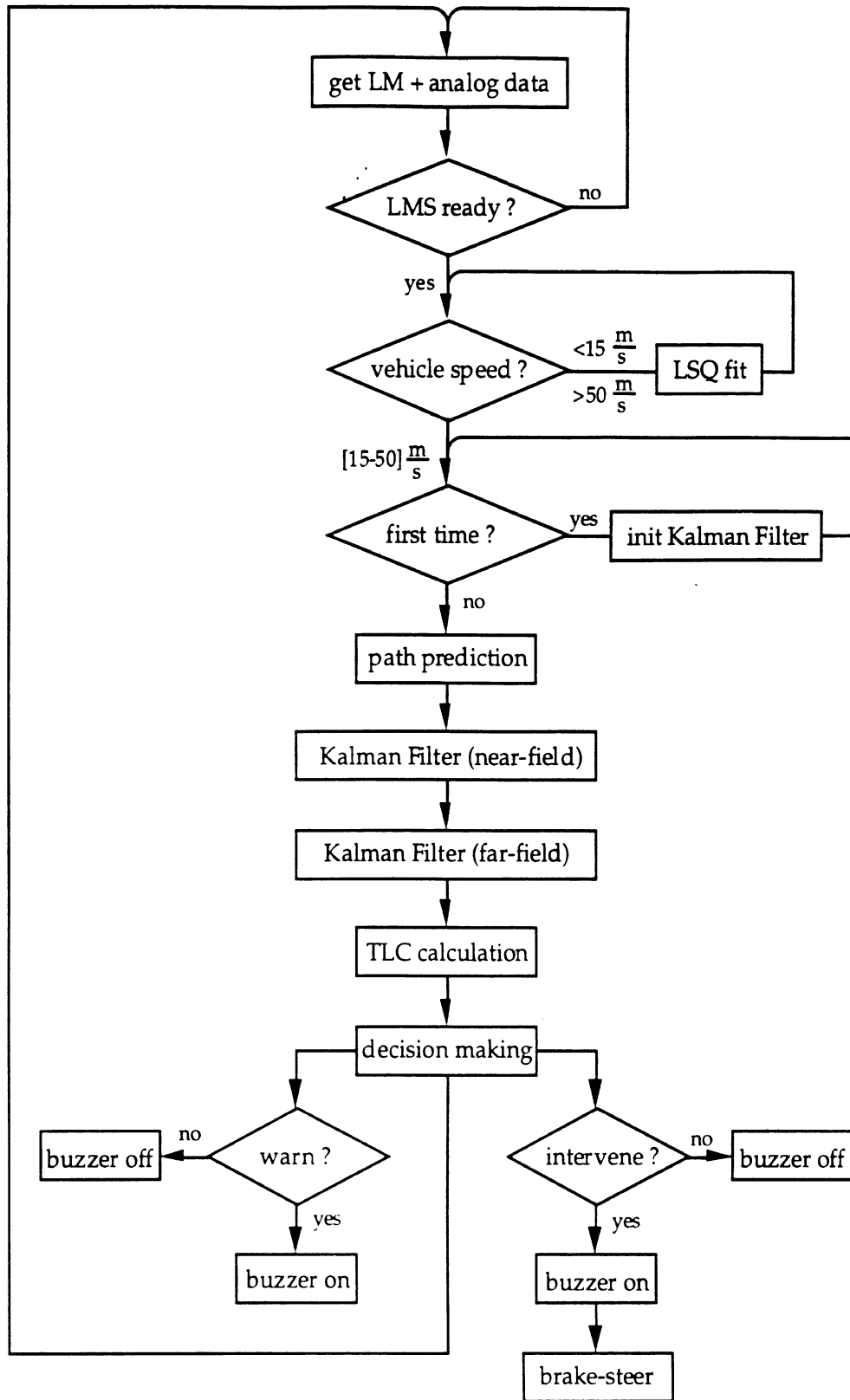


Figure 8.3.2.3: CAPC function flowchart

LMS Flowchart

The Lane-Mark Sensor (LMS) System is a turnkey system which interfaces to the CAPC Controller via a serial communications connection. The operation of the system is summarized in the flowchart in Figure 8.3.2.4.

The LMS system has two distinct modes of operation. Before a START command is given from the CAPC controller, the system is in a wait state where no data is sent back and forth to/from the CAPC controller. Once the system receives INIT and START command packets, it sets the exposure of the camera, and starts sending lane data, corrected for the pitch and roll of the vehicle from the CAPC controller, to the CAPC controller. [The timing of this transaction is driven by transitions of the Frame Clock triggers the LMS camera to capture an image.] This image is automatically transferred to the LMS computer, and upon completion of the transfer, the "Frame Ready" flag is set. It is this flag that the LMS code uses to send old lane geometry to the CAPC controller, and process the newly captured image. A detailed explanation of the communications can be found in the Communication Interface Specification, ERIM document #2498003, which is included as Appendix B.

The LMS Processor communicates with the remainder of the CAPC system via an RS-232 Interface. The internal timing of the LMS system is synchronized with a Frame Clock from the CAPC controller. A detailed explanation of the Electrical Interface can be found in the Lane-Mark-Sensor Mechanical & Electrical Interface Specification, ERIM document #2498001. This is Appendix C in this report.

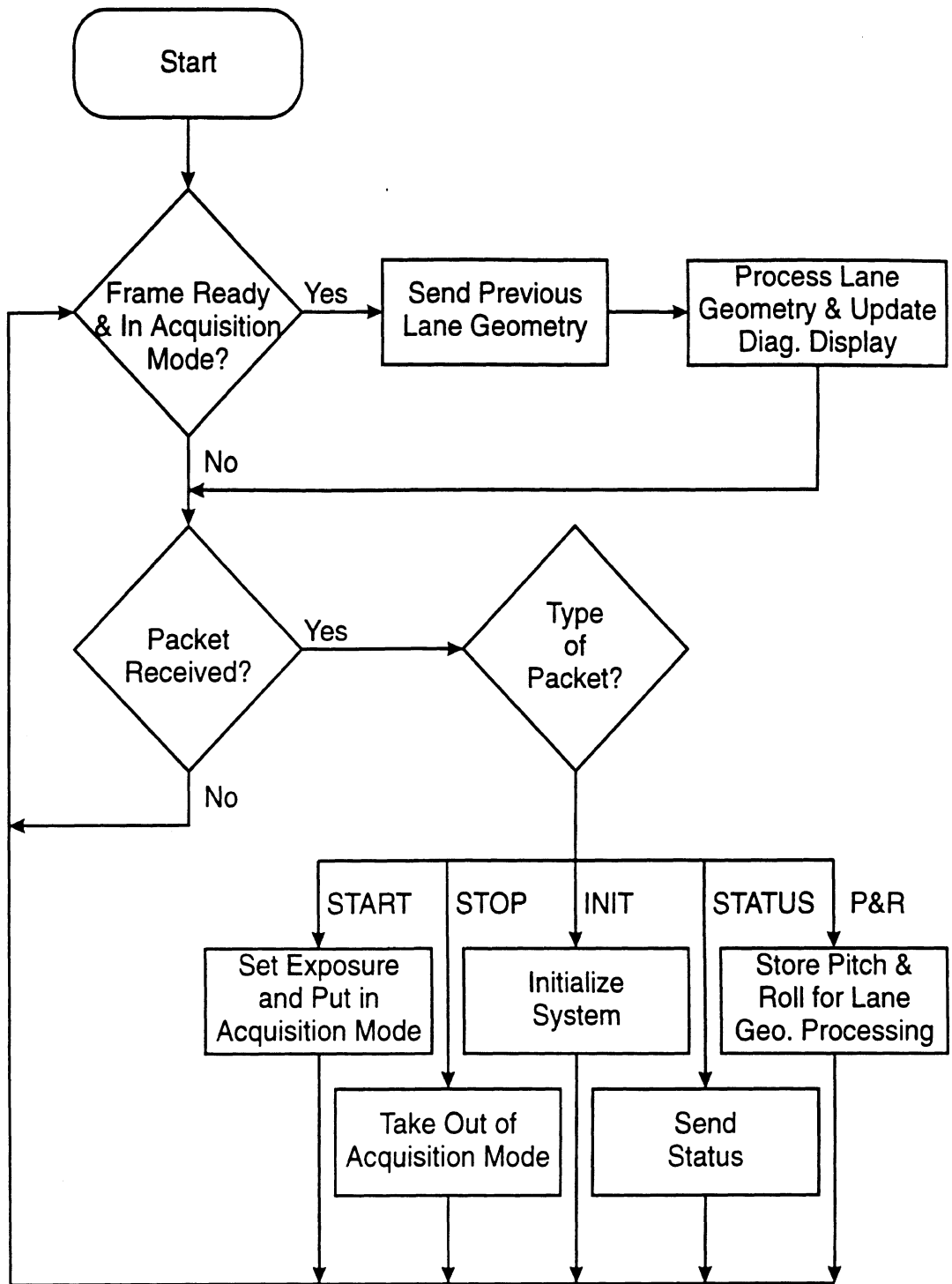


Figure 8.3.2.4: Flowchart of LMS processing

8.3.3. CAPC Prototype Software Code

This section addresses the CAPC control code onboard the prototype vehicle. The CAPC control code is that part of the code which is specific to the CAPC function. It does not include the generic MacDas code described in previous sections, nor the LMS computer code, also described and listed elsewhere. The CAPC control code is limited to those operations which take the digitized analog transducer data and the LMS data and compute decisions to either activate or leave silent the warning and intervention actuators.

In this section the code structure is reviewed, variables are identified, operations to compute vehicle speed, roll and pitch are presented, and finally a list describing the code functions is presented. A listing of the CAPC control code itself is included as an additional appendix.

CAPC control code structure

The structure of the CAPC algorithms was explained in Section 3, and the flowchart presented in Section 8.3.2. The core of the algorithm with its inputs and outputs is depicted again in Figure 8.3.3.1.

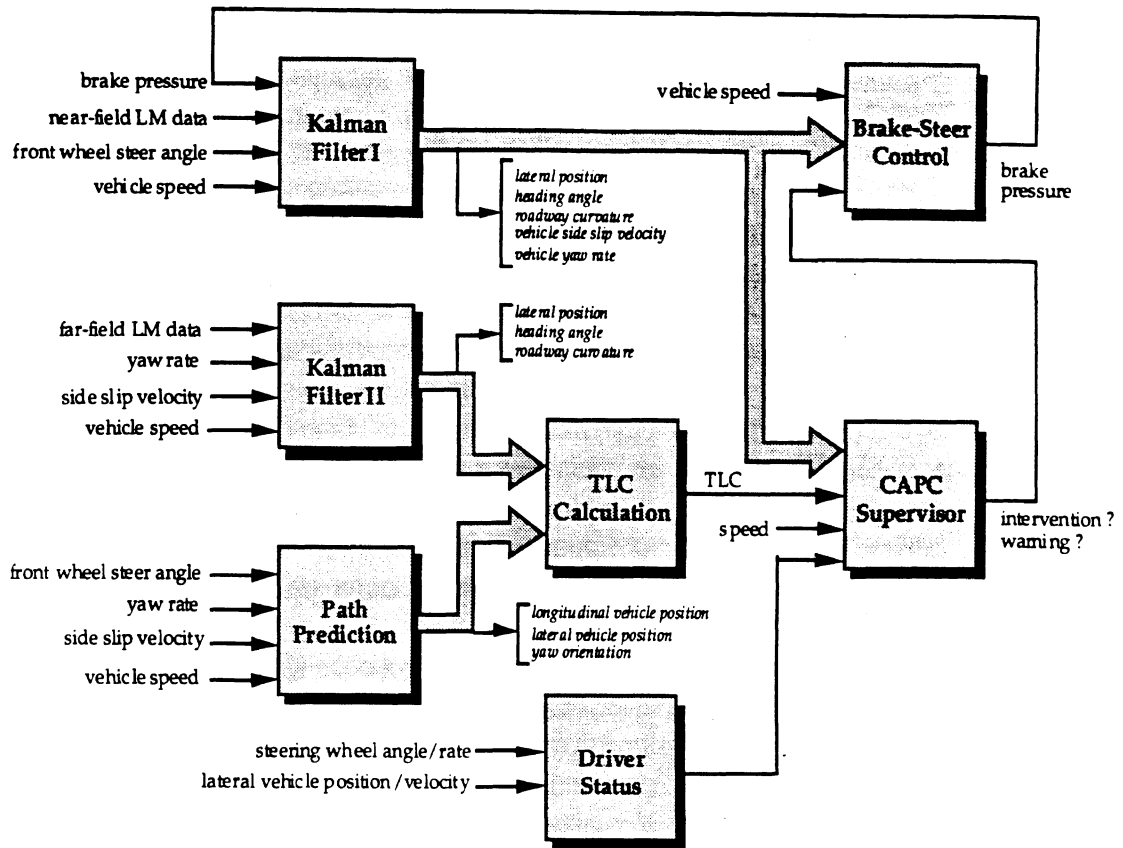


Figure 8.3.3.1 CAPC system lay-out.

Variables

The first operation done in the CAPC control code is retrieving the sensor signals from the A-D converter and the lane marker DSP. The analog sensor variables are denoted by:

d_sw	steering wheels angle (degrees)
d_fw	front wheel steer angle (degrees)
defl_RF	suspension deflection, right front (mm)
defl_LF	suspension deflection, left front (mm)
defl_RR	suspension deflection, right rear (mm)
defl_LR	suspension deflection, left rear (mm)
u_RF	wheelspin speed, right front (m/s)
u_LF	wheelspin speed, left front (m/s)
u_RR	wheelspin speed, right rear (m/s)
u_LR	wheelspin speed, left rear (m/s)
r	vehicle yaw rate (deg/s)
A _y	vehicle lateral acceleration (m/s ²)

The lane marker data set is characterized by an array of x-y data points and some additional variables that describe the amount of points and the state of the lane lines. A distinction has been made between near- and far-range data. The near-range contains all data points between 5 and 20 m longitudinal distance in front of the vehicle and the far-range range extends from 20 to 100 m. The variables of interest are:

SolidL	Boolean variable (0 = dashed left lane line, 1 = solid)
SolidR	Boolean variable (0 = dashed right lane line, 1 = solid)
n_leftN	number of left lane marker points in the Near-range range
n_rightN	number of right lane marker points in the Near-range range
LlaneN_xe[]	array with left lane line x-coordinates in the Near-range range (m)
LlaneN_ye[]	array with left lane line y-coordinates in the Near-range range (m)
RlaneN_xe[]	array with right lane line x-coordinates in the Near-range range (m)
RlaneN_ye[]	array with right lane line y-coordinates in the Near-range range (m)
n_leftF	number of left lane marker points in the Far-range range
n_rightF	number of right lane marker points in the Far-range range
LlaneF_xe[]	array with left lane line x-coordinates in the Far-range range (m)
LlaneF_ye[]	array with left lane line y-coordinates in the Far-range range (m)
RlaneF_xe[]	array with right lane line x-coordinates in the Far-range range (m)
RlaneF_ye[]	array with right lane line y-coordinates in the Far-range range (m)

The derived and output variables of the CAPC control code are given by:

u	vehicle speed (m/s)
ePitch	estimated pitch angle (deg)
eRoll	estimated roll angle (deg)
TLCi	time-to-lane crossing (sec)
sigmaTLC	time-to-lane crossing uncertainty (sec)
P_brake	command brake pressure (Pa)
v_est	estimated vehicle side-slip velocity (m/s)
r_est	estimated vehicle yaw rate (deg/s)
y_est	estimated lateral vehicle position (m)
h_est	estimated vehicle heading angle (deg)
k_est	estimated roadway curvature (1/m)
a0_L	0th-order polynomial coefficient of left lane line (m)
a1_L	1st-order polynomial coefficient of the left lane line (1/m)
a2_L	2nd-order polynomial coefficient of the left lane line (1/m ²)
a0_R	0th-order polynomial coefficient of right lane line (m)
a1_R	1st-order polynomial coefficient of the right lane line (1/m)
a2_R	2nd-order polynomial coefficient of the right lane line (1/m ²)
warning	Boolean variable for warning (0 or 1)

Vehicle speed, roll, and pitch

The vehicle speed has been calculated by averaging the two front wheelspin derived velocities. It has been chosen to ignore the rear wheel derived speeds because of the application of brake-steer control on the rear axle. The speed of travel reads

$$u = \frac{u_{RF} + u_{LF}}{2}$$

The roll and pitch angle of the sprung mass has been derived from the four suspension deflection LVDT sensors. The LVDTs have been calibrated for the displacement in the wheel plane at the middle of the tire. The calibration was carried out by moving the wheels up and down while the sprung mass was fixed with straps to the ground. The gains and offsets of the sensors were calculated by measuring the output voltage of each of the LVDTs and regress these values by a linear 1st-order function with the measured displacement of the hydraulic actuator that moved the wheel. This methods takes into account the deflection of the suspension as well as the tire.

The pitch angle can be determined by subtracting the front and rear suspension deflections (in mm) and dividing it over the wheelbase Wb. Both left and right side LVDTs have been used (averaging). The pitch angle (in degrees) is given by

$$ePitch = \frac{180}{\pi} \cdot \frac{\left(\frac{defl_LR + defl_RR}{2 \cdot 1000} - \frac{defl_LF + defl_RF}{2 \cdot 1000} \right)}{Wb}$$

The roll angle (in degrees) has been derived in a similar fashion as the pitch angle and yields

$$eRoll = \frac{180}{\pi} \cdot \frac{1}{2} \left(\frac{(defl_RF - defl_LF)}{1.167 \cdot Tw1 \cdot 1000} \cdot (1 + RollRatio1) + \frac{(defl_RR - defl_LR)}{1.104 \cdot Tw2 \cdot 1000} \cdot (1 + RollRatio2) \right)$$

with

$$RollRatio1 = \frac{\frac{1}{2} Tw1^2 \cdot c_sw1 + c_rs1}{\frac{1}{2} Tw1^2 \cdot c_t1 + c_rs1}$$

$$RollRatio2 = \frac{\frac{1}{2} Tw2^2 \cdot c_sw2 + c_rs2}{\frac{1}{2} Tw2^2 \cdot c_t2 + c_rs2}$$

For the roll angle a correction needs to be applied for the antiroll bars with stiffnesses c_rs1 and c_rs2 . These bars were *Not* activated during the calibration because both wheels of an axle were moved in plane up and down. Therefore the roll angle is not equal to the difference between the left and right suspension deflection over the trackwidth (Tw). In case of a roll motion, the antiroll bars will introduce an extra force in the suspension to resist the roll motion. Therefore the vertical tire force will be different from the situation as created during the LVDT calibration. The extra tire force due to the antiroll bars will result in an extra tire deflection and thus a different roll angle. The factors 1.167 and 1.104 are the gains representing the ratio between measured deflection at the sensor and measured deflection in the wheel plane (including tire deflections) for a pure vertical motion of the suspension at the front and rear axle respectively. From these numbers the vertical tire stiffness can be calculated quite easily and the values for the front and rear tires are given by

$$c_{t1} = \frac{c_{sw1}}{(1.167 - 1)} = 196,407 \frac{N}{m}$$

$$c_{t2} = \frac{c_{sw2}}{(1.104 - 1)} = 231,731 \frac{N}{m}$$

Vehicle parameters within the CAPC control code

The parameters of interest valid for the Ford Taurus SHO test vehicle are presented in Table 8.3.3.1.

symbol	description	value	unit
C_Fa1	cornering stiffness of 1 front tire	6.376E+4	N/rad
C_Fa2	cornering stiffness of 1 rear tire	6.643E+4	N/rad
a_v	distance CG to front axle	1.073E+0	m
b_v	distance CG to rear axle	1.620E+0	m
Tw1	trackwidth of front axle	1.568E+0	m
Tw2	trackwidth of rear axle	1.521E+0	m
Wb	wheelbase	2.693E+0	m
m_v	total mass of vehicle	1.814E+3	kg
I_zv	total yaw moment of inertia	3.960E+3	kgm ²
TireWidth	tire contact patch width	2.150E-1	m
c_sw1	front suspension spring rate at wheel	3.280E+4	N/m
c_sw2	rear suspension spring rate at wheel	2.410E+4	N/m
c_rs1	roll stiffness, front anti-roll bar	8.586E+4	Nm/rad
c_rs2	roll stiffness, rear anti-roll bar	6.688E+4	Nm/rad

Table 8.3.3.1 CAPC prototype vehicle parameters (vehicle.h).

Functions within the CAPC control code

A flow chart of the CAPC core code was shown in Figure 8.3.2.3. Every 100 ms one complete cycle is computed. The following function calls correspond with each operation of the flow chart:

Get analog + LM data:

- Function: **GetLMSdata();**

This function retrieves the LMS data from the packages as sent through the serial connection between the image processing PC and the Macintosh computer. It stores all the data points in eight

arrays (LlaneN_xe[], LlaneN_ye[], RlaneN_xe[], RlaneN_ye[], LlaneF_xe[], LlaneF_ye[], RlaneF_xe[] and RlaneF_ye[]). If the LMS doesn't send any data points the sensor is supposed to be nonoperational and the flag LMS_ready = false.

- Function: **CheckPosition();**
This function checks the vehicle position on the road. In the prototype car the operations within this function have been disabled meaning that the CAPC system will operate as a lane keeping system rather than a road-keeping system.
- Outputs: LanePosN = 1 → right lane line is used for near-range, 2 → left line
LanePos = integer number denoting the lane number
OnShoulder = Boolean indicating that the vehicle is on the shoulder

Initialize Kalman filter:

- Function: **Init_KalmanN();**
This function initializes the near-range Kalman filter when the CAPC system is started up. The initial conditions of the state estimator are calculated from a sample of near-range lane marker data. The vehicle should be operated on a straight road. A first-order polynomial fit of the lane marker data is used to determine the relative lateral position and heading of the vehicle. All other Kalman filters states are made equal to zero.
- Function: **Init_KalmanF(30.0);**
This function initializes the far-range Kalman filter when the CAPC system is started up. The initial conditions of the state estimator are calculated from a sample of far-range lane marker data. The vehicle should be operated on a straight road. A first-order polynomial fit of the lane marker data is used to determine the relative lateral position and heading of the vehicle. All other Kalman filters states are made equal to zero. The far-range Kalman filter is designed only for one vehicle speed. The input argument represents the design speed (30 m/s in this case). A discretization of the state-space equations will be applied assuming a zero-order sample and hold scheme.
- Function: **Init_BrakeSteer();**
This function initializes the brake-steer controller. Initial conditions are made equal to zero and the brake-steer feedback gains will be calculated for the current vehicle speed.

Path Prediction:

- Function: **Path_Prediction(X_pth);**
This function calculates the future trajectory of the vehicle based on the current speed (u), front wheel steer angle (d_fw) of the vehicle, measured yaw rate (r) and estimated vehicle side-slip velocity (v_est). A simple 2 DOF vehicle model is integrated for one time step each time this function is called. In total this function will be called n_prd times where n_prd equals the total prediction time (4 sec) over the simulation time step. The x-y

coordinates and the yaw angle of the vehicle are stores in an array denoted by $X_pth[]$. Every prediction time step the values are copied to $X_ndl[k]$, $Y_ndl[k]$, $P_ndl[k]$ were k runs from 0 to n_prd .

- Outputs: $X_ndl[k]$ = x-position of vehicle (m)
 $Y_ndl[k]$ = y-position of vehicle (m)
 $P_ndl[k]$ = yaw orientation of vehicle (rad)

Kalman filter (near-range):

- Function: **KalmanFilterN();**
This function computes the near-range Kalman filter. The inputs for the near-range Kalman filter are the near-range lane marker coordinates $LlaneN_xe[]$, $LlaneN_ye[]$, $RlaneN_xe[]$, $RlaneN_ye[]$, the front wheel steer angle (d_fw), the vehicle speed (u) and an estimate of the command brake pressure during interventions (P_brake). Depending on the vehicle position within the lane line ($LanePosN$), either the left or right line markers are used for the near-range Kalman filter. First the Kalman filter feedback gains are calculated using the current vehicle speed and then the Kalman filter differential equations are solved using a 2nd-order Runge-Kutta method with a fixed time step. The estimated states after integration are stores in an array denoted by $Xkf_near[]$;
- Outputs: $Xkf_near[0]$ = estimated side-slip velocity (m/s)
 $Xkf_near[1]$ = estimated yaw rate (rad/s)
 $Xkf_near[2]$ = estimated relative lateral position (m)
 $Xkf_near[3]$ = estimated heading angle (rad)
 $Xkf_near[4]$ = estimated roadway curvature (1/m)

Kalman filter (far-range):

- Function: **KalmanFilterF();**
This function computes the far-range Kalman filter. The inputs for the far-range Kalman filter are the far-range lane marker coordinates $RlaneF_xe[]$, $RlaneF_ye[]$, the estimated vehicle side-slip velocity from the near-range Kalman filter (v_est) and the measured yaw rate (r). The far-range Kalman filter is represented in the discrete-time domain and integration of the dynamics corresponds with a recursive matrix manipulation. The estimated states are stores in an array denoted by $Xkf_far[]$. This array contains information for a right lane only. The Kalman filter states are used to compose the coefficients of a 2nd-order polynomial expression of the left and right lane lines. The two arrays containing the polynomial coefficients are denoted by $LlaneF_pol[]$ and $RlaneF_pol[]$. The left lane line is constructed by shifting the right lane line one lane width to the right.
- Outputs: $Xkf_far[0]$ = estimated relative lateral position (m)
 $Xkf_far[1]$ = estimated - heading angle (rad)
 $Xkf_far[2]$ = estimated 1/2 curvature (1/m)

coordinates and the yaw angle of the vehicle are stores in an array denoted by $X_pth[]$. Every prediction time step the values are copied to $X_ndl[k]$, $Y_ndl[k]$, $P_ndl[k]$ were k runs from 0 to n_prd .

- Outputs: $X_ndl[k]$ = x-position of vehicle (m)
 $Y_ndl[k]$ = y-position of vehicle (m)
 $P_ndl[k]$ = yaw orientation of vehicle (rad)

Kalman filter (near-range):

- Function: **KalmanFilterN();**
This function computes the near-range Kalman filter. The inputs for the near-range Kalman filter are the near-range lane marker coordinates $LlaneN_xe[]$, $LlaneN_ye[]$, $RlaneN_xe[]$, $RlaneN_ye[]$, the front wheel steer angle (d_fw), the vehicle speed (u) and an estimate of the command brake pressure during interventions (P_brake). Depending on the vehicle position within the lane line ($LanePosN$), either the left or right line markers are used for the near-range Kalman filter. First the Kalman filter feedback gains are calculated using the current vehicle speed and then the Kalman filter differential equations are solved using a 2nd-order Runge-Kutta method with a fixed time step. The estimated states after integration are stores in an array denoted by $Xkf_near[]$;
- Outputs: $Xkf_near[0]$ = estimated side-slip velocity (m/s)
 $Xkf_near[1]$ = estimated yaw rate (rad/s)
 $Xkf_near[2]$ = estimated relative lateral position (m)
 $Xkf_near[3]$ = estimated heading angle (rad)
 $Xkf_near[4]$ = estimated roadway curvature (1/m)

Kalman filter (far-range):

- Function: **KalmanFilterF();**
This function computes the far-range Kalman filter. The inputs for the far-range Kalman filter are the far-range lane marker coordinates $RlaneF_xe[]$, $RlaneF_ye[]$, the estimated vehicle side-slip velocity from the near-range Kalman filter (v_est) and the measured yaw rate (r). The far-range Kalman filter is represented in the discrete-time domain and integration of the dynamics corresponds with a recursive matrix manipulation. The estimated states are stores in an array denoted by $Xkf_far[]$. This array contains information for a right lane only. The Kalman filter states are used to compose the coefficients of a 2nd-order polynomial expression of the left and right lane lines. The two arrays containing the polynomial coefficients are denoted by $LlaneF_pol[]$ and $RlaneF_pol[]$. The left lane line is constructed by shifting the right lane line one lane width to the right.
- Outputs: $Xkf_far[0]$ = estimated relative lateral position (m)
 $Xkf_far[1]$ = estimated - heading angle (rad)
 $Xkf_far[2]$ = estimated 1/2 curvature (1/m)

LlaneF_pol[0] = 2nd-order coefficient, left lane line
 LlaneF_pol[1] = 1st-order coefficient, left lane line
 LlaneF_pol[2] = 0th-order coefficient, left lane line
 RlaneF_pol[0] = 2nd-order coefficient, right lane line
 RlaneF_pol[1] = 1st-order coefficient, right lane line
 RlaneF_pol[2] = 0th-order coefficient, right lane line

TLC calculation:

- Function: **Calculate_TLC();**
 This function computes the time-to-lane crossing (TLC) by calculating the point of intersection between the perceived lane lines and the future vehicle trajectory. Both left and right lane lines are checked for intersections. If a lane line is marked as a dashed line, *No* TLC value will be determined for that specific line. If a TLC occurs within the maximum path prediction time (4 sec) a Boolean variable Cross_L or Cross_R will denote whether the intersection took place on the left or right side.
- Outputs: TLC = TLC value (sec)
 Cross_L = true → road departure on the left side
 Cross_R = true → road departure on the right side

Decision making:

- Function: **CAPC_Supervisor();**
 This function determines when to issue a warning and when to intervene. The major inputs are the TLC value and the lateral position of the vehicle with respect to the road surface. Two buzzers will be activated, one for warning and one for intervention.
- Outputs: Warn_on = true → issue warning
 Brake_on = true → intervene

Brake-Steer control:

- Function: **BrakeSteer();**
 This function computes the brake command pressure for the two rear wheel brakes in case of an intervention. The brake-steer controller is based on a combination of a feed-forward (using the front wheel steer angle and roadway curvature) and a feedback of all the near-range Kalman filter states (except for the curvature). The output of the brake-steer controller is at the command brake pressure denoted by P_brake. If P_brake > 0 then the right rear command brake pressure Pa_2R = P_brake, if P_brake < 0 then the left rear command brake pressure Pa_2L = P_brake. Both front axle brakes are not used (Pa_1L = Pa_1R = 0). A 200 ms prediction of the Kalman filter states will be used in order to overcome stability problems while closing the loop due to the latency. The predicted Kalman filter states are stored in the array denoted by Xkf_pred[].
- Outputs: Pa_1L = command pressure Left Front
 Pa_1R = command pressure Right Front

8.3.4. CAPC Simulation Tool code

The CAPC simulation tool was described in detail in Section 3.7. The code was developed on an Apple Macintosh Quadra 800 and consists of about 90,000 lines of C-code. A listing is available upon request. The following C-code files are necessary in order to run the CAPC simulation tool on a Macintosh equipped with a floating point unit:

main_menu.c:

- Function: **LoadMenus();**
This function inserts the menus in the menu bar.
- Function: **ToggleMenuCheck();**
This function toggles a selected menu on or off.
- Function: **SetMenuCheck();**
This function sets a menu check.
- Function: **ToggleDefaults();**
This function toggles the defaults.
- Function: **HandleMenuToggle();**
This function handles selecting the menu toggles.
- Function: **EventLoop();**
This function contains the event loop.
- Function: **HandleEvent();**
This function handles the event in case a mouse button has been pushed or a key has been touched.
- Function: **IsAppWindow();**
- Function: **DoMenu();**
This function is a part of the menu selection.
- Function: **DoMenuSelect();**
This function selects the menu items.
- Function: **DoKeyDown();**
This function handles a key-down event.
- Function: **DoMouseEvent();**
This function handles a mouse event.
- Function: **DoAppleMenu();**
This function handles to "Apple" menu.
- Function: **HiliteButton();**
This function hilites a button in a dialog.
- Function: **CenterDialog();**
This function centers a dialog on the screen.

main_graphics.c:

- Function: **Init_ToolBox();**
This function initializes the graphical toolbox.
- Function: **DetermineScreen();**

- This function determines the type of monitor the user is using. The screen resolution and colors will be set here too.
- Function: **DrawBackGround();**
This function paints the background of the screen.
 - Function: **InitAnimWindow();**
This function initializes the four animation windows.
 - Function: **InitDrivSimWindow();**
This function initializes the driving simulator window.
 - Function: **DrawCar();**
This function draws a top-view of the CAPC car.
 - Function: **DrawTireForces();**
This function draws the magnitude of the tire longitudinal and lateral forces as vectors at each wheel.
 - Function: **DrawRoad();**
This function draws a top-view road and paints it black.
 - Function: **DrawMarkers();**
This function draws the white lane markers on the road.
 - Function: **DrawMarkers_est();**
This function draws the estimated position derived from the imaging system of the lane markers on top of the white lane markers.
 - Function: **DrawNudle();**
This function draws a top-view of the predicted vehicle trajectory.
 - Function: **DrawRuler();**
This function draws a top-view of the ruler with a length of 100 meters and tick-marks at 10-meter intervals.
 - Function: **DrawNearFieldRange();**
This function draws a projection of the range of the near-range camera on the road surface.
 - Function: **DrawFarFieldRange();**
This function draws a projection of the range of the far-range camera on the road surface.
 - Function: **ShowNearFieldScreen();**
This function paints the perceived image as seen through the near-range camera.
 - Function: **ShowFarFieldScreen();**
This function paints the perceived image as seen through the far-range camera.
 - Function: **DrawAnimScreen1();**
This function draws a top view of the car on the road with the magnitude of the tire forces as vectors.
 - Function: **DrawAnimScreen2();**
This function draws a top view of the car on the road. The predicted vehicle path (yellow) and the perceived lane markers

(purple) are also shown. The perceived roadway geometry as a result of Kalman filtering is shown as a blue line.

- Function: **DrawTestScreen();**
This function draws a top view of the car on the road with the magnitude of the tire forces as vectors in the post-processing mode of the CAPC simulation tool.
- Function: **ShowSimulatorScreen();**
This function draws the entire scenery as seen through the eyes of the human driver in the CAPC driving simulator mode.
- Function: **MakeHood();**
This function determines the shape of the hood of the vehicle as seen in through the lane marker sensors.
- Function: **MakeArrow();**
This function determines the shape and size of the arrow as used as warning signals in the driving simulator mode.
- Function: **MakeBrakeMeter();**
This function determines the shape and size of the brake pressure indicator as used in the driving simulator mode.
- Function: **MakeCar();**
This function determines the shape and size of the car (Ford Taurus).

main fileIO.c:

- Function: **PrepareERDfile();**
This function prepares the header of the ERD output file.
- Function: **SaveSetup();**
This function save the setup file (*.STP) that contains all the model and simulation related parameters.
- Function: **ScanParmFile();**
This function reads one line if the setup parameter file.
- Function: **OpenSetup();**
This function opens the dialog for the setup parameter file.
- Function: **LoadSetup();**
This function loads the data from the setup file.

dialog about.c:

- Function: **DoAboutDialog();**
This function displays the "Apple about" dialog.

dialog driver.c:

- Function: **DriverSParmSetDlg();**
This function handles the parameter dialog of the simple preview driver model.
- Function: **DriverOParmSetDlg();**

This function handles the parameter dialog of the optimal preview driver model.

- Function: **DriverMParmSetDlg();**
This function handles the parameter dialog of the mouse driver .

dialog_path.c:

- Function: **PathPredParmSetDlg();**
This function handles the parameter dialog of the vehicle path prediction.

dialog_sensor.c:

- Function: **SensNearParmSetDlg();**
This function handles the dialog of the parameters related to the near-range camera.
- Function: **SensFarParmSetDlg();**
This function handles the dialog of the parameters related to the far-range camera.

dialog_simulation.c:

- Function: **ScenarioParmSetDlg();**
This function handles the parameters related to the scenarios.
- Function: **TimeSetDlg();**
This function handles the parameters related to simulation timing.
- Function: **ElevationParmSetDlg();**
This function handles the parameters related to roadway grade and superelevation.
- Function: **ChangeCheckBox();**
This function handles toggling check boxes on or off.
- Function: **OutPutSetDlg();**
This function handles the selection of output channels of the ERD file.
- Function: **DefaultERDoutput();**
This function toggles the default ERD output channels.

dialog_wind.c:

- Function: **WindContinueParmSetDlg();**
This function handles the parameters related to the continuous wind speeds in 3 directions.
- Function: **WindGustParmSetDlg();**
This function handles the parameters related to the side wind gust.
- Function: **WindRandomParmSetDlg();**

This function handles the parameters related to the random side wind.

- Function: **WindGustRandParmSetDlg();**
This function handles the parameters related to a combination of a side wind gust with randomly distributed wind speeds.

dialog_ABS.c:

- Function: **ABSParmSetDlg();**
This function handles the parameters related to the brake system and antilock brake system dynamics.

dialog_BrakeSteer.c:

- Function: **BrakeSteerParmSetDlg();**
This function handles the parameters related to the brake-steer controller.

dialog_DriverStatus.c:

- Function: **DriverStatusParmSetDlg();**
This function handles the parameters related to the driver status assessment.

dialog_supervisor.c:

- Function: **SupervisorParmSetDlg();**
This function handles the parameters related to the decision making module.

dialog_tire.c:

- Function: **TireParmSetDlg();**
This function handles the road friction adjustment.

dialog_vehicle.c:

- Function: **VehicleParmSetDlg();**
This function handles the basic parameters (such as mass, moments of inertia, wheelbase, speed, etc.) related to the vehicle.
- Function: **SuspensionParmSetDlg();**
This function handles the parameter adjustment of the suspension properties (springs/dampers).
- Function: **SuspensionParmGeo1SetDlg();**
This function handles the parameter adjustment of the front suspension geometry.
- Function: **SuspensionParmGeo2SetDlg();**

This function handles the parameter adjustment of the rear suspension geometry.

- Function: **AeroParmSetDlg();**
This function handles the parameter changes related to the aerodynamic vehicle properties.
- Function: **CruiseControlParmSetDlg();**
This function handles the parameter changes related to the cruise control.

initialization.c:

- Function: **Init_System();**
This function initializes all the parameters (model and simulation related) of interest for the CAPC simulation Tool. It is called only once during start-up.
- Function: **Reset_System();**
This function resets all counters and state variables.
- Function: **Init_DrivSim();**
This function initializes the driving simulator mode.
- Function: **Init_Vehicle();**
This function initializes all the vehicle-model-related parameters and computed derived variables.
- Function: **Init_Road();**
This function initializes all the road model parameters.
- Function: **Init_Driver();**
This function initializes all the driver model parameters.
- Function: **Init_Scenarios();**
This function initializes all the scenario related parameters.

measure.c:

- Function: **GetMeasurements();**
This function copies all analog and image related data such that they are delayed one time step.

Simulation.c:

- Function: **HandleSimulate();**
This function handles the CAPC simulation loop.
- Function: **DrivSimulate();**
This function handles the driving simulation mode. It contains all the function calls to be carried out in one simulation cycle.
- Function: **Simulate();**
This function handles the ordinary simulation mode. It contains all the function calls to be carried out in one simulation cycle.
- Function: **System_Equations();**
This function contains all the equations of motion of the entire CAPC system which need to be integrated in the time domain.

model 7dof.c:

- Function: **Vehicle7_Equations();**
This function contains the equations of motion of the 7 DOF vehicle model.

model 8dof.c:

- Function: **Vehicle8_Equations();**
This function contains the equations of motion of the 8 DOF vehicle model.

model 14dof.c:

- Function: **Vehicle14_Equations();**
This function contains the equations of motion of the 14 DOF vehicle model.

model abs.c:

- Function: **ABSmod_Equations();**
This function contains the equations of motion of the ABS dynamics model.

model cruise.c:

- Function: **CruiseControl();**
This function described the cruise control dynamics.

model tire.c:

- Function: **Magic_Ftx();**
This function contains the equations of the Magic Formula for pure longitudinal braking or traction tire slip forces.
- Function: **Magic_Fty();**
This function contains the equations of the Magic Formula for pure lateral tire slip forces.
- Function: **Magic_Mtz();**
This function contains the equations of the Magic Formula for the aligning torque.
- Function: **TireCFa();**
This function determines the tire slip forces as a function of constant slip stiffnesses.
- Function: **TireCFa_Fz();**
This function determines the tire slip forces as a function of tire load (Fz) dependent slip stiffnesses.
- Function: **Magic_table();**
This function calculates the tire slip forces based on a table look-up version of the Magic Formula.
- Function: **MakeMagicTables();**
This function fills a table (side force versus slip angle, brake force versus slip) based on the Magic Formula equations.

- Function: **Magic_Simple();**
This function determines the tire slip forces (pure slip) based on the Magic Formula.
- Function: **Magic_Tire();**
This function determines the tire slip forces (combined slip) based on the Magic Formula.

model_driver.c:

- Function: **DriveSMod_Equations();**
This function contains the differential equations of the simple preview driver model.
- Function: **DriveOMod_Equations();**
This function contains the differential equations of the optimal preview driver model.
- Function: **CalcPathDev();**
This function calculates the path deviation for the simple preview model.
- Function: **DriverVehicle_Equations();**
This function contains the equations of motion of a simple 2 DOF vehicle model.
- Function: **TransDriver();**
This function calculates the transition matrix for the optimal preview driver model.
- Function: **Steer();**
This function determines the steering wheel angle for the optimal preview driver model.

model_path.c:

- Function: **Path_Prediction();**
This function contains the integration routine for the path prediction based on a Runge-Kutta second-order integration.
- Function: **PathPred2_Equations();**
This function contains the equations of motion of a 2 DOF vehicle model (lateral and yaw) as used for prediction of the future vehicle trajectory.
- Function: **PathPred3_Equations();**
This function contains the equations of motion of a 3 DOF vehicle model (lateral, yaw, and roll) as used for prediction of the future vehicle trajectory.
- Function: **FastPath_Prediction();**
This function contains the equations of motion of a 2 DOF vehicle model (lateral and yaw) and an integration routine based

on a first-order Euler method as used for the prediction of the future vehicle trajectory in the driving simulator mode.

model_road.c:

- Function: **MakeStraight();**
This function contains the code to make a straight piece of road.
- Function: **MakeCurve();**
This function contains the code to make a curved piece of road with a given radius, superelevation and spiral constants.

- Function: **MakeRoad();**
This function determines the geometry and road roughness profiles of several hard-coded test tracks which are available in the CAPC simulation code. The road roughnesses are read from an ERD file which contains information about two tracks.
- Function: **Elevat();**
This function determines the road elevations at a point on the vehicle with given local coordinates.
- Function: **UnevenRoad();**
This function determines the 4 road elevations at each wheel of the vehicle.

model_wind.c:

- Function: **AeroForces();**
This function determines the 3 aerodynamic forces and 3 moments acting on the vehicle body given the wind speeds and aerodynamic coefficients of the car.

CAPC BrakeSteer.c:

- Function: **BrakeSteer();**
This function determines the brake force at each wheel in case an intervention is commanded from the CAPC decision making module.
- Function: **Get_BSGains();**
This function determines the brake-steer feedback gains as a function of the vehicle speed
- Function: **Init_BrakeSteer();**
This function initializes the brake-steer controller.

CAPC FarField.c:

- Function: **PolyShiftFit();**
This function contains a least square polynomial fit routine. The origin of the data set has been shifted such that numerical ill-conditioning is avoided.
- Function: **RecordFarField();**

- This function records the far-range image.

• Function: **GetXYcoordsF();**
 This function abstracts the x-y lane marker coordinates from the far-range image.
- Function: **FitLaneGeoF();**
 This function fits the far-range lane marker coordinates with a polynomial function.
- Function: **RecordPost();**
 This function records the posts along the road in the driving simulator mode.

CAPC NearField.c:

- Function: **RecordNearField();**
 This function records the near-range image.
- Function: **GetXYcoordsN();**
 This function abstracts the x-y lane marker coordinates from the near-range image.
- Function: **FitLaneGeoN();**
 This function fits the near-range lane marker coordinates with a polynomial function.

CAPC RollPitch.c:

- Function: **EstimateRollPitch();**
 This function estimates the roll and pitch angle of the vehicle based on suspension deflection measurements.

CAPC TLC.c:

- Function: **Calculate_eTLC();**
 This function computes the time-to-lane crossing based on a perceived (estimated) roadway geometry.
- Function: **Calculate_TLC();**
 This function computes the time-to-lane crossing based on the true known roadway geometry as available from the roadway geometry model within the simulation.
- Function: **Uncertainty_TLC();**
 This function computes the time-to-lane crossing uncertainty in case polynomial fits are used for the roadway geometry characterization.
- Function: **UncertaintyKF_TLC();**
 This function computes the time-to-lane crossing uncertainty in case a Kalman filter is used for the roadway geometry characterization.

CAPC DriverStatus.c:

- Function: **rMean();**
This function computes a recursive mean.
- Function: **rVar();**
This function computes a recursive variance.
- Function: **ds_perclos();**
This function combines the regression function for the driver status assessment.
- Function: **Init_DriverState();**
This function initializes the driver status assessment.
- Function: **GetDriverState();**
This function determines the driver status assessment quantities.

CAPC initialize.c:

- Function: **Init_ABS_model();**
This function initializes the ABS brake-steer model.
- Function: **Init_RoPi_Estimation();**
This function initializes the roll/pitch estimation routine.
- Function: **Init_Path_Prediction();**
This function initializes the path prediction.
- Function: **Init_Lane_Sensor();**
This function initializes the lane marker sensors.

CAPC KalmanF CF.c:

- Function: **KalmanFilterF();**
This function computes the far-range Kalman filter.
- Function: **EvalLeftF();**
This function evaluates the y-coordinate of a left far-range lane markers at a given x-position.
- Function: **EvalRightF();**
This function evaluates the y-coordinate of a right far-range lane markers at a given x-position.
- Function: **EvalRightN();**
This function evaluates the y-coordinate of a right near-range lane markers at a given x-position.
- Function: **Get_KFinputF();**
This function determines the Kalman filter lane marker coordinate inputs.
- Function: **Init_stateKalmanF();**
This function finds the initial conditions of the Kalman filter states.
- Function: **Init_KalmanF();**
This function initializes the far-range Kalman filter.
- Function: **C2D();**
This function computes the discrete-time representation of a continuous-time state-space representation.

CAPC KalmanN.c:

- Function: **fake_KalmanFilterN();**
This function fakes a Kalman filter in case an LQ brake-steer controller is used without a near-range Kalman filter.
- Function: **KF_equationsN();**
This function contains the differential equations of the near-range Kalman filter.
- Function: **KalmanFilterN();**
This function computes the near-range Kalman filter states.
- Function: **CheckPosition();**
This function determines the vehicle position on the road using variables obtained through the imaging system.
- Function: **Get_KFinputN();**
This function determines the Kalman filter lane marker coordinate inputs.
- Function: **Get_KFgainsN();**
This function determines the Kalman filter feedback gains as a function of the vehicle speed.
- Function: **Init_stateKalmanN();**
This function finds the initial conditions of the Kalman filter states.
- Function: **Init_KalmanN();**
This function initializes the near-range Kalman filter.

CAPC supervisor.c:

- Function: **CAPC_Supervisor();**
This function contains the CAPC decision-making module.

model 2dof.c:

- Function: **Vehicle2_Equations();**
This function contains the equations of motion of a 2 DOF vehicle model.

testing.c:

- Function: **GoTesting();**
This function contains one cycle of the post-processing mode simulation loop.
- Function: **ReadERDfile();**
This function reads the binary (floating-point) ERD file obtained from the CAPC prototype vehicle.
- Function: **Init_PostSim();**
This function initializes the CAPC post-processor mode.

8.3.4 Lane-Mark-Sensor Software Code

Appendix E is a code listing of the control software used in the LMS Processor. This program was written in Forth, using Polyforth Inc's pF32-386/pMSD Forth for the Intel 386+ processors.

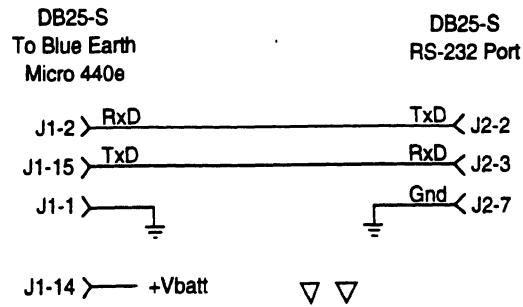
8.4 Drawings

The following pages include three sets of drawings. Section 4 refers to these drawings; revisit that section to put these drawings in context. The schematics are:

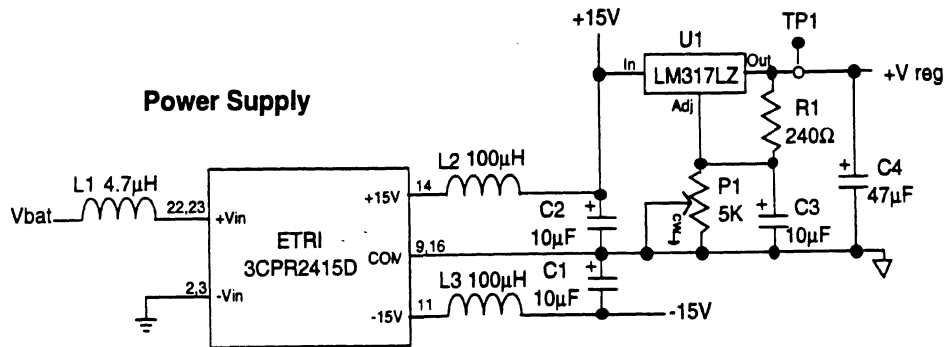
- Brake-steer pressure servo design (incomplete):
 - pressure transducer signal conditioning
 - power supply
 - communications to BlueEarth microcontroller
 - interface wiring
- Design of signal conditioning converters for the LVDTs and the wheel speed sensors
- Transducer cable diagrams

Brake-steer Schematics

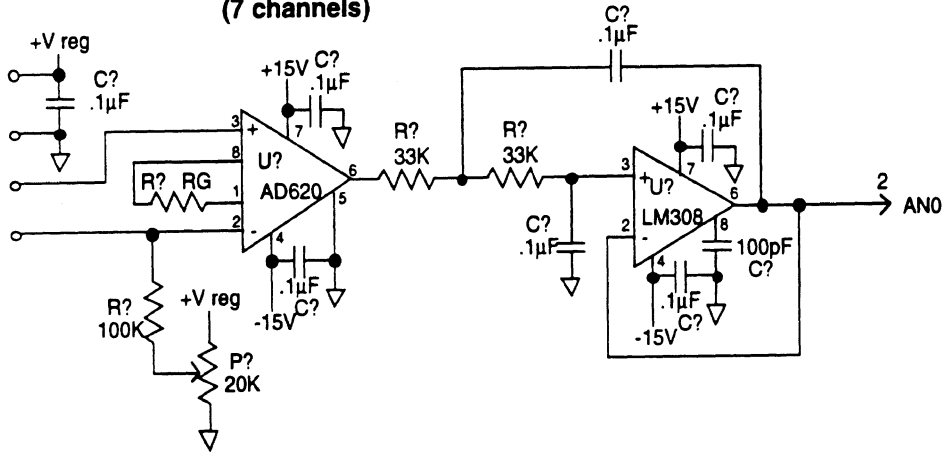
Communications



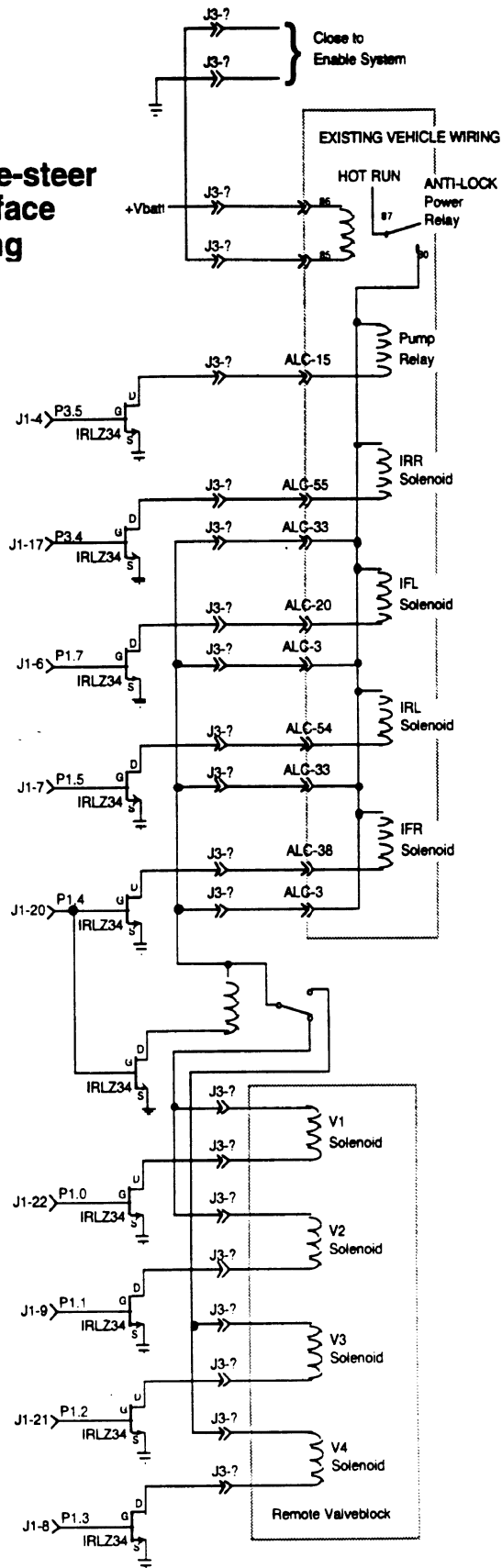
Power Supply



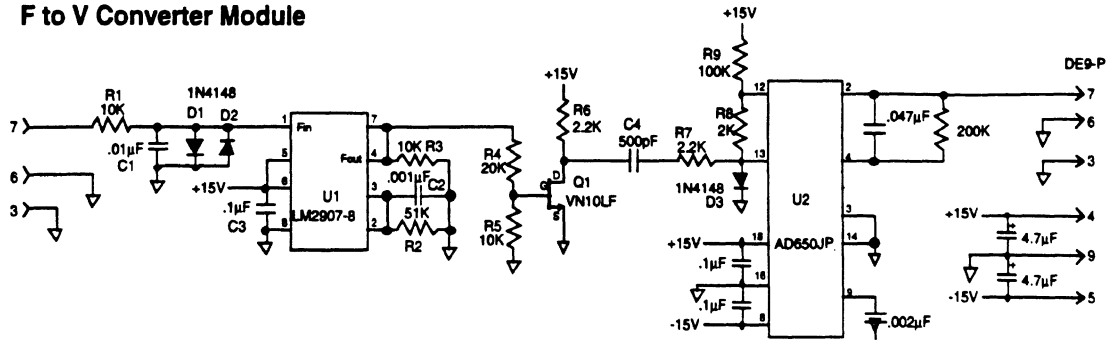
Pressure Xducer Sig. Cond. (7 channels)



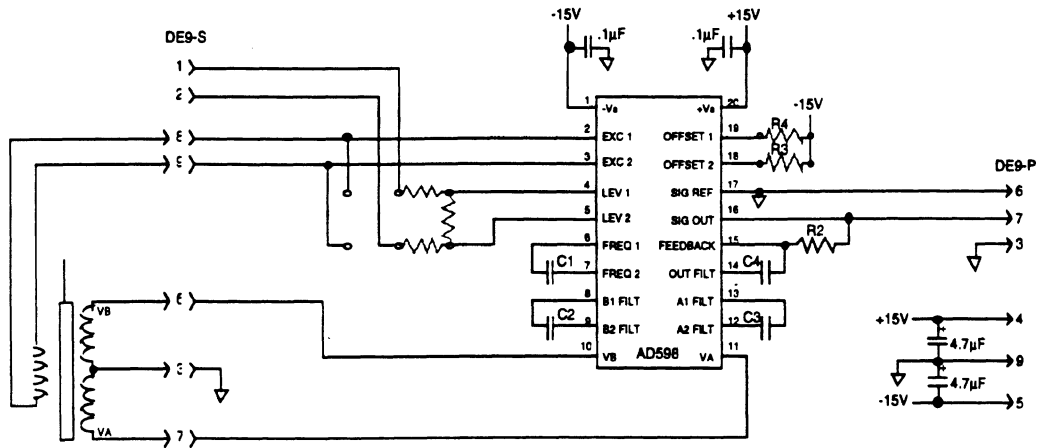
Brake-steer interface Wiring



F to V Converter Module

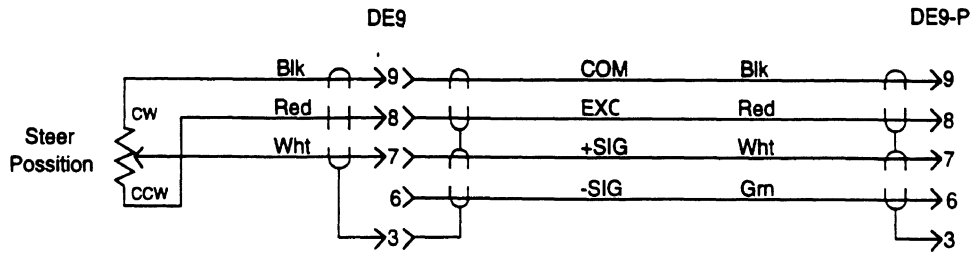


LVDT Signal Conditioning Module

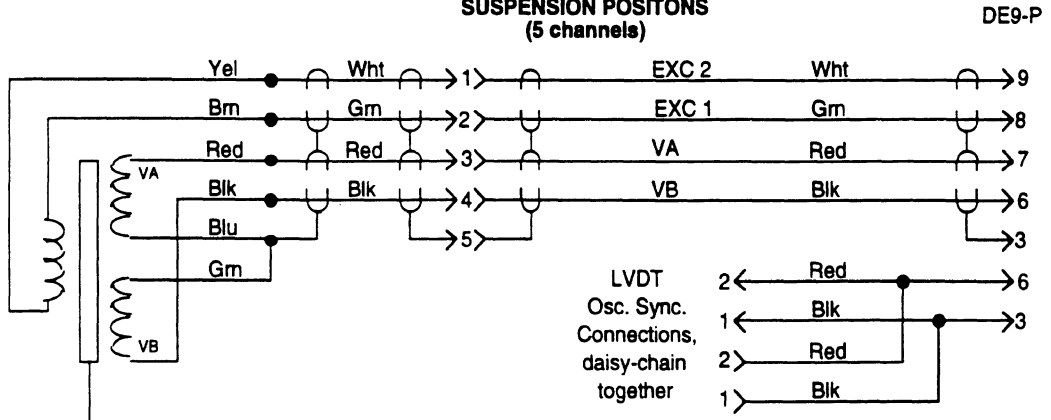


Transducer Cables

STEERING WHEEL POSITION POT.

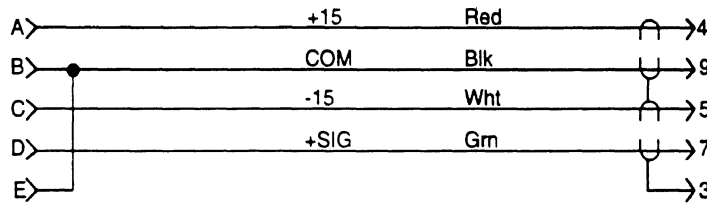


LVDT WIRING FOR STEER RACK AND SUSPENSION POSITIONS (5 channels)



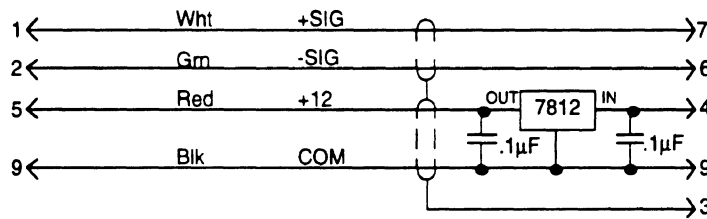
AY

DE9-P



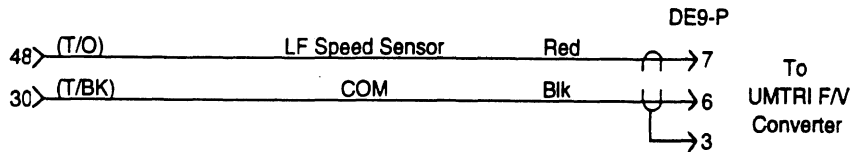
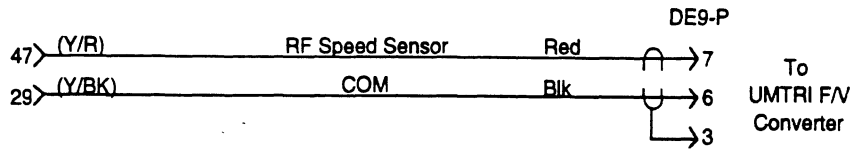
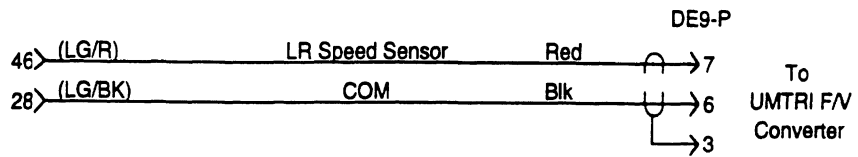
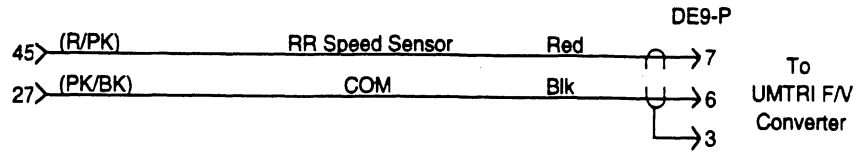
YAW RATE

DE9-P



Transducer Cables (continued)

C1057 on
ANTI-LOCK BRAKE
CONTROL MODULE



Appendices

Appendix A

User Manual for CAPC Simulation Tool

Simulation Tool for a Road Departure Avoidance System

User's Manual

CAPC v4.1

July 1995

Paul J.Th. Venhovens
The University of Michigan
Transportation Research Institute

The different sub-systems of a lane departure avoidance system have been combined on a simulation level in the CAPC (Crewman's Associate for Path Control) simulation tool. The simulation tool is a modular concept written in C-language and is running on a Macintosh computer. The main objective of the tool is to combine the sub-systems to one total lane departure avoidance system and to study the performance and interaction among the different modules. The seven major parts of the CAPC simulation tool are: the vehicle, the lane marker recording and processing, the estimation of the future trajectory of the vehicle, the time-to-lane crossing calculation, the brake-steer controller, the driver status assessment and the CAPC supervisory controller.

The core of the simulation is the vehicle. While driving on the road the vehicle is subjected to all kinds of external inputs like for example steering wheel rotation, road unevennesses and wind. The CAPC simulation tool offers a variety of vehicle models, each designed for a specific application. The most extensive model contains 14 degrees of freedom (DOFs). The sprung mass is able to move in 3 directions (longitudinal, lateral and vertical) and to rotate about 3 axes (roll, pitch and yaw). Each wheel suspension has one DOF with respect to the vehicle body and the rotation of a wheel also accounts for one DOF. This 14 DOF model is the most extensive vehicle dynamics model available in the CAPC simulation tool and contains all the major DOFs necessary to describe the motions of the vehicle of interest for the CAPC application. Two simplified models being a 7 and a 8 DOF vehicle model are also available. The models are simplified with respect to the wheel suspensions. The 7 DOF model doesn't contain suspensions (and can therefore only describe the longitudinal, lateral and yaw motion of the vehicle) and the 8 DOF model adds the roll motion of the sprung mass to the 7 DOF model. The simplified models are more computationally efficient but less accurate than the 14 DOF model.

The tire plays a crucial role in vehicle dynamics. It accomplishes essentially three basic functions: (1) support the vehicle weight, cushioning road irregularities, (2) develop lateral forces for cornering, (3) develop longitudinal forces for accelerating and braking. Several tire models are available in the CAPC simulation tool. The most extensive model is based on an empirical model known under the name Magic Formula tire model. It can describe the tire slip force for a large range of load and slip quantities. Furthermore, it offers the possibility to treat the case of combined slip (cornering and braking or accelerating at the same time). The less extensive models are all based on the Magic Formula but contain certain simplifications in order to speed up the simulation time.

The CAPC simulation vehicle can be operated on various roadway types with different geometries and unevennesses. Besides an oval test track also a straight road, a winding road and a skid pad have been pre-programmed. The geometry can be extended with grades and superelevations. Furthermore, several data sets of different artificially generated road unevenness profiles are available. The road friction coefficient can be changed too.

Most of the maneuvers done with the simulation vehicle will be carried out by a driver. Two driver models are available in the CAPC simulation tool. The simple driver model is characterized by a preview model that looks at a single point in front of the vehicle. The other option is a more elaborate optimal preview model that minimizes the tracking error at several points in front of the vehicle.

The CAPC vehicle will be equipped with two vision systems: one for the near-field range and one for the far-field area. Both cameras are modeled in the CAPC simulation tool. The image of the camera can be determined by applying several coordinate transformations to the known roadway geometry in front of the vehicle assuming a flat earth model. The hardware specifications being the

camera's CCD chip size and resolution have been modeled too. Furthermore, the position of the cameras on the vehicle, the orientation and the update frequency can be chosen arbitrary.

The following subsystems are available in the CAPC simulation tool:

- ◆ **Vehicle Models:**
 - 7 DOF flat vehicle model: longitudinal, lateral, yaw and 4 wheel rotational DOFs
 - 8 DOF yaw/roll model: 7 DOF model + roll DOF
 - 14 DOF full vehicle model: 6 DOFs for the vehicle body, 1 DOF for each axle and 4 wheel rotational DOFs.
- ◆ **Tire Models:**
 - Steady-state (different road friction coefficients possible):
 - * Cornering stiffness as a function of vertical tire load
 - * Magic Formula (pure slip, combined slip)
 - Transient: first-order relaxation system (tire load dependent).
- ◆ **Anti-Lock Brake System Model:**
 - First-order lag system with brake pressure saturation.
- ◆ **Driver Model:**
 - Simple preview model with driver limitations (time lag).
 - UMTRI's optimal preview driver model.
- ◆ **Cruise-Control** (set, resume, accelerate)
- ◆ **Far-Field Camera** determining the future roadway geometry.
- ◆ **Near-Field Camera** determining the heading angle and lateral deviation.
- ◆ **Path Prediction** based on:
 - 2 DOF linear flat vehicle model (lateral and yaw DOF).
 - * Linear tires, fixed cornering stiffnesses
 - * Non-linear tires, cornering stiffnesses as a function of vertical tire load
 - 3 DOF linear yaw/roll vehicle model (lateral, roll and yaw DOF).
 - * Linear tires, fixed cornering stiffnesses
 - * Non-linear tires, cornering stiffnesses as a function of vertical tire load
- ◆ **Lane Margin Calculation:** time-to-lane crossing.
- ◆ **Driver Status** assessment based on vehicle states and steering wheel activation.
- ◆ **Brake-Steer Controller** based on LQ feedback of the lateral deviation, heading angle, side slip velocity and yaw rate.
- ◆ **CAPC Supervisory Controller** based on driver status and time-to-lane crossing.
- ◆ **Roll & Pitch angle estimation** based on measured suspension deflections.
- ◆ **Road Unevennesses** collected from empirical road models.
- ◆ **Roadway Geometry:** straight lines, curves w/lw/o superelevations and/or slopes.
- ◆ **Wind Disturbances:** constant wind, crosswind gust, random crosswind.
- ◆ **Real-Time Driving Simulator :**

- 8 DOF yaw/roll model including wheelspin DOF, Magic Formula tires, cruise control, path-prediction, lane margin calculation, warnings and intervention.

The CAPC simulation tool is a modular concept written in C-language. The simulation tool is menu driven and is supported by various animations. The user is able to modify all important model parameters using pull-down menus and dialogs. The simulations are supported by graphical and numerical outputs. The graphical animation shows the vehicle from a top view including the roadway geometry, predicted future trajectory and perceived roadway geometry. The scenery of the roadway as recorded by both vision systems is also displayed during the animation. Figure 1 gives an impression of the animation.

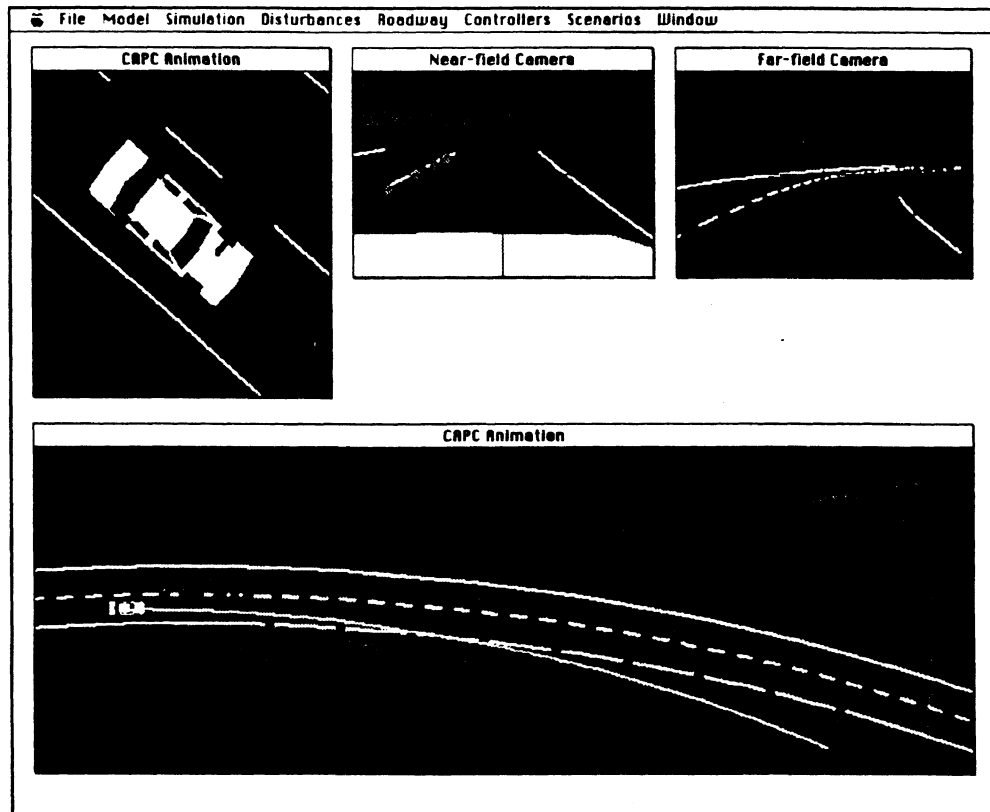


Figure 1. CAPC simulation.

The numerical output contains all important states of the vehicle, roadway geometry, disturbances, vision system outputs and CAPC control in- and outputs. The data is stored in an ERD format and can be displayed by a separate public domain engineering plotter. It is also possible to look at the simulation in progress (real-time) from the point of view of the driver.

The CAPC simulation tool offers the possibility to study the interactions among the various lane keeping subsystems. For example, the influence of external disturbances like road unevennesses and wind on the determination of the perceived roadway geometry has been studied using the CAPC simulation tool. Since the vision system is mounted rigidly on the coach of the vehicle, the motion of the vehicle will affect the estimates of the lane marker locations due to heave, roll and pitch motions of the sprung mass. The image of the vision system can be stabilized successfully for the low frequency vehicle motions using suspension deflection sensor.

The simulation tool has played a significant role as the CAPC system grew to its final (hardware) design. A priori to the tests on the proving ground, control strategies have been implemented in the simulation tool to verify their efficiency and stability. The final controller and the CAPC

simulation code have both been written in C-language. The ability to exchange code has simplified and shortened the implementation phase significantly. When starting up the CAPC simulation tool the following screen will appear:

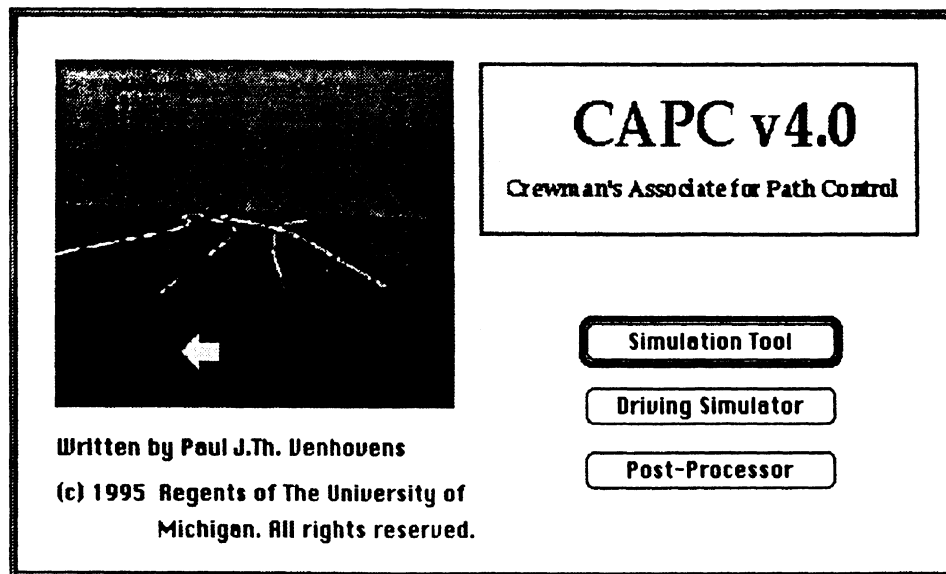
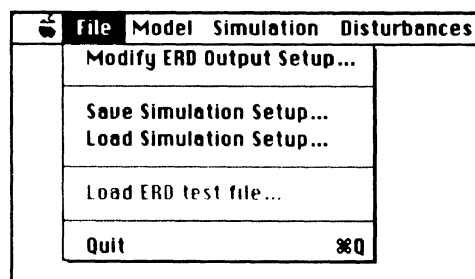


Figure 2. CAPC start-up screen.

The simulation tool offers three modes of operation. The first mode is the *Simulation* mode that includes all the features of the lane departure avoidance system. The second mode is a real-time *driving simulator* mode and it is a down-scaled version of the ordinary *simulation* mode. Some items have been disabled (such as the 14 DOF vehicle model) because of the computational burden. The *driving simulator* mode offers an animated view of the vehicle and roadway as seen through the eyes of the driver. The third mode is a post-processor option that allows the user to input measured steering wheel angle and vehicle speed from the prototype car directly into the simulation. The input file needs to be an ERD binary file with specific channel short names. After selecting one of the modes, a menu-bar will appear on top of the screen. The following section will explain the meaning of each of the menu-items. It might be used as a quick user manual for operating the CAPC simulation tool.

The File menu



The file menu enables to user to modify the simulation IO. It contains the following menu items:

- *Modify ERD Output Setup*

The user may select and modify the variables to be saved for plotting after a simulation run. The output selection can be done by selecting this item. A dialog will appear with a large amount of check boxes. Choose the signals that you would like to study after a simulation run. The data will be stored in a text ERD-format and the signals can be viewed using the ERD plotter.

- **Save Simulation Setup**

All the parameters of a specific vehicle are stored in a setup file (*.STP). Besides vehicle parameters the setup file contains also simulation related parameters, such as time-step, road unevennesses and controller gains. After modifying the vehicle and simulation related parameters, the setup can be saved by selecting this menu item. The setup file needs to have the extension .STP. When starting up the simulation tool the default vehicle parameters will be loaded from the file **default.STP**. This file may not be erased. The program comes with two vehicle configurations: Taurus_SHO.STP and Crown_Victoria.STP. The default car corresponds with Taurus_SHO.

- **Load Simulation Setup**

A previously saved setup can be loaded by selecting this item. All the current vehicle and simulation related parameters will be replaced by the data from the setup file.

- **Load ERD test file**

This option is only selectable in the Post-Processor mode of the CAPC simulation Tool. A file in an ERD format with specific short names will be used to run a simulation in a post-processor mode. This file must be recorded with the CAPC prototype vehicle + MacDAS software. The front wheel steer angle and vehicle speed as stored in the ERD file are ported directly into the simulation tool such that a simulation can be carried out with measured driver inputs. The ERD file must be in a binary floating-point format and needs to contain the following short names:

- d_sw (steering wheel angle in degrees)
- d_fw (front wheel steer angle in degrees)
- u_RF (right front wheel speed in m/s)
- u_LF (left front wheel speed in m/s)
- u_RR (right rear wheel speed in m/s)
- u_LR (left rear wheel speed in m/s)

The vehicle speed is derived by averaging all four wheelspin derived speeds.

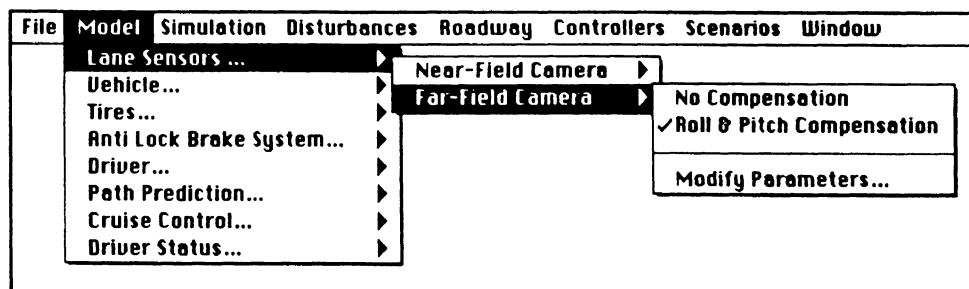
- **Quit**

Selecting this menu item will quit the simulation program.

The Models menu

The simulation menu allows the user to choose and modify model related items. Different models for the vehicle, tires, driver and path prediction can be selected. The complexity of the model varies with the selection and it is also possible to change specific parameters of each model. The models menu contains 8 items being:

Vision System



The CAPC vehicle is equipped with two vision systems (digital video cameras):

- **Near-Field Camera**

The near-field vision system is scanning the roadway geometry close in front of the vehicle.

- *Far-Field Camera*

The far-field vision system is also scanning the roadway geometry in front of the vehicle. However, the range is longer as with the near-field camera.

Both vision system related menus come with a submenu. The cameras are fixed on the vehicle body. This implies that vehicle motions such as body roll during cornering, pitch during braking or acceleration or a combination of heave, roll and pitch when the vehicle is operated on an uneven road affect the images of the cameras. The transformation from screen coordinates to roadway x-y coordinates can be compensated for the motion of the vehicle. Two options are available:

- *No Compensation*

No compensation means that the motion of the vehicle is not included in the transformations from image coordinates to roadway geometry coordinates.

- *Roll & Pitch Compensation*

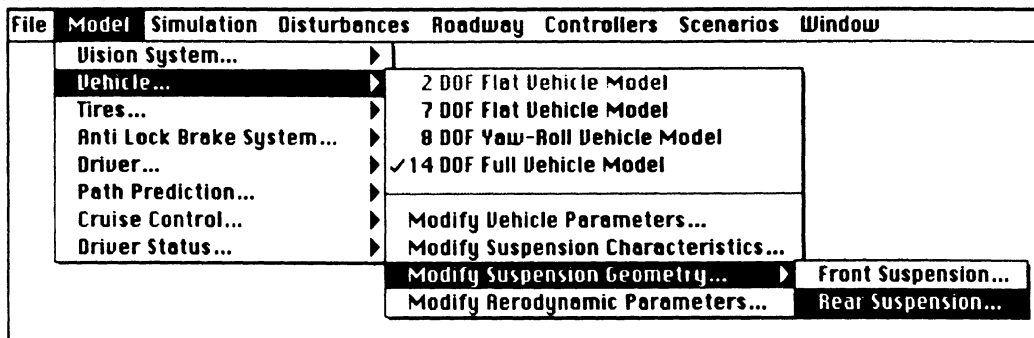
A correction will be applied in the transformation based on estimates of the roll and pitch angle of the vehicle determined by suspension deflection measurements.

The last submenu item is:

- *Modify Parameters*

Selecting this menu item enables the user to modify the position and orientation of the cameras on the vehicle. Furthermore, the focal length, image update rate and the range of interest can be changed.

Vehicle Model



The Vehicle model menu enables the user to change the type and complexity of the vehicle model and to alter the parameters of this model. It contains the following selections:

- *2 DOF Flat-Vehicle Model*

The 2 DOF vehicle model is characterized by the lateral and yaw DOFs. This model can only be used in the Post-Processor mode.

- *7 DOF Flat-Vehicle Model*

The 7 DOF vehicle model is characterized by the following seven degrees of freedom: longitudinal, lateral, yaw and 4 wheel rotational DOFs.

- *8 DOF Yaw-Roll Vehicle Model*

The 8 DOF vehicle model is characterized by the following eight degrees of freedom: longitudinal, lateral, roll, yaw and 4 wheel rotational DOFs.

- *14 DOF Full Vehicle Model*

The 14 DOF vehicle model is characterized by the 6 DOFs for the vehicle body, 1 DOF for each axle and 4 wheel rotational DOFs.

- *Modify Vehicle Parameters*

This menu item enables the user to change important vehicle parameters such as the masses, moments of inertia and vehicle speed of travel.

- **Modify Suspension Characteristics**

The suspension spring and damping constants can be changed if this item is selected. Furthermore it allows the user to change the vertical tire stiffness and auxiliary roll stiffness of the anti-roll bars.

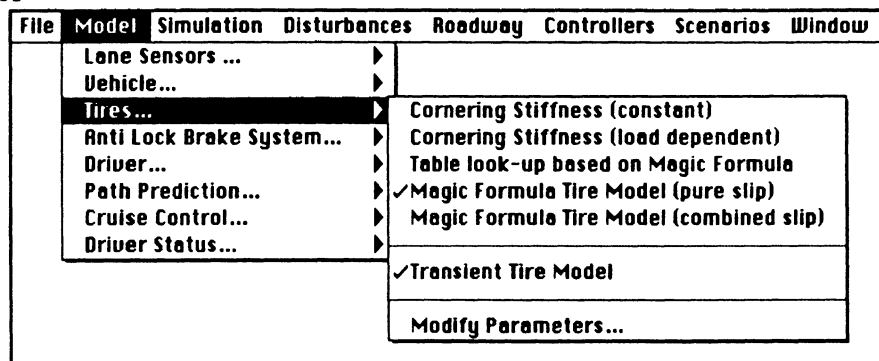
- **Modify Suspension Geometry**

The geometry of the front and rear suspension can be altered by selecting this item. The geometry of the true suspension has been represented by a simplified suspension model. Modifying the parameters will affect the track width and roll center height.

- **Modify Aerodynamic Parameters**

The six aerodynamic coefficients of the vehicle body can be changed when this item is selected. Furthermore, the frontal area of the car, the barometric pressure and air temperature can be altered.

Tire Model



The tires are separated from the vehicle menu item. The following slip force models can be chosen:

- **Cornering Stiffness (constant)**

This tire model is the simplest model available. It is using constant slip stiffnesses to calculate the tire slip forces and moments

- **Cornering Stiffness (load dependent)**

This tire model takes the load dependency of the slip stiffnesses into account while calculating the tire slip forces.

- **Table look-up based on Magic Formula**

The Magic Formula is an empirical tire model that is valid for a wide range of load/slip combinations. The output of the model is a longitudinal and lateral tire slip force, and the aligning moment. The inputs are the longitudinal slip, the side slip angle, the vertical tire load and the wheel camber angle. The table look-up version of the Magic Formula is based on a table that is filled with numbers generated by the original version of the Magic Formula. This version is faster because less computations have to be carried out. The table look-up version doesn't consider combined slip (cornering and braking at the same time).

- **Magic Formula Tire Model (pure slip)**

This tire model is identical to the look-up version as described above. It uses the original Magic Formulae and considers only pure slip situations.

- **Magic Formula Tire Model (combined slip)**

The combined slip version of the Magic Formula is the most extensive tire model available in this simulation tool. Unfortunately, the computational burden is significant.

- **Transient Tire Model**

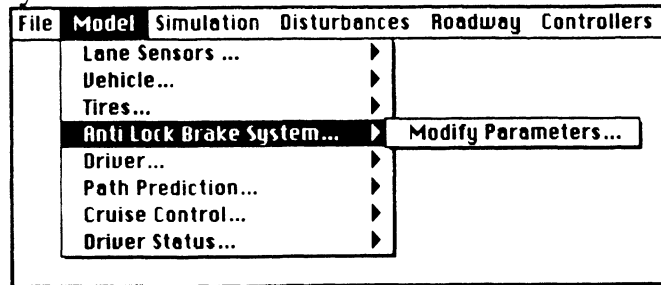
Selecting this menu item toggles the transient tire model on or off. The transient tire model is based on a first-order relaxation system with a load dependent relaxation

length. This option can be combined with any of the five previously described steady-state tire models.

- *Modify Parameters*

Selecting this item will allow the user to modify the tire-road friction coefficient.

Anti-Lock Brake System Model

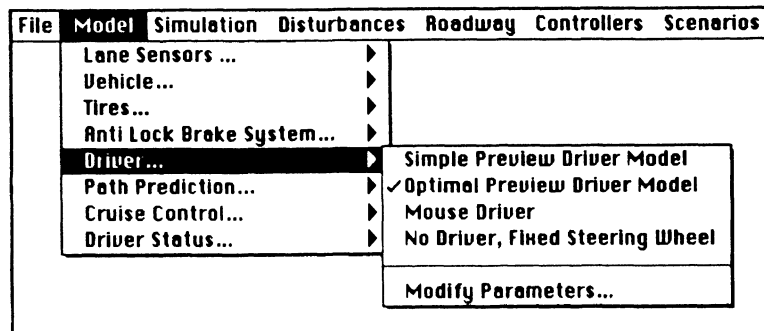


Some of the dynamic properties and parameters of the anti-lock brake system can be altered by selecting:

- *Modify Parameters*

Beside changing the brake system related parameters such as effective rotor radius, brake cylinder bore and friction coefficient between pad and rotor also the time constant of the brake system model (first-order system) can be modified.

Driver Model



The CAPC simulation tool has two different kinds of driver models built in:

- *Simple Preview Model*

This model is relatively simple and steers the vehicle by tracking one point at a fixed distance in front of the vehicle. The model includes limitations of the human driver such as a neuromuscular related dynamics and a pure time delay. The simplicity of the model restricts the application only to straight line driving eventually in combination with disturbances like side wind gusts.

- *Optimal Preview Model*

This model is a more sophisticated version of the simple preview model. It minimizes the tracking error at several points in front of the vehicle. This model is better for complicated maneuvers and driving through curves. The model also included a pure time delay.

Besides the two driver models the following options are available:

- *Mouse Driver*

The vehicle can be steered by moving the mouse from left to right. The neutral position corresponds to the middle of the screen.

- *No Driver, Fixed Steering Wheel*

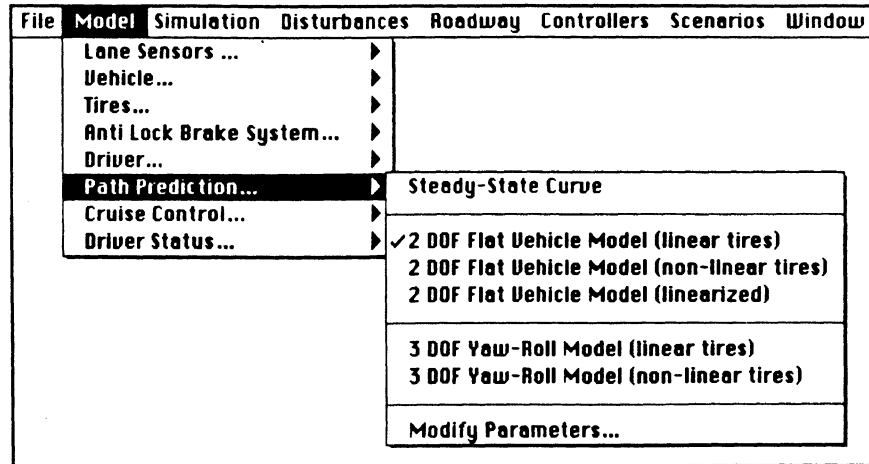
As the name already suggest, the steering wheel is fixed in the neutral position with this option.

The characteristic parameters of the driver model can be altered by selecting:

- *Modify Parameters*

Depending on the driver model chosen (simple, optimal preview, mouse driver) the selection will enable the user to modify time constants and gains.

Path Prediction Model



The time-to-lane crossing algorithm as used in the CAPC program uses the future vehicle trajectory as an input to the calculations. The future path of the vehicle is based on a simple vehicle model assuming that the steering wheel position and vehicle speed remain constant during the projection. The path is obtained by integrating the differential equations of the vehicle. Appropriate initial conditions are necessary. The following path prediction models/methods are available:

- *Steady-state Curve*

The steady-state curve prediction is based on the steady-state solution of the differential equations of a 2 DOF vehicle model (lateral and yaw). The solution has the shape of a curve with a fixed radius. The radius depends on the steering wheel input, vehicle parameters and speed of travel.

- *2 DOF Vehicle Model (linear tires)*

The path projection is based on a 2 DOF vehicle model (lateral and yaw) and a linear tire model (constant cornering stiffnesses).

- *2 DOF Vehicle Model (non-linear tires)*

The path projection is based on a 2 DOF vehicle model (lateral and yaw) and a non-linear tire model (load dependent cornering stiffnesses).

- *2 DOF Vehicle Model (linearized)*

The path projection is based on a 2 DOF vehicle model (lateral and yaw) and a linear tire model (constant cornering stiffnesses). The geometry related to the path prediction has been linearized: $\cos(\psi) \approx 1$, $\sin(\psi) \approx \psi$.

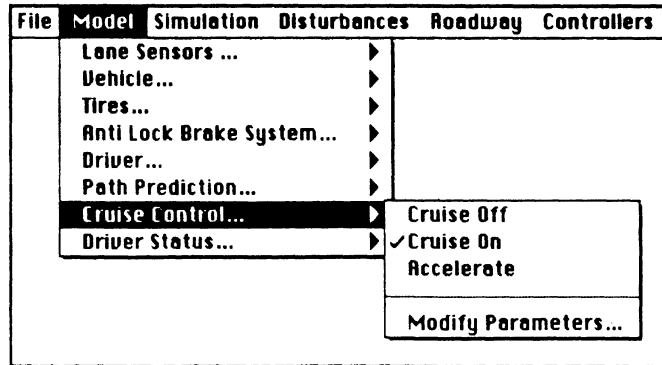
- *3 DOF Vehicle Model (linear tires)*

The path projection is based on a 3 DOF vehicle model (lateral, roll and yaw) and a linear tire model (constant cornering stiffnesses).

- *3 DOF Vehicle Model (non-linear tires)*

The path projection is based on a 3 DOF vehicle model (lateral, roll and yaw) and a non-linear tire model (load dependent cornering stiffnesses).

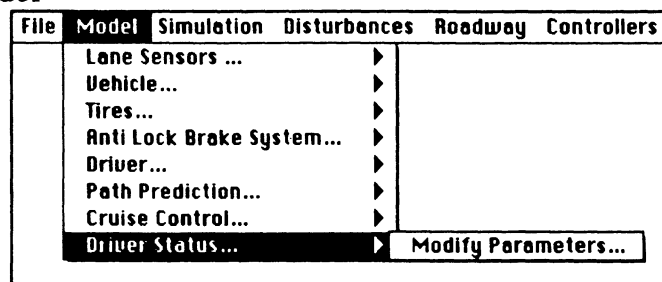
Cruise Control



The cruise control has three modes of operation:

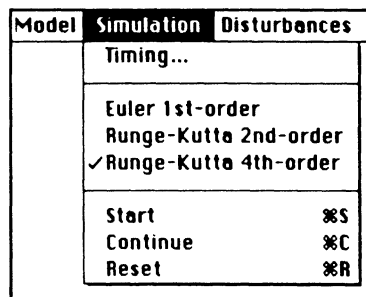
- *Cruise Off*
The cruise control is switched off.
- *Cruise On*
The cruise control is switched on and holds the forward speed of the vehicle constant.
- *Accelerate*
The cruise control is switched on and the controller follows a predefined speed pattern. The speed pattern can be modified by selecting the *Modify Parameters* menu item.

Driver Status Model



The status of the driver is an input for the CAPC decision module. The algorithm monitors several vehicle-roadway related signals such as lateral deviation of the vehicle and steering wheel input. The moving average and standard deviation values are combined to one quantity denoted by PERCLOS (the proportion of the time that the driver's eyes are 80 to 100 percent closed). The regression parameters can be changed by selecting the *Modify Parameters* menu item.

The Simulation Menu



Pulling down the simulation menu enables the user to start a simulation or change integration related setting. The following menu items are selectable:

- *Timing*

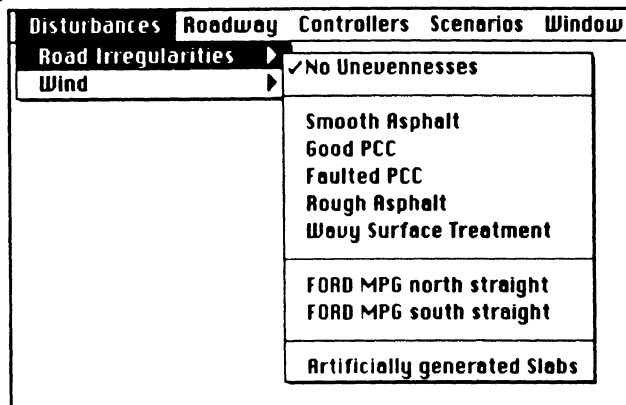
Selecting the timing option enable the user to change the simulation time step, duration and communication interval. The communication interval determines the output to the ERD-data file and to the screen.

- *Euler 1st-order*
This is the fastest integration routine available. The integration of the differential equations is less accurate as with the more elaborate methods. The integration time step is constant.
- *Runge-Kutta 2nd-order*
This method is a down-scaled version of the Runge-Kutta 4th-order integration method. It is faster than 4th-order routine but also less accurate. The integration time step is constant.
- *Runge-Kutta 4th-order*
This is the most elaborate integration method available in the CAPC software. The integration time step is constant.
- *Start*
Selecting this item will start the simulation. A menu will appear were the user is able to assign a name to the ERD output file. The simulation can be interrupted by pressing a button on the mouse.
- *Continue*
After termination of a simulation run (by pressing a mouse button) it is possible to continue with the simulation from the current position. The initial values correspond with the values after the termination of the previous simulation run.
- *Reset*
This option resets all the state variables in the simulation. All initial conditions will be set to zero (with exception of the vehicle speed).

The Disturbance Menu

The vehicle can be subjected to different kinds of disturbances. Two sources are available being road unevennesses and wind.

Road Irregularities



This menu offers the following selectable items:

- *No Unevennesses*
The road surface is smooth.
- *Good PCC*
where PCC stands for Portland Cement Concrete.
- *Faulted PCC*

- *Rough Asphalt*

- *Wavy Surface Treatment*

- *FORD MPG north straight*

This is a road unevenness data file derived from a measured data file recorded at the Ford Michigan Proving Ground, north straight-away.

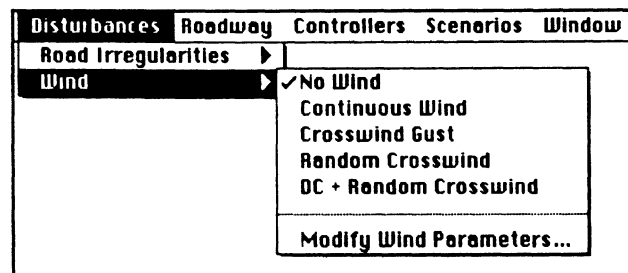
- *FORD MPG south straight*

This is a road unevenness data file derived from a measured data file recorded at the Ford Michigan Proving Ground, south straight-away.

- *Artificially Generated Slabs*

The length of the slabs is 9 m. They are modeled as smooth flat plates with a slight positive slope. The joints between the plates is discontinuous and the discontinuity is modeled as a random process. The height difference during the transition from plate to plate ranges between 6.4 and 12.7 mm.

Wind



The wind disturbance menu offers the following items to be selected:

- *No Wind*

The vehicle will not be exposed to wind disturbances. There will be no aerodynamic forces acting on the vehicle. Even the speed of the vehicle itself does not generate aerodynamic forces.

- *Continuous Wind*

There will be a continuous wind speed present. The direction and the magnitude of the wind speed can be altered in the *Modify Wind Parameters* menu item. The orientation of the wind speeds is according to the inertial frame. The default wind speed is zero. This means that the vehicle is only subjected to aerodynamic forces only due to its forward speed.

- *Crosswind Gust*

The gust is a side wind speed with an orientation according to the y-axis of the inertial coordinate system. The gust is of a pulse shape. The start time, stop time and DC wind speed can be altered by selecting the *Modify Wind Parameters* menu item.

- *Random Crosswind*

The random crosswind gust is identical to the *Crosswind Gust* as explained above except that the wind speed is randomly distributed. The random speed effect has been obtained by low-pass filtering of white noise. In addition to the start and stop time, the standard deviation and frequency contents of the random component can be changes by selecting the *Modify Wind Parameters* menu item.

- *DC + Random Crosswind*

This option is identical to the *Random Crosswind* except that the random wind speed is superimposed on a DC side wind speed.

- *Modify Wind Parameters*

Depending on the type of wind disturbances chosen, selecting this item will allow the user to modify particular wind parameters such as speed and pulse duration times.

The Roadway Menu

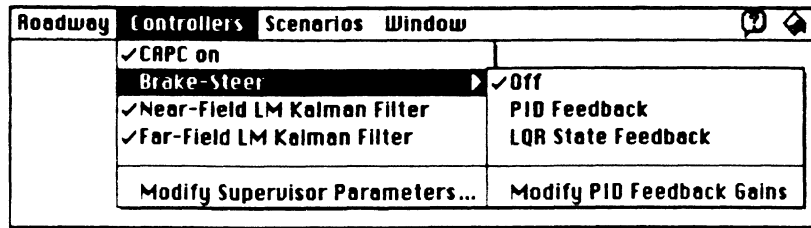
Roadway	Controllers	Scenarios
Straight Road	▶	
Oval Test Track	▶	
Tacom Test Track		
Dana Test Track		
Sinus Road		
ISO Double Lane Change		
Skidpad		
<hr/>		
Season	▶	

The roadway menu enables the user to select different roadway geometries. The following items can be selected:

- *Straight Road*
The road is straight and the user is allowed to choose different values for the grade and superelevation.
- *Oval Test Track*
This roadway geometry corresponds with 1 lap of the FORD Michigan Proving Ground oval. Both turns have the same radius (381 m) and the straight-aways are each 1609 meter long. This oval contains 6 lanes, each with a different superelevation (0, 0.5, 2.5, 6, 15, 28 degrees).
- *Tacom Test Track*
The geometry of this track corresponds with one lap of the Tacom test track in Michigan. It has two tight curves (102 m radius, 0° superelevation) and one larger bend (145 m radius, 8.5° superelevation).
- *Dana Test Track*
The geometry of this track corresponds with the track from the Dana Corporation in southern Michigan. It is an oval with two identical bends (279 meter radius, 6.2° superelevation) at the end of the straight-aways (each 408 meter long).
- *Sinus Road*
This particular roadway has a sine-shape with a wavelength of 200 m and an amplitude of 2 m.
- *ISO Double Lane Change*
This is a free interpretation of the ISO/TR 3888 norm for double lane changes.
- *Skid pad*
The skid pad is a circular track with a radius of 100 m.
- *Seasons*
The season switch enables the user to change the colors of the animated screens according to the 4 seasons of the year.

All roadway geometries can be combined with any of the seven road unevenness types from the *Disturbance* menu.

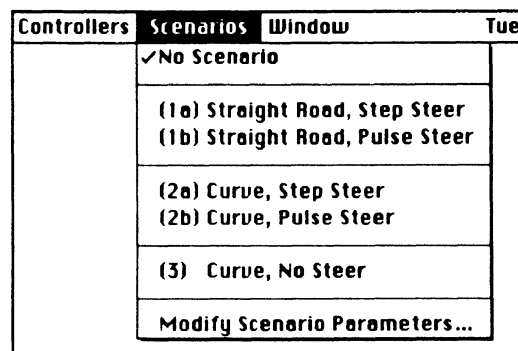
The Controllers Menu



The controller menu controls the specific hard- and software related items for the road departure avoidance (CAPC) system. It has the following items:

- *CAPC*
The switch enables or disables the entire CAPC system. If it is switched off the program will simulate only the vehicle dynamics related part (including a driver model).
- *Brake-Steer*
This menu option allows the used to enable or disable brake-steer control. If it is switched off and the CAPC system switch is on, the entire CAPC system will operate in an open-loop. The brakes will not be activated if a road departure is sensed when brake-steer control has been disabled. Two types of feedback controllers are available: a PID feedback of the lateral position error and a more sophisticated Linear Quadratic state feedback controller which uses the lateral position error, heading angle, side-slip velocity, yaw rate and integral of the lateral position error as inputs. The gains of the PID feedback controller can be changed in the *Modify PID Feedback Gains* option. The LQR gains are hard-coded in the simulation software as a function of the vehicle speed.
- *Near-Field LM Kalman Filter*
Switching this option on means that a Kalman filter will be used to match measured near-field lane marker (LM) data with a roadway geometry model and thus to filter-out noise. If the Kalman filter is disabled, least square curve fitting will be used instead.
- *Far-Field LM Kalman Filter*
Switching this option on means that a Kalman filter will be used to match measured far-field lane marker (LM) data with a roadway geometry model. If the Kalman filter is disabled, least square curve fitting will be used instead.
- *Modify Supervisor Parameters*
The supervisory controller decides whether or not to warn/intervene if a road departure is detected. By selecting this option the user can change the sampling rate and some threshold values related to the decision making module.

The Scenarios Menu

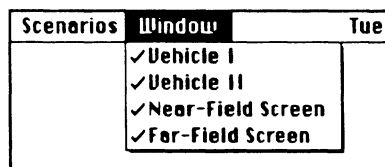


The scenario menu contains some pre-programmed scenarios that will cause a road departure due to a forced steering input. The following selections can be made:

- *No Scenario*
No additional steer input will be generated if this menu item is selected.
- *(1a) Straight Road, Step Steer*
A step steer input will initiate a road departure. The magnitude and timing of the step can be altered in the *Modify Scenario Parameters* menu item. The steering wheel will remain in the non-neutral position.
- *(1b) Straight Road, Pulse Steer*
A pulse steer input will initiate a road departure. The magnitude and width of the pulse can be altered in the *Modify Scenario Parameters* menu item. The steering wheel will be in the neutral position again after the pulse has been applied.
- *(2a) Curve, Step Steer*
This scenario corresponds with scenario (1a) except that the road departure will occur in a curve rather than on a straight-away.
- *(2b) Curve, Pulse Steer*
This scenario corresponds with scenario (1b) except that the road departure will occur in a curve rather than on a straight-away.
- *(3) Curve, No Steer*
The steering wheel is fixed in the neutral position and the vehicle is approaching a curve.
- *Modify Scenario Parameters*
Selecting this item enables the user to change the timing and magnitude of the forced steering wheel input.

Selecting one of the scenarios will not only initiate a road departure, but also other menu options such as *roadway* and *tire model* will be altered depending on the scenario chosen. With all scenarios, the driver model will be disabled once a step/pulse steer is initiated.

The Windows Menu



With the window menu the user is able to enable or disable the animated screen output of the simulation tool.


- *Vehicle I*
This window corresponds with the large rectangular window on the bottom of the screen and shows the vehicle, roadway and camera ranges.
- *Vehicle II*
This window shows a close-up of the vehicle only. The red lines on every corner of the vehicle represent the magnitude of the tire slip forces.
- *Near-Field Screen*
This screen corresponds with the image as seen through the near-field camera.
- *Far-Field Screen*
This screen corresponds with the image as seen through the far-field camera.

The Vehicle I/II windows and the two camera images are updated with different frequencies. The camera images are updated with the scanning frequency of the cameras which can be altered in the *Models*, *Lane Sensor*, *Modify Parameters* menu and the two vehicle images are updated with the

communication interval as set in the *Simulation, Timing* menu. This menu is inactive in the *driving simulator* or *post-processor* mode.

Appendix B

CAPC/LMS Communications Interface Specification

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 1 of 11


DWN	QA
CHK	MFG
MECH	CMPNT
ELECT	RELBL
DFTG	PROJ
Apvd. Customer	

Proprietary Notice

This document contains proprietary information that may not be disclosed to others for any purpose or used for manufacturing purposes without written permission from ERIM. The U.S. Government and other recipients shall not be bound by this restriction to the extent they have by written contract acquired greater rights.

Revisions

Rev.	Description	Date	Approved
1	Initial Release	2/13/95	RDD
2	Corrective Release		RDD
3	Corrective Release	8/24/95	RDD

	Title: Communications Interface Specification	Next Assy:
	Drwg No: 2582003 Revision: 3	Size: A
Code Ident:	Contract: CAPC	Page 2 of 11

Introduction

The CAPC Lane-Mark Sensor (LMS) Processor developed by ERIM communicates with the UMTRI CAPC Processor via an RS-232 line and frame clock. The purpose of this document is to define various aspects of this interface.

The first section characterizes the Frame Clock and the timing that it controls. Section 2 gives the RS-232 communications format.

1.0 The Frame Clock Input


The purpose of this clock is to synchronize the two systems so that the data received from the CAPC system corresponds in time to the image data from the cameras. The Frame Clock input will be supplied to the LMS by the CAPC system.

This signal is a 5 Hz TTL clock with a 50% duty cycle. Upon a transition of the clock, the LMS processor captures a frame from the camera. This frame may or may not be processed, depending on the current mode of the LMS processor (see section 2).

1.1 Timing During Lane-Marker Acquisition Mode

All activity in the LMS System during lane-marker acquisition mode, which is the mode when the LMS system is sending lane-marker data, is directly or indirectly slaved in time to the Frame Clock. This clock triggers the Camera Subsystem to expose a frame of image data. When this image data is completely exposed, it is transferred to the LMS Processor where it is processed. Figure 1 shows the timing of the major processes in the LMS system.

The LMS keeps a variable value called Frame Count. This value is incremented at every rising and falling edge of the Frame Clock during lane-marker acquisition mode only. The pitch & roll information packet and the Lane Geometry information packet are both tagged with the Frame Count for which the data is valid.

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC	Page 3 of 11	

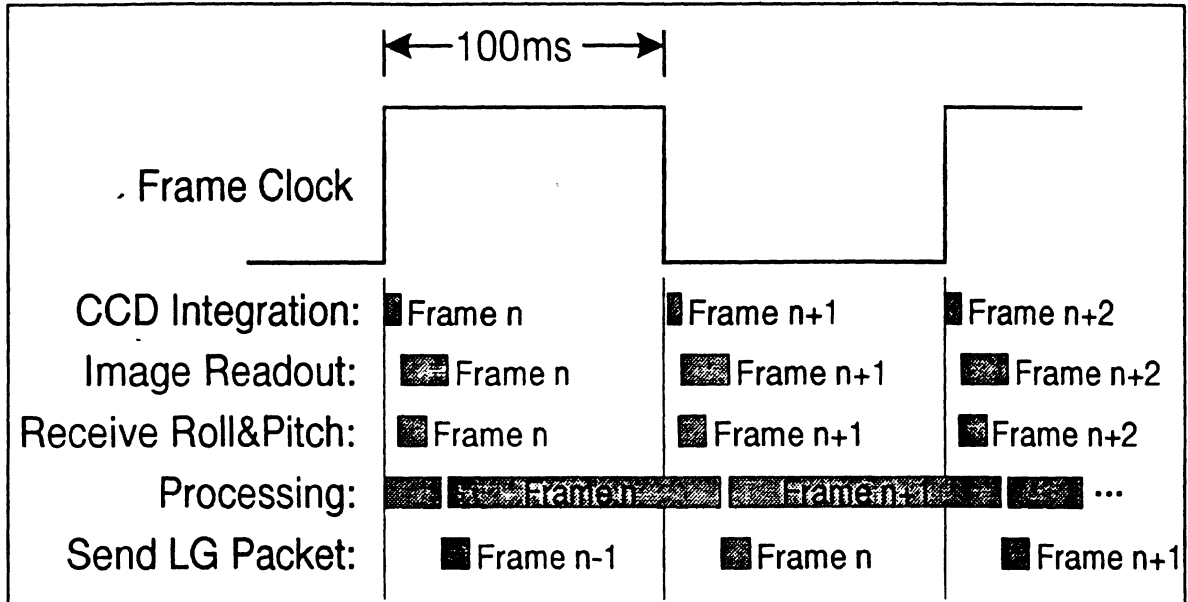


Figure 1. Timing of the LMS system.

2.0 The RS-232 Communications


Control and data information transfer between the LMS and CAPC Processors is in the form of packets. All packets have the same general format shown below:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh ¹
Packet Type	1-Byte	00h-A9h ABh - FFh
Data Length	2-Byte Unsigned Integer	0000h - FFFFh
Data	Defined in Section 2.2	
Checksum	1-Byte	00h - FFh

The following two sections outline the contents of the packet's fields.

2.1 The Header and Checksum Information

¹ h at the end of a number denotes hexadecimal format.

	Title: Communications Interface Specification	Next Assy:
	Drwg No: 2582003 Revision: 3	Size: A
Code Ident:	Contract: CAPC	Page 4 of 11

The first three fields (Packet Marker, Packet Type, and Data Length) make up a packet's header, and the last byte of the packet is a checksum of the whole packet. The definitions of these fields are common among all packet data types discussed in section 2.2.

Packet Marker

The Packet Marker always has the value of AAh. The purpose of this is to mark the beginning of a packet independent of context. To accomplish this, this byte value, AAh, must be distinguishable from other AAh bytes that may appear in all other fields of the packet. This is accomplished by the swapping of all occurrences of the byte AAh not in the Packet Marker field with the sequence of two AAh bytes (AAAAh) prior to transmit and following receive of data on the RS-232. An occurrence of AAh without a following AAh on the input data stream should be interpreted as a packet marker.

Packet Type


This 1-byte field identifies the type of information that is contained in the Data field. The value of this field is defined along with the Data field in section 2.2.

Data Length

The number of bytes of the 'Data' field is given in the 'Data Length' field. Any AAh bytes occurring in the packet must be handled as outlined in the Packet Marker description, but it should only count as one byte in the Data Length (i.e., a AAh replaced by AAAAh still only counts as one byte in the Data Length field).

Checksum

This byte is a no-carry sum of all bytes that preceded this field in the packet. Though, no error correction scheme is to be implemented, an error in checksum will void all data, and will increase the checksum error count in the STATUS packet. The checksum should be calculated without the doubling of AAh bytes, as discussed in the Packet Marker description above. Note, an occurrence of an AAh checksum will be transmitted as AAAAh

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 5 of 11

2.2 Data Field


Two types of Packet Types will be sent to the LMS system: asynchronous and synchronous. Asynchronous packets can be sent anytime when the LMS is not in lane-maker acquisition mode (see the START packet type). Synchronous packets are sent to the LMS only when the LMS system is receiving a frame clock and is in lane-marker acquisition mode. Synchronous packets are synchronized with the frame clock input. A summary of packet types are given below:

<i>Packet Name</i>	<i>Packet-Type</i>	<i>Sync/Async</i>	<i>Originator</i>
INITIALIZE	01h	Async	CAPC System
CALIBRATE	02h	Async	CAPC System
PROBE	03h	Async	CAPC System
STATUS	81h	Async	LMS System
START	04h	Async	CAPC System
STOP	05h	Async	CAPC System
P&R	10h	Sync	CAPC System
LG	91h	Sync	LMS System

The INITIALIZE Packet

This asynchronous-mode packet initializes the LMS system, and any previous operation parameters are flushed. The Frame Count is also reset to the value 00h. The packet format is:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	01h
Data Length	2-Byte Unsigned Integer	0005h
Checksum	1-byte	B1h

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 6 of 11

The CALIBRATE Packet

This asynchronous-mode packet initiates a calibration routine in the LMS System. The packet format is as follows:


<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	02h
Data Length	2-Byte Unsigned Integer	0001h
Restore	1-byte	00h or 01h
Checksum	1-byte	B3h - B4h

Restore field defines the LMS if it should restore the last calibration stored (Restore=01h), or actually begin the calibration procedure outlined in Calibration Routine Specification (Restore=00h). If Restore is 00h, the LMS system assumes that all prerequisites for a calibration are satisfied.

The PROBE Packet

This packet stimulates the LMS system to return the STATUS packet. This packet can be sent anytime, but excessive use of this in lane-marker acquisition mode may cause a delay in lane-marker data delivery. The format of this packet follows:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	03h
Data Length	2-Byte Unsigned Integer	0005h
Checksum	1-byte	B2h

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 7 of 11


The STATUS Packet

This packet is sent by the LMS system to report the status and health of the LMS sensor. Two conditions results in this packet being sent: 1) it was requested via a previous packet sent to the LMS system or 2) a error occurred in the operation. The packet has the following format:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	81h
Data Length	2-Byte Unsigned Integer	0005h
Initialized	1-byte	00h or 01h
Calibrated	1-byte	00h or 01h
Camera Health	1-byte	00h or 01h
Frame Count	1-byte	00h-FFh
Error	1-byte	00h-FFh
LMS Mode	1-byte	00h-FFh
Checksum	1-byte	00h-FFh

The data returned is defined as:

- Initialized: 00h = Not previously initialized via INITIALIZE Packet
01h = Initialized
- Calibrated: 00h = Not successfully calibrated previously via CALIBRATE Packet
01h = Calibrated
- Camera Health: 00h = Camera-Subsystem requires re-calibration
01h = Camera-Subsystem passed all operational tests
- Frame Count: The next Frame Count value
- Error: 00h = LMS system is operating normally
01h = LMS system is not receiving Frame Clock
02h-FEh = Currently undefined error codes.
FFh = General Fatal Error. re-initialize and re-calibrate
- LMS Mode 00h = Not in Acquisition Mode
01h = In Acquisition Mode

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 8 of 11

If the LMS system reports a Error, it will not send any lane-marker data to the CAPC system. Depending on the error, the LMS may have to be re-initialized and/or re-calibrated.

The START Packet


This asynchronous-mode packet puts the LMS system to lane-marker acquisition mode, which is the mode that starts the LMS to process the camera images and returns lane-marker data. While in the lane-marker acquisition mode, the only synchronous-type packets and asynchronous-type packets STOP and PROBE should be sent to the LMS system. The START Packet has the following format:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	04h
Data Length	2-Byte Unsigned Integer	0000h
Checksum	1-byte	B3h

The STOP Packet

This asynchronous-mode packet takes the LMS system out of lane-marker acquisition mode. After the LMS system receives this, it will accept all asynchronous-mode packets, and it will not send or process any synchronous packets.

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	05h
Data Length	2-Byte Unsigned Integer	0005h
Checksum	1-byte	B4h

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 9 of 11

The P&R Packet

This synchronous-mode packet, sent from the CAPC system, contains Pitch and Roll data. The format of this packet is as follows:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	10h
Data Length	2-Byte Unsigned Integer	0005h
Frame Count	1-Byte	00h - FFh
Pitch	2-byte Signed Integer	0000h - FFFFh
Roll	2-byte Signed Integer	0000h - FFFFh
Velocity	2-byte Unsigned Integer	0000h - FFFFh
Checksum	1-byte	00h-FFh


The Frame Count identifies the frame for which the pitch and roll data is valid.

The Pitch and Roll data is reported in 1/100th of a degree, i.e.,

Pitch in degrees = 'Pitch' field / 100 and

Roll in degrees = 'Roll' field / 100.

The Velocity data is reported in XXX units.

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 10 of 11

The LG Packet

This synchronous-mode packet, sent from the CAPC system, contains lane-marker data. The format of this packet is as follows:

<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
Packet Marker	1-Byte	AAh
Packet Type	1-Byte	91h
Data Length	2-Byte Unsigned Integer	Variable
Frame Count	1-Byte	00h-FFh
# Records	1-byte	00h - 02h
Lane Data Records	see below	defined below
Checksum	1-byte	00h-FFh

The Frame Count identifies the frame for which the pitch and roll data is valid.


The # Records field contains the number of Lane Data Records that the packet contains. Each Lane Data Record has the following format:

<i>Field</i>	<i>Data Size</i>	<i>Valid Values</i>
Lane-Type	1-Byte	see below.
ID	1-Byte	00h - FFh
# Data-Point Records	1-Byte	00h - 20h
Data-Point Records	4-bytes x # of Records	defined below

The ID field is the identification code that is assigned to the lane by the LMS. As lanes are acquired, it is assigned an 1-byte ID and this ID is kept until it is no longer tracked by the LMS system.

The lane-type is defined as follows:

- 00h : Solid
- 01h: Dashed

	Title: Communications Interface Specification		Next Assy:
	Drwg No: 2582003	Revision: 3	Size: A
Code Ident:	Contract: CAPC		Page 11 of 11

The # of Data-Point Records field contains the number of data-points associated with the lane. Each Data-Point Record has the following format:


<i>Field</i>	<i>Data Format</i>	<i>Valid Values</i>
X Location	2-Byte Unsigned Integer	0000h - FFFFh
Y Location	2-byte Signed Integer	0000h - FFFFh

The X location values are in the units of centimeters, e.g.,
 $X \text{ in meters} = (X \text{ location reported}) / 100.$

The Y location values are in the units of centimeters, e.g.,
 $Y \text{ in meters} = (Y \text{ location reported}) / 100.$

Appendix C

CAPC/LMS Mechanical and Electrical Interface Specification

	Title: Lane-Mark-Sensor Mechanical & Electrical Interface Specification		Next Assy:
	Drwg No: 2582001	Revision: 4	Size: A
Code Ident:	Contract: CAPC		Page 1 of 6


DWN	QA
CHK	MFG
MECH	CMPNT
ELECT	RELBL
DFTG	PROJ
Apvd. Customer	

Proprietary Notice

This document contains proprietary information that may not be disclosed to others for any purpose or used for manufacturing purposes without written permission from ERIM. The U.S. Government and other recipients shall not be bound by this restriction to the extent they have by written contract acquired greater rights.

Revisions			
Rev.	Description	Date	Approved
1	Draft Release	1/30/95	RDD
2	Initial Release to UMTRI	2/95	RDD
3	Corrective Release	5/1/95	RDD
4	Corrective Release	8/23/95	RDD

--	--	--	--

	Title: Lane-Mark-Sensor Mechanical & Electrical Interface Specification		Next Assy:
	Drwg No: 2582001	Revision: 4	Size: A
Code Ident:	Contract: CAPC		Page 2 of 6

Introduction

The CAPC Lane-Mark Sensor (LMS) developed by ERIM consists of two main parts: the Camera Subsystem, and the LMS Processor. The system block diagram is shown in figure 1.

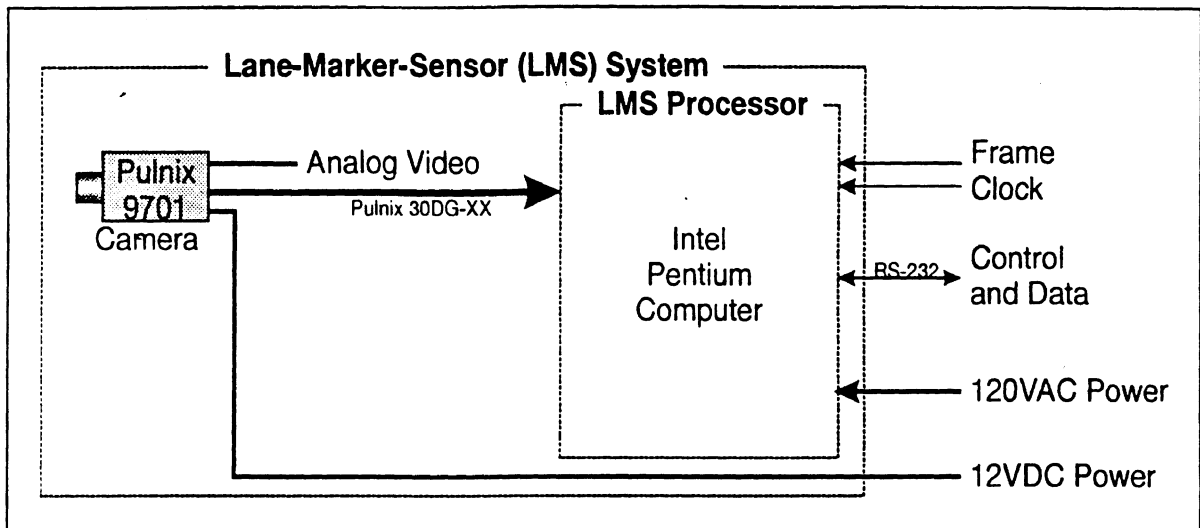


Figure 1. Detailed Block Diagram of the Lane-Marker Sensor System


The following sections discuss the mechanical and electrical requirements of the LMS subsystems.

1.0 The CCD Camera

The LMS system contains a Pulnix TM9701 CCD digital camera. The exact camera position and mounting in the car is determined in accordance to the following mechanical and electrical requirements.

1.1 Mechanical Requirements

The camera should be mounted in the test car's interior space as high and as close to the windshield as possible while creating an unobstructed forward-looking view. The camera's dimensions (without lenses) is shown in Figure 2. An additional amount of space should be allocated for a lens, which mount directly to the front of the camera (see Figure 2). The lens used does not exceed 43mm x 48mm (diameter x length) in size.

	Title: Lane-Mark-Sensor Mechanical & Electrical Interface Specification		Next Assy:
	Drwg No: 2582001	Revision: 4	Size: A
Code Ident:	Contract: CAPC		Page 3 of 6

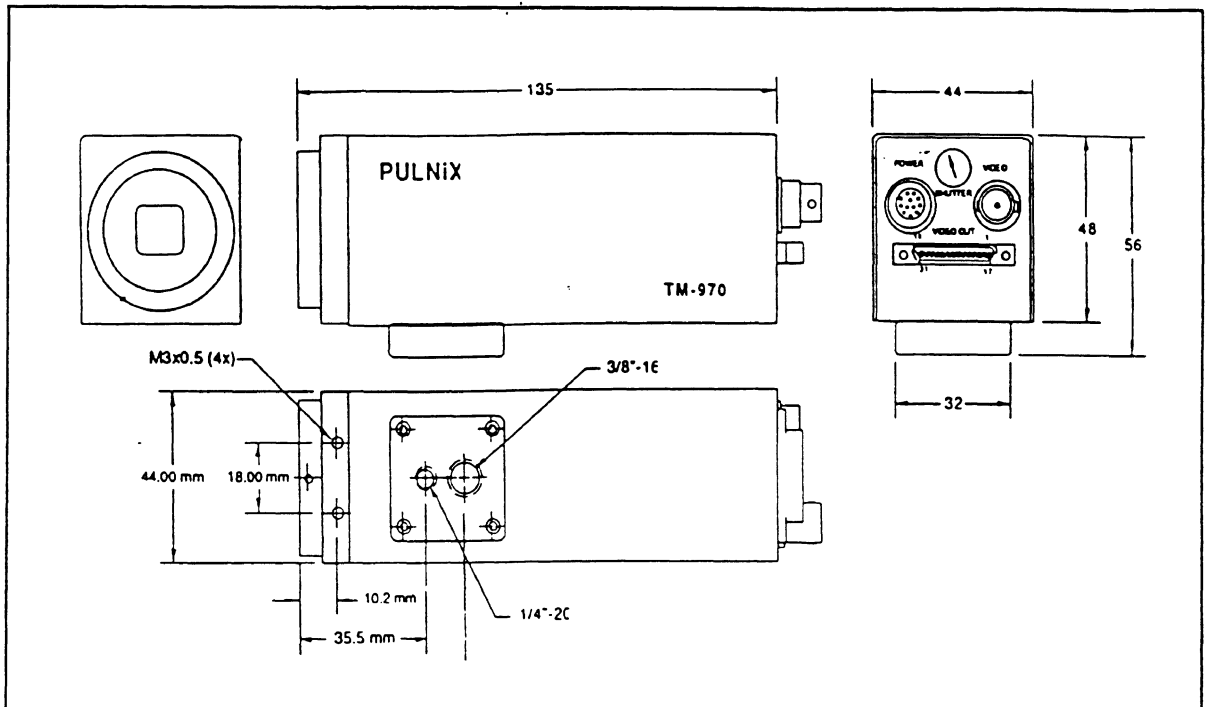



Figure 2. The Pulnix TM9701 CCD Camera's Physical Dimensions

The camera is connected to the other parts of the system via the interconnections shown in Figure 3 and the following table.

<i>Qty.</i>	<i>Cable Type Required</i>	<i>Data Type</i>	<i>Cable Diameter</i>	<i>Routing Destination</i>
1	2-Conductor 24AWG	12VDC Power (see section 1.2)		UMTRI Power Supply
1	75-Ohm Coax	RS-170 Video		UMTRI Spec'd (documentation use)
1	Pulnix 30DG-XX Digital Cable	RS-422 and TTL Digital	8mm	LMS Processor

	Title: Lane-Mark-Sensor Mechanical & Electrical Interface Specification		Next Assy:
	Drwg No: 2582001	Revision: 4	Size: A
Code Ident:	Contract: CAPC	Page 4 of 6	

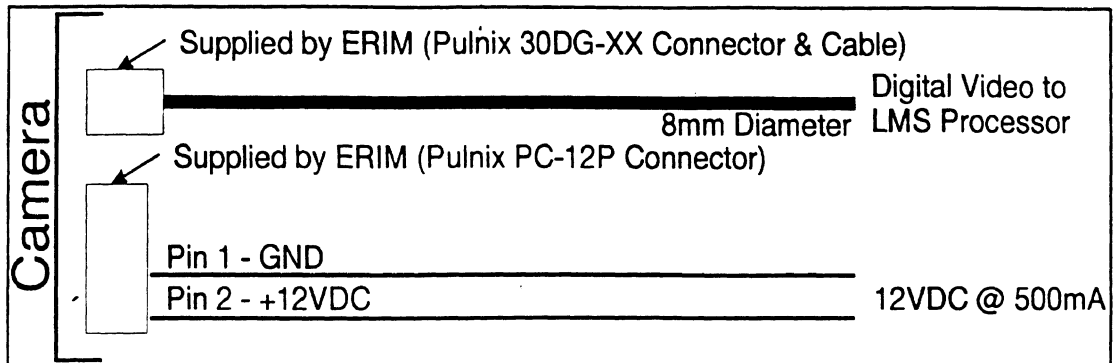


Figure 3. Camera Interconnection Requirements

1.2 Electrical Requirements

The camera must be supplied with 10.8-14.2 Volts DC at 500 milliamps. The power is delivered to the camera through a Pulnix PC-12P connector via pin 1 (ground) and pin 2 (+VDC). The connector is supplied as part of the LMS System, but the power source is **not** supplied.


All required camera signals are incorporated into the Pulnix 30DG-XX Digital Cable, which is connected to the LMS Processor. A RS-170 video signal is available from the camera, via the BNC connector on the back of the camera, for documentation and debugging uses. This signal, however, will only be active when the LMS System is **not** operating. Due to this limitation, the RS-170 signal is not be used by the LMS System itself.

1.3 The Camera Lens

The camera lens is a 12.5mm focal length 2/3" C-mount f/1.7 Cosmical lens. This size was chosen to give adequate field-of-view when mated with the Pulnix camera.

2.0 The LMS Processor

The LMS Processor captures the data coming from the CCD camera using a MuTech MV-1000 commercial frame grabber with a MV-1100 digital input daughterboard. The camera information is then processed real-time to produce lane-marker data. All communications with the Processor is accomplished via a bi-directional RS-232 connection and a frame clock input, as discussed in detail in the Communications Interface Specification, ERIM document #2582003.

	Title: Lane-Mark-Sensor Mechanical & Electrical Interface Specification		Next Assy:
	Drwg No: 2582001	Revision: 4	Size: A
Code Ident:	Contract: CAPC		Page 5 of 6

2.1 Mechanical Requirements

The LMS Processor is a standard PC-class Pentium computer with the approximate size of 18cm x 46cm x 44cm (height x width x depth). A space of 10cm should be reserved behind the processor for the cable inputs. No mounting provisions are supplied.

The LMS processor requires the backpanel interconnections shown in Figure 4 and the following chart:

<i>Qty</i>	<i>Cable Type Required</i>	<i>Data Type</i>	<i>Diameter</i>	<i>Routing Destination</i>
1	Standard U.S. 3-prong power plug	120VAC Power	7mm	UMTRI's Power Inverter
2	Pulnix 30DG-XX Digital Cable	RS-422 & TTL Digital	8mm	Camera Subsystem
1	RS-232 (9-Pin Sub-D Female Connector).	RS-232 Digital	UMTRI-Spec'd	CAPC Processor
2	Twisted Pair (9-pin Sub-D Male Connectors - pin 5: clock, pin 9: signal ground)	Frame Clock	UMTRI-Spec'd	CAPC Processor


2.2 Electrical Requirements

The LMS Processor requires 120±10% Volts at 50-60Hz AC at no more than 100 watts consumption. This power is **not** supplied by the LMS System.

A frame clock input from the main CAPC Processor is required for operation of the LMS System. This clock should be a ~5Hz TTL clock with the 50% duty cycle. The leading- and trailing-edge of this clock corresponds to the sampling time of the next incoming vehicle-dynamics data supplied to the LMS Processor by the CAPC System.

The control and data information is transferred between the LMS and CAPC system using a standard full-duplex RS-232 link with the following characteristics:

- 57,600 baud
- 8 data bits
- No parity
- 1 stop bit
- No hardware or software handshaking

	Title: Lane-Mark-Sensor Mechanical & Electrical Interface Specification		Next Assy:
	Drwg No: 2582001	Revision: 4	Size: A
Code Ident:	Contract: CAPC	Page 6 of 6	

The format of the information transferred via the RS-232 link is specified in the Communications Interface Specification, ERIM document #2582003.

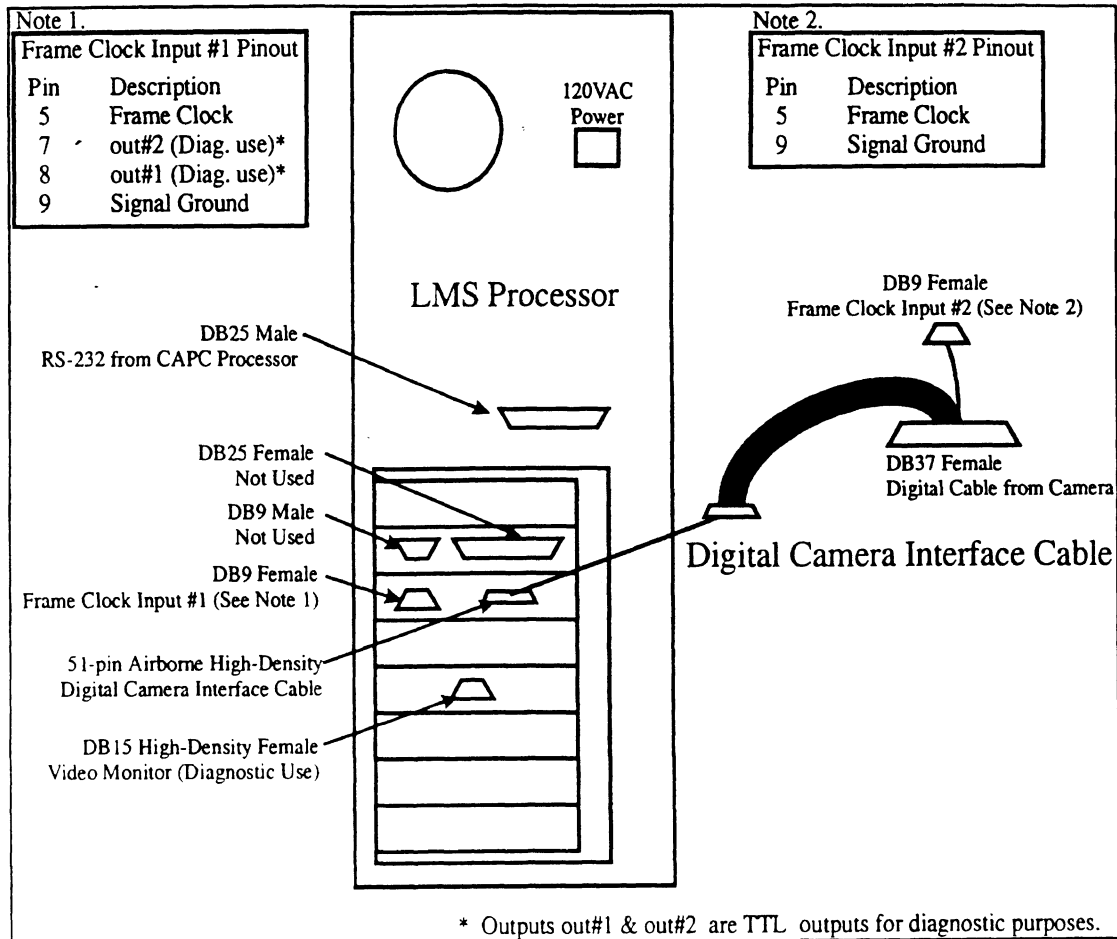


Figure 4. LMS Processor Interconnections

Appendix D

Computer code listing:
CAPC Prototype Control Code

CAPC Prototype Control Code listings follow for these files:

BrakeSteer.cp
BrakeSteer.h
CAPCcycler.cp
CAPCcycler.h
CAPCinit.cp
CAPCinit.h
KalmanF.cp
KalmanF.h
KalmanN.cp
KalmanN.h
LMS.c
LMS.h
MathSupport.cp
MathSupport.h
PathPrediction.c
PathPrediction.h
StringUtils.c

StringUtils.h
Supervisor.cp
Supervisor.h
SupervisorDlg.cp
TLCalculation.cp
TLCalculation.h
Vehicle.h
XYWinSetup.c LG
BGWinSetup.c
BrakePressure.cp
BrakePressure.h
BrakeToolsDlg.cp
BrakeToolsDlg.hp
LMSSerial.cp
LMSSerial.h
LMSToolsDlg.cp
LMSToolsDlg.hp

BrakeSteer.ccp

```

// CAPC Controller Code: Brake Steer Control
// Last Update: 29 March 1995 (JV)
#include "BrakeSteer.h"
#include "math.h"
#include "Kalmant.h"
#include "MathSupport.h"

//----- FUNCTION: BrakeSteer -----*
void BrakeSteer(void)
{
    double offset;

    // Pa_1L = command brake pressure, left front wheel [Pa]
    // Pa_1R = command brake pressure, right front wheel [Pa]
    // Pa_2L = command brake pressure, left rear wheel [Pa]
    // Pa_2R = command brake pressure, right rear wheel [Pa]
    //----- Enable brake steering -----*/
    if ((|BrakeSteer| != |Bsoff|) && (Brake_cn)) // Switch off cruise control
    {
        // (mass_lanePost == 1) // Right roadedge is
        // (mass_lanePost == 2) // Left roadedge is
        //----- PID control -----*/
        switch (|BrakeSteer|)
        {
            case (|Bspid|): // Brake steer based on PID feedback
                // Ep = position error
                // Ei = integral of position error [m/a]
                // Ed = derivative of position error [m/a]
                Ep = Xkf_pred[2] + offset;
                Ei = C_Fai * a_v / ((Tv2 * BrakeGain2) + (4.0 * C_Fai * a_v));
                Ed = Kkf_Dpred[0] * coe(Xkf_pred[3]);
                P_brake = - Kp_bac * Ep - Ki_bac * Ei - kd_bac * Ed;
                // Brake steer based on IQR state feedback
                case (|Bsig|): // Brake steer based on IQR state feedback
                    P_brake = - Kfb_bac[0] * Xkf_pred[0] - Kfb_bac[1] * Xkf_pred[1]
                        - Kfb_bac[2] * Xkf_pred[2] - Kfb_bac[3] * Xkf_pred[3]
                        - Kfb_bac[4] * Xkf_pred[4]
                        - (mass_delta_fw - delta_kln);
                    //----- Calculate the brake steer feedback gains as a function of vehicle speed -----*/
                    for (i = 0; i < nfb_bac; i++)
                        Kfb_bac[i] = Pfb_bac[i][0] * mass_u_car + Pfb_bac[i][1] * mass_u_car + Pfb_bac[i][2] * mass_u_car + Pfb_bac[i][3] * mass_u_car + Pfb_bac[i][4] * mass_u_car;
                    // These are the polynomial coefficients of the Brake Steer Controller feedback
                    // a gains as a function of the vehicle speed (m/s) (3rd-order fit)
                    Pfb_bac[0][0] = 1.3281e+01;
                    Pfb_bac[0][1] = -2.4024e+03;
                    Pfb_bac[0][2] = 1.4856e+05;
                    Pfb_bac[0][3] = 5.1638e+03;
                    Pfb_bac[1][0] = 9.6382e+04;
                    Pfb_bac[1][1] = -1.0737e+06;
                    Pfb_bac[1][2] = 2.0971e+06;
                    Pfb_bac[1][3] = 3.0971e+06;
                    Pfb_bac[2][0] = 3.0339e+02;
                    Pfb_bac[2][1] = -3.1774e+04;
                    Pfb_bac[2][2] = 1.0901e+06;
                    Pfb_bac[2][3] = -3.0564e+03;
                    Pfb_bac[3][0] = 3.2809e+02;
                    Pfb_bac[3][1] = -8.0159e+06;
                    Pfb_bac[3][2] = 2.5798e+08;
                    Pfb_bac[3][3] = 9.4690e+01;
                    Pfb_bac[4][0] = -1.0494e+02;
                    Pfb_bac[4][1] = -2.2144e+00;
                    Pfb_bac[4][2] = -2.2144e+00;
                    Pfb_bac[4][3] = -2.2144e+00;
                    for (i = 0; i < nfb_bac; i++)
                        Kfb_bac[i] = 1.0;
                    // Initialize Brake Steer Feedback Matrix
                    // Compute Brake Steer Feedback Matrix
                }
            }
        }
    }
}

```



```

*%2_L = LaneF_poli[0];
*%0_R = LaneF_poli[1];
*%1_L = LaneF_poli[2];
*%1_R = LaneF_poli[3];
*Warning = Warn_on;

// If 0 // ---- make some coords avail for viewing
*%0_R = g1c.Mdata[0]; // numpts in
right If ( g1c.Mdata[0] > 2 ) { // numpts in
    *%1_R = g1c.Y[ 0 ][ 1 ]; //
second coordinate *%2_R = g1c.Y[ 0 ][ g1c.Mdata[0]-1 ]; // last coordinate
} else { *%1_R = -1e9; *%2_R = -1e9; }

*%0_L = g1c.Mdata[1]; // numpts in
left If ( g1c.Mdata[1] > 2 ) { // numpts in
    *%1_L = g1c.Y[ 1 ][ 1 ]; //
second coordinate *%2_L = g1c.Y[ 1 ][ g1c.Mdata[1]-1 ]; // last coordinate
} else { *%1_L = +1e9; *%2_L = +1e9; }

#endif

/----- Turn on the brakes -----/
If ( brakeEnabled & 0.0 )
    If ( brakePressure < 0.0 ) {
        also ExecutBrakePressure( 0, (int)( Pa_2k/10.0 ) );
    }
    If ( Pa_2k > 0.0010 )
        also ExecutBrakePressure( 1, (int)( Pa_2k/10.0 ) );
    also ExecutBrakePressure( -1, BRAKES_OFF );
}

// -- store internal data -> circular buffer
If ( !firstLoop ) {
    digital_output(1, ON, serris);
    for ( int i = 0; i < 10; i++ ) {
        for ( int j = 0; j < 10; j++ ) {
            *%i+j = (int) ( InOut[ i+j ] );
        }
        *%i = (int) ( InOut[ i ] );
        *%i = (int) ( InOut[ i ] );
        *%i = (int) ( InOut[ i ] );
    }
}

// ---- show the loop exit
digital_output(CAPC_DOUT_TV0, OFF, serris);
return( 1 );
}

```

CAPCCYCLER.H

```

/* Macro DAS 3.0 - CAPCCYCLER.H */

#finddef _g9j_cyclcr
#finddef _g9j_cyclcr
#finddef EXT EXT extern
#endif

/*-----*/
/*
/* CONSTANTS */
/*-----*/
/*
/*

```

```

/* STRUCTURES */
/*-----*/
/*
/* GLOBALS */
/*-----*/
/*
/* PROTOTYPES */
/*-----*/
/*
/* OneControllerCycle( int thcn )
*/
#endif

```

CAPCINIT.CP

```

/* CAPC Controller Code: CAPC Initialization */
/* Last Update: 29 March 1995 (PV) */
#include "CAPCInit.h"
#include "Kalmann.h"
#include "KalmannF.h"
#include "Supervisor.h"
#include "BrakeSteer.h"
#include "Vehicle.h"

/*-----*/
void Init_CAPC(void)
{
    Init_LMS(); // Init LMS sensor

    // Get first image here!
    GetMSdata(); // Get first batch of LMS signals
    Init_Kalmann(20,0); // Initialize Far-Field Kalmann filter
    Init_Kalmann(); // Initialize Near-Field Kalmann filter
    Init_BrakeSteer(); // Initialize Brake Steer Control
    Init_Supervisor(); // Initialize Supervisor
    fireTriVecAPC = false;
    !KalmannFar = true;
    !KalmannFar = true;

    /*
    *%0 = car = 3.0;
    *%1 = car = 0.0;
    *%2 = car = 0.0;
    *%3 = delta_tv = 0.0;
    *%4 = delta_tv = 0.0;
    *%5 = P_brake = 0.0;
    *%6 = OldAngle = 0.0;
    *%7 = HeadAngle_est = 0.0;
    *%8 = LatDev_est = 0.5 * LatDev(1rh);
    *%9 = OldLatDev_est = LatDev_est;
    *%10 = LanePos = 1;
    *%11 = LanePos = 1;
    *%12 = LanePos = 1;
    *%13 = LanePos = 1;
    *%14 = LanePos = 1;
    *%15 = LanePos = 1;
    *%16 = LanePos = 1;
    *%17 = LanePos = 1;
    *%18 = LanePos = 1;
    *%19 = LanePos = 1;
    *%20 = LanePos = 1;
    *%21 = LanePos = 1;
    *%22 = LanePos = 1;
    *%23 = LanePos = 1;
    *%24 = LanePos = 1;
    *%25 = LanePos = 1;
    *%26 = LanePos = 1;
    *%27 = LanePos = 1;
    *%28 = LanePos = 1;
    *%29 = LanePos = 1;
    *%30 = LanePos = 1;
    *%31 = LanePos = 1;
    *%32 = LanePos = 1;
    *%33 = LanePos = 1;
    *%34 = LanePos = 1;
    *%35 = LanePos = 1;
    *%36 = LanePos = 1;
    *%37 = LanePos = 1;
    *%38 = LanePos = 1;
    *%39 = LanePos = 1;
    *%40 = LanePos = 1;
    *%41 = LanePos = 1;
    *%42 = LanePos = 1;
    *%43 = LanePos = 1;
    *%44 = LanePos = 1;
    *%45 = LanePos = 1;
    *%46 = LanePos = 1;
    *%47 = LanePos = 1;
    *%48 = LanePos = 1;
    *%49 = LanePos = 1;
    *%50 = LanePos = 1;
    *%51 = LanePos = 1;
    *%52 = LanePos = 1;
    *%53 = LanePos = 1;
    *%54 = LanePos = 1;
    *%55 = LanePos = 1;
    *%56 = LanePos = 1;
    *%57 = LanePos = 1;
    *%58 = LanePos = 1;
    *%59 = LanePos = 1;
    *%60 = LanePos = 1;
    *%61 = LanePos = 1;
    *%62 = LanePos = 1;
    *%63 = LanePos = 1;
    *%64 = LanePos = 1;
    *%65 = LanePos = 1;
    *%66 = LanePos = 1;
    *%67 = LanePos = 1;
    *%68 = LanePos = 1;
    *%69 = LanePos = 1;
    *%70 = LanePos = 1;
    *%71 = LanePos = 1;
    *%72 = LanePos = 1;
    *%73 = LanePos = 1;
    *%74 = LanePos = 1;
    *%75 = LanePos = 1;
    *%76 = LanePos = 1;
    *%77 = LanePos = 1;
    *%78 = LanePos = 1;
    *%79 = LanePos = 1;
    *%80 = LanePos = 1;
    *%81 = LanePos = 1;
    *%82 = LanePos = 1;
    *%83 = LanePos = 1;
    *%84 = LanePos = 1;
    *%85 = LanePos = 1;
    *%86 = LanePos = 1;
    *%87 = LanePos = 1;
    *%88 = LanePos = 1;
    *%89 = LanePos = 1;
    *%90 = LanePos = 1;
    *%91 = LanePos = 1;
    *%92 = LanePos = 1;
    *%93 = LanePos = 1;
    *%94 = LanePos = 1;
    *%95 = LanePos = 1;
    *%96 = LanePos = 1;
    *%97 = LanePos = 1;
    *%98 = LanePos = 1;
    *%99 = LanePos = 1;
    */
    RollIncl1 = (0.5 * Tv1 + Tv1 + c_rv1) / (0.5 + c_rv1);
    RollIncl2 = (0.5 * Tv2 + Tv2 + c_rv2) / (0.5 + c_rv2);
    RollIncl3 = (0.5 * Tv3 + Tv3 + c_rv3) / (0.5 + c_rv3);
    RollIncl4 = (0.5 * Tv4 + Tv4 + c_rv4) / (0.5 + c_rv4);
    RollIncl5 = (0.5 * Tv5 + Tv5 + c_rv5) / (0.5 + c_rv5);
    RollIncl6 = (0.5 * Tv6 + Tv6 + c_rv6) / (0.5 + c_rv6);
    RollIncl7 = (0.5 * Tv7 + Tv7 + c_rv7) / (0.5 + c_rv7);
    RollIncl8 = (0.5 * Tv8 + Tv8 + c_rv8) / (0.5 + c_rv8);
    RollIncl9 = (0.5 * Tv9 + Tv9 + c_rv9) / (0.5 + c_rv9);
    RollIncl10 = (0.5 * Tv10 + Tv10 + c_rv10) / (0.5 + c_rv10);
    RollIncl11 = (0.5 * Tv11 + Tv11 + c_rv11) / (0.5 + c_rv11);
    RollIncl12 = (0.5 * Tv12 + Tv12 + c_rv12) / (0.5 + c_rv12);
    RollIncl13 = (0.5 * Tv13 + Tv13 + c_rv13) / (0.5 + c_rv13);
    RollIncl14 = (0.5 * Tv14 + Tv14 + c_rv14) / (0.5 + c_rv14);
    RollIncl15 = (0.5 * Tv15 + Tv15 + c_rv15) / (0.5 + c_rv15);
    RollIncl16 = (0.5 * Tv16 + Tv16 + c_rv16) / (0.5 + c_rv16);
    RollIncl17 = (0.5 * Tv17 + Tv17 + c_rv17) / (0.5 + c_rv17);
    RollIncl18 = (0.5 * Tv18 + Tv18 + c_rv18) / (0.5 + c_rv18);
    RollIncl19 = (0.5 * Tv19 + Tv19 + c_rv19) / (0.5 + c_rv19);
    RollIncl20 = (0.5 * Tv20 + Tv20 + c_rv20) / (0.5 + c_rv20);
    RollIncl21 = (0.5 * Tv21 + Tv21 + c_rv21) / (0.5 + c_rv21);
    RollIncl22 = (0.5 * Tv22 + Tv22 + c_rv22) / (0.5 + c_rv22);
    RollIncl23 = (0.5 * Tv23 + Tv23 + c_rv23) / (0.5 + c_rv23);
    RollIncl24 = (0.5 * Tv24 + Tv24 + c_rv24) / (0.5 + c_rv24);
    RollIncl25 = (0.5 * Tv25 + Tv25 + c_rv25) / (0.5 + c_rv25);
    RollIncl26 = (0.5 * Tv26 + Tv26 + c_rv26) / (0.5 + c_rv26);
    RollIncl27 = (0.5 * Tv27 + Tv27 + c_rv27) / (0.5 + c_rv27);
    RollIncl28 = (0.5 * Tv28 + Tv28 + c_rv28) / (0.5 + c_rv28);
    RollIncl29 = (0.5 * Tv29 + Tv29 + c_rv29) / (0.5 + c_rv29);
    RollIncl30 = (0.5 * Tv30 + Tv30 + c_rv30) / (0.5 + c_rv30);
    RollIncl31 = (0.5 * Tv31 + Tv31 + c_rv31) / (0.5 + c_rv31);
    RollIncl32 = (0.5 * Tv32 + Tv32 + c_rv32) / (0.5 + c_rv32);
    RollIncl33 = (0.5 * Tv33 + Tv33 + c_rv33) / (0.5 + c_rv33);
    RollIncl34 = (0.5 * Tv34 + Tv34 + c_rv34) / (0.5 + c_rv34);
    RollIncl35 = (0.5 * Tv35 + Tv35 + c_rv35) / (0.5 + c_rv35);
    RollIncl36 = (0.5 * Tv36 + Tv36 + c_rv36) / (0.5 + c_rv36);
    RollIncl37 = (0.5 * Tv37 + Tv37 + c_rv37) / (0.5 + c_rv37);
    RollIncl38 = (0.5 * Tv38 + Tv38 + c_rv38) / (0.5 + c_rv38);
    RollIncl39 = (0.5 * Tv39 + Tv39 + c_rv39) / (0.5 + c_rv39);
    RollIncl40 = (0.5 * Tv40 + Tv40 + c_rv40) / (0.5 + c_rv40);
    RollIncl41 = (0.5 * Tv41 + Tv41 + c_rv41) / (0.5 + c_rv41);
    RollIncl42 = (0.5 * Tv42 + Tv42 + c_rv42) / (0.5 + c_rv42);
    RollIncl43 = (0.5 * Tv43 + Tv43 + c_rv43) / (0.5 + c_rv43);
    RollIncl44 = (0.5 * Tv44 + Tv44 + c_rv44) / (0.5 + c_rv44);
    RollIncl45 = (0.5 * Tv45 + Tv45 + c_rv45) / (0.5 + c_rv45);
    RollIncl46 = (0.5 * Tv46 + Tv46 + c_rv46) / (0.5 + c_rv46);
    RollIncl47 = (0.5 * Tv47 + Tv47 + c_rv47) / (0.5 + c_rv47);
    RollIncl48 = (0.5 * Tv48 + Tv48 + c_rv48) / (0.5 + c_rv48);
    RollIncl49 = (0.5 * Tv49 + Tv49 + c_rv49) / (0.5 + c_rv49);
    RollIncl50 = (0.5 * Tv50 + Tv50 + c_rv50) / (0.5 + c_rv50);
    RollIncl51 = (0.5 * Tv51 + Tv51 + c_rv51) / (0.5 + c_rv51);
    RollIncl52 = (0.5 * Tv52 + Tv52 + c_rv52) / (0.5 + c_rv52);
    RollIncl53 = (0.5 * Tv53 + Tv53 + c_rv53) / (0.5 + c_rv53);
    RollIncl54 = (0.5 * Tv54 + Tv54 + c_rv54) / (0.5 + c_rv54);
    RollIncl55 = (0.5 * Tv55 + Tv55 + c_rv55) / (0.5 + c_rv55);
    RollIncl56 = (0.5 * Tv56 + Tv56 + c_rv56) / (0.5 + c_rv56);
    RollIncl57 = (0.5 * Tv57 + Tv57 + c_rv57) / (0.5 + c_rv57);
    RollIncl58 = (0.5 * Tv58 + Tv58 + c_rv58) / (0.5 + c_rv58);
    RollIncl59 = (0.5 * Tv59 + Tv59 + c_rv59) / (0.5 + c_rv59);
    RollIncl60 = (0.5 * Tv60 + Tv60 + c_rv60) / (0.5 + c_rv60);
    RollIncl61 = (0.5 * Tv61 + Tv61 + c_rv61) / (0.5 + c_rv61);
    RollIncl62 = (0.5 * Tv62 + Tv62 + c_rv62) / (0.5 + c_rv62);
    RollIncl63 = (0.5 * Tv63 + Tv63 + c_rv63) / (0.5 + c_rv63);
    RollIncl64 = (0.5 * Tv64 + Tv64 + c_rv64) / (0.5 + c_rv64);
    RollIncl65 = (0.5 * Tv65 + Tv65 + c_rv65) / (0.5 + c_rv65);
    RollIncl66 = (0.5 * Tv66 + Tv66 + c_rv66) / (0.5 + c_rv66);
    RollIncl67 = (0.5 * Tv67 + Tv67 + c_rv67) / (0.5 + c_rv67);
    RollIncl68 = (0.5 * Tv68 + Tv68 + c_rv68) / (0.5 + c_rv68);
    RollIncl69 = (0.5 * Tv69 + Tv69 + c_rv69) / (0.5 + c_rv69);
    RollIncl70 = (0.5 * Tv70 + Tv70 + c_rv70) / (0.5 + c_rv70);
    RollIncl71 = (0.5 * Tv71 + Tv71 + c_rv71) / (0.5 + c_rv71);
    RollIncl72 = (0.5 * Tv72 + Tv72 + c_rv72) / (0.5 + c_rv72);
    RollIncl73 = (0.5 * Tv73 + Tv73 + c_rv73) / (0.5 + c_rv73);
    RollIncl74 = (0.5 * Tv74 + Tv74 + c_rv74) / (0.5 + c_rv74);
    RollIncl75 = (0.5 * Tv75 + Tv75 + c_rv75) / (0.5 + c_rv75);
    RollIncl76 = (0.5 * Tv76 + Tv76 + c_rv76) / (0.5 + c_rv76);
    RollIncl77 = (0.5 * Tv77 + Tv77 + c_rv77) / (0.5 + c_rv77);
    RollIncl78 = (0.5 * Tv78 + Tv78 + c_rv78) / (0.5 + c_rv78);
    RollIncl79 = (0.5 * Tv79 + Tv79 + c_rv79) / (0.5 + c_rv79);
    RollIncl80 = (0.5 * Tv80 + Tv80 + c_rv80) / (0.5 + c_rv80);
    RollIncl81 = (0.5 * Tv81 + Tv81 + c_rv81) / (0.5 + c_rv81);
    RollIncl82 = (0.5 * Tv82 + Tv82 + c_rv82) / (0.5 + c_rv82);
    RollIncl83 = (0.5 * Tv83 + Tv83 + c_rv83) / (0.5 + c_rv83);
    RollIncl84 = (0.5 * Tv84 + Tv84 + c_rv84) / (0.5 + c_rv84);
    RollIncl85 = (0.5 * Tv85 + Tv85 + c_rv85) / (0.5 + c_rv85);
    RollIncl86 = (0.5 * Tv86 + Tv86 + c_rv86) / (0.5 + c_rv86);
    RollIncl87 = (0.5 * Tv87 + Tv87 + c_rv87) / (0.5 + c_rv87);
    RollIncl88 = (0.5 * Tv88 + Tv88 + c_rv88) / (0.5 + c_rv88);
    RollIncl89 = (0.5 * Tv89 + Tv89 + c_rv89) / (0.5 + c_rv89);
    RollIncl90 = (0.5 * Tv90 + Tv90 + c_rv90) / (0.5 + c_rv90);
    RollIncl91 = (0.5 * Tv91 + Tv91 + c_rv91) / (0.5 + c_rv91);
    RollIncl92 = (0.5 * Tv92 + Tv92 + c_rv92) / (0.5 + c_rv92);
    RollIncl93 = (0.5 * Tv93 + Tv93 + c_rv93) / (0.5 + c_rv93);
    RollIncl94 = (0.5 * Tv94 + Tv94 + c_rv94) / (0.5 + c_rv94);
    RollIncl95 = (0.5 * Tv95 + Tv95 + c_rv95) / (0.5 + c_rv95);
    RollIncl96 = (0.5 * Tv96 + Tv96 + c_rv96) / (0.5 + c_rv96);
    RollIncl97 = (0.5 * Tv97 + Tv97 + c_rv97) / (0.5 + c_rv97);
    RollIncl98 = (0.5 * Tv98 + Tv98 + c_rv98) / (0.5 + c_rv98);
    RollIncl99 = (0.5 * Tv99 + Tv99 + c_rv99) / (0.5 + c_rv99);
    RollIncl100 = (0.5 * Tv100 + Tv100 + c_rv100) / (0.5 + c_rv100);
}

```

CAPCINIT.F.H

```

/*
 * CAPC Controller Code: CAPC Initialization
 *
 * Last Update: 29 March 1995 (PV)
 */

```

```

#Include CAPC_Init
#Include EXT extern
#endif

```

```

/*
 * CONSTANTS
 *
 * STRUCTURES
 *
 * GLOBALS
 *
 * EXT double RollRate(), RollRateLoz()
 *
 * PROTOTYPES
 *
 * void Init_CAPC(void)
 */
#endif

```

KALMANF.CPP

```

/*
 * CAPC Controller Code: Kalman Filter
 *
 * Last Update: 20 June 1995 (PV)
 */

```

```

#include "Kalman.h"
#include "LMS.h"
#include "SuperVector.h"
#include "MethodSupport.h"
#include "math.h"

```

```

void KalmanFilter(void)
{
  double temp[nhfo_fari], inv[nhfi_fari],
         temp2[nhfo_fari], temp3[nhfo_fari], temp4[nhfo_fari],
         det[nhfo_fari], detInv[nhfo_fari],
         i, j, k, m;
  /*
   * XKF_fari[0] = lateral position
   * XKF_fari[1] = heading angle
   * XKF_fari[2] = half of curvature
   * V_Gar = lateral velocity
   * F_Gar = yaw rate
   */
}

```

```

Get_RInputF(i)
/*
 * Pr = Cov_XKF - Cov_XKF * CKF
 *
 * Inv(CKF + Cov_XKF) * CKF + Cov_XKF * CKF + Cov_XKF
 *
 * temp1 = Cov_XKF * CKF
 *
 * temp2 = CKF * temp1 + Cov_XKF
 *
 * temp3 = temp1 * Inv_temp2
 *
 * Pr = Cov_XKF - temp4 * Cov_XKF
 */

```

```

temp2[1] = Cov_XKF[1] * CKF[1]
temp2[2] = Inv_far * nhf_far * nhfo_far
temp2[3] = Inv_far * nhf_far * nhfo_far
temp2[4] = Inv_far * nhf_far * nhfo_far
for (i = 0; i < nhf_fari; i++)
  for (k = 0; k < nhfo_fari; k++)
  {
    Sum = 0.0;
    for (j = 0; j < nhf_fari; j++)
      for (l = 0; l < nhfo_fari; l++)
        temp2[1+l*k] = Sum + Cov_XKF[1+l*j] * CKF[1+l*k];
  }

```

```

Inverse(nhfo_far, temp2, Inv_temp2)
/*
 * temp2[1] = temp2[1] * Inv_temp2
 *
 * Inv_temp2[1] = Inv_far * nhf_far * nhfo_far
 *
 * temp3[1] = Inv_far * nhf_far * nhfo_far
 *
 * for (i = 0; i < nhf_fari; i++)
 *   for (k = 0; k < nhfo_fari; k++)
 *   {
 *     Sum = 0.0;
 *     for (j = 0; j < nhfo_fari; j++)
 *       for (l = 0; l < nhfo_fari; l++)
 *         temp3[1+l*k] = Sum + Cov_XKF[1+l*j] * CKF[1+l*k];
 *   }
 */

```

```

temp4[1] = temp3[1] * CKF
temp4[2] = Inv_far * nhf_far * nhfo_far * CKF
temp4[3] = Inv_far * nhf_far * nhfo_far * CKF
temp4[4] = Inv_far * nhf_far * nhfo_far * CKF
for (i = 0; i < nhf_fari; i++)
  for (k = 0; k < nhfo_fari; k++)
  {
    Sum = 0.0;
    for (j = 0; j < nhfo_fari; j++)
      for (l = 0; l < nhfo_fari; l++)
        temp4[1+l*k] = Sum + Cov_XKF[1+l*j] * CKF[1+l*k];
  }

```

```

Pr[1] = temp4[1] * Cov_XKF[1]
Pr[2] = Inv_far * nhf_far * nhfo_far * Cov_XKF[1]
Pr[3] = Inv_far * nhf_far * nhfo_far * Cov_XKF[1]
Pr[4] = Inv_far * nhf_far * nhfo_far * Cov_XKF[1]

```

```

for (k = 0; k < nht_farc; k++)
{
    for (l = 0; l < nht_farc; l++)
    {
        Sum = 0.0;
        for (j = 0; j < nht_farc; j++)
            Sum += temp1[j] * Cov_Xht(l|j) + Cov_Xht(j|l)k);
        Pr(l|k) = Cov_Xht(l|k) - Sum;
    }
}

/*----- Observation Update -----*/
for (k = 0; k < nhto_farc; k++)
{
    temp1[k] = Yht_farc[k];
    for (j = 0; j < nht_farc; j++)
        temp1[k] -= CR(l|j) * Xht_farc[j];
    for (l = 0; l < nht_farc; l++)
    {
        Xht_atarc[l] = Xht_farc[l];
        for (j = 0; j < nhto_farc; j++)
            Xht_atarc[l] += Xht_farc[j] * temp1[j];
    }
}

/*----- Time Update -----*/
for (k = 0; k < nht_farc; k++)
{
    Xht_farc[k] = 0.0;
    for (j = 0; j < nht_farc; j++)
        Xht_farc[k] += ARcd(l|j) * Xht_atarc[j];
    for (l = 0; l < nht_farc; l++)
        Xht_farc[k] += ARcd(l|k) * Uht(k);
}

/*-----*/
/*
Cov_Xht = ARcd * Pr + ARcd * Rvr
*/
/*-----*/
temp1[k] = Pr(l) * ARcd;
Pr(l) .. nht_farc x nht_farc
ARcd .. nht_farc x nht_farc
temp1 .. nht_farc x nht_farc
for (l = 0; l < nht_farc; l++)
{
    for (k = 0; k < nht_farc; k++)
    {
        Sum = 0.0;
        for (j = 0; j < nht_farc; j++)
            Sum += Pr(l|j) * ARcd(k|j);
        temp1[l|k] = Sum;
    }
}

/*
Cov_Xht(l) = ARcd * temp1
ARcd .. nht_farc x nht_farc
temp1 .. nht_farc x nht_farc
Cov_Xht(l) .. nht_farc x nht_farc
*/
for (l = 0; l < nht_farc; l++)
{
    for (k = 0; k < nht_farc; k++)
    {
        Sum = 0.0;
        for (j = 0; j < nht_farc; j++)
            Cov_Xht(l|k) = Sum + Arcd(j|k);
    }
}

/*----- New Coefficients for Right Ims -----*/
Rlanef_pol[0] = Xht_farc[0];
Rlanef_pol[1] = Xht_farc[1];
}

```

```

Rlanef_pol[0] = Xht_farc[2];
/*----- New Coefficients for Left Ims -----*/
for (k = 0; k < nhto_farc; k++)
{
    Datax[k] = X_p[k];
    derivat = 0.0;
    for (m = 1; m <= OrderF; m++)
        derivat += m * Rlanef_pol[OrderF-m] * Power(Datax[k], m-1);
    Datax[k] = 0.0;
    for (m = 0; m <= OrderF; m++)
        Datax[k] += Rlanef_pol[OrderF-m] * Power(Datax[k], m);
    if (mean.Solid == true)
    {
        Datax[k] += (Lanewidth - extra_offset) * sin(atan(derivat));
        Datax[k] -= (Lanewidth - extra_offset) * cos(atan(derivat));
    }
    else
    {
        Datax[k] += (2.0 * Lanewidth - extra_offset) * sin(atan(derivat));
        Datax[k] -= (2.0 * Lanewidth - extra_offset) * cos(atan(derivat));
    }
}

Lanewidths to the left
Lanewidths to the right
PolynomialFit(Datax, DataY, nhto_farc, OrderF, Rlanef_pol);
if (mean.Solid == false) /* Shift 1 Lanewidth to the right */
{
    for (k = 0; k < nhto_farc; k++)
    {
        Datax[k] = X_p[k];
        derivat = 0.0;
        for (m = 1; m <= OrderF; m++)
            derivat += m * Rlanef_pol[OrderF-m] * Power(Datax[k], m-1);
        Datax[k] = 0.0;
        for (m = 0; m <= OrderF; m++)
            Datax[k] += Rlanef_pol[OrderF-m] * Power(Datax[k], m);
        Datax[k] -= (Lanewidth - extra_offset) * sin(atan(derivat));
        Datax[k] += (Lanewidth - extra_offset) * cos(atan(derivat));
    }
}

PolynomialFit(Datax, DataY, nhto_farc, OrderF, Rlanef_pol);
return;
}

/*----- FUNCTION: EvalDeriv -----*/
double EvalDeriv(double x, int, int M, shift)
{
    double temp_xe, temp_ye, deriv, err_x;
    int iter, max_iter, m;
    if (M_shift == 0)
    {
        temp_ye = 0.0;
        for (m = 0; m <= OrderF; m++)
            temp_ye += Rlanef_pol[OrderF-m] * Power(x, m);
    }
    else
    {
        max_iter = 10;
        iter = 0;
        err_x = 1.0;
        while ((fabs(err_x) > 0.01) && (iter < max_iter))
    }
}

```



```

    deriv = 0.0;
    for (m = 1; m <= OrderF; m++)
        deriv += m * LlansF_pol[OrderF-m] * PowerI(temp_xe,m-1);

    temp_ye = 0.0;
    for (m = 0; m <= OrderF; m++)
        temp_ye += LlansF_pol[OrderF-m] * PowerI(temp_xe,m);

    temp_xe -= LaneWidth * sin(atan(deriv)) * M_shift;
    temp_ye += LaneWidth * cos(atan(deriv)) * M_shift;

    err_x = (x_int - temp_xe) / x_int;
    temp_xe += (x_int - temp_xe) * 1.1;
    iter++;
}
}
return(temp_ye);
}
/*----- FUNCTION: EvalRightF -----*/
double EvalRightF(double x_int, int M_shift)
{
    double temp_xe, temp_ye, deriv, err_x;
    int iter, max_iter, m;

    if (M_shift == 0)
    {
        temp_ye = 0.0;
        for (m = 0; m <= OrderF; m++)
            temp_ye += RlansF_pol[OrderF-m] * PowerI(x_int,m);
    }
    else
    {
        max_iter = 10;
        iter = 0;
        temp_xe = x_int;
        err_x = 1.0;
        while ((fabs(err_x) > 0.01) && (iter < max_iter))
        {
            deriv = 0.0;
            for (m = 1; m <= OrderF; m++)
                deriv += m * RlansF_pol[OrderF-m] * PowerI(temp_xe,m-1);

            temp_ye = 0.0;
            for (m = 0; m <= OrderF; m++)
                temp_ye += RlansF_pol[OrderF-m] * PowerI(temp_xe,m);

            temp_xe -= LaneWidth * sin(atan(deriv)) * M_shift;
            temp_ye += LaneWidth * cos(atan(deriv)) * M_shift;

            err_x = (x_int - temp_xe) / x_int;
            temp_xe += (x_int - temp_xe) * 1.1;
            iter++;
        }
    }
    return(temp_ye);
}
/*----- FUNCTION: EvalRightM -----*/
double EvalRightM(double x_int)
{
    float LocX[MaxScArrayM], LocY[MaxScArrayM];
    double temp_xe, temp_ye, deriv, err_x;
    int iter, max_iter, j, m;

    for (j = 0; j < nkfo_near; j++)
    {
        LocX[j] = X_pn[j];
        LocY[j] = Ykf_near[j];
    }

    /*----- Fit Linear Polynomial -----*/
    PolyFit(LocX, LocY, nkfo_near, OrderM, laneM_pol);

    if (meas.LanePosM == 1) /* Right LM data set is reference */
    {
        temp_ye = 0.0;
        for (m = 0; m <= OrderM; m++)
            temp_ye += laneM_pol[OrderM-m] * PowerI(x_int,m);
    }
    else /* Left LM data set is reference */
    {
        max_iter = 10;
        iter = 0;
        temp_xe = x_int;
        err_x = 1.0;
        while ((fabs(err_x) > 0.01) && (iter < max_iter))

```

```

    deriv = 0.0;
    for (m = 1; m <= OrderM; m++)
        deriv += m * laneM_pol[OrderM-m] * PowerI(temp_xe,m-1);

    temp_ye = 0.0;
    for (m = 0; m <= OrderM; m++)
        temp_ye += laneM_pol[OrderM-m] * PowerI(temp_xe,m);

    temp_xe -= LaneWidth * sin(atan(deriv));
    temp_ye += LaneWidth * cos(atan(deriv));

    err_x = (x_int - temp_xe) / x_int;
    temp_xe += (x_int - temp_xe) * 1.1;
    iter++;
}
}
return(temp_ye);
}
/*----- FUNCTION Get_KFInputF -----*/
void Get_KFInputF(void)
{
    float DataX[2*MaxScArrayF*MaxScArrayM], DataY[2*MaxScArrayF*MaxScArrayM];
    double L_xe[MaxScArrayF*MaxScArrayM], L_ye[MaxScArrayF*MaxScArrayM];
    double R_xe[MaxScArrayF*MaxScArrayM], R_ye[MaxScArrayF*MaxScArrayM];
    double distL, distR;
    int k, n_L, n_R;

    /*----- Combine Near-field and Far-field Data Points -----*/
    n_L = 0;
    while ((meas.LlaneM_xe[n_L] < meas.LlaneF_xe[0]) && (n_L < meas.n_leftM))
    {
        L_xe[n_L] = meas.LlaneM_xe[n_L];
        L_ye[n_L] = meas.LlaneM_ye[n_L];
        n_L++;
    }
    for (k = 0; k < meas.n_leftF; k++)
    {
        L_xe[n_L] = meas.LlaneF_xe[k];
        L_ye[n_L] = meas.LlaneF_ye[k];
        n_L++;
    }
    n_R = 0;
    while ((meas.RlaneM_xe[n_R] < meas.RlaneF_xe[0]) && (n_R < meas.n_rightM))
    {
        R_xe[n_R] = meas.RlaneM_xe[n_R];
        R_ye[n_R] = meas.RlaneM_ye[n_R];
        n_R++;
    }
    for (k = 0; k < meas.n_rightF; k++)
    {
        R_xe[n_R] = meas.RlaneF_xe[k];
        R_ye[n_R] = meas.RlaneF_ye[k];
        n_R++;
    }

    distL = L_xe[n_L - 1] - L_xe[0];
    distR = R_xe[n_R - 1] - R_xe[0];

    /*----- Fit Left and Right Lane Marker Arrays -----*/
    for (k = 0; k < n_L; k++)
    {
        DataX[k] = L_xe[k];
        DataY[k] = L_ye[k];
    }
    PolyShiftFit(DataX, DataY, n_L, OrderF, LlansF_pol);

    for (k = 0; k < n_R; k++)
    {
        DataX[k] = R_xe[k];
        DataY[k] = R_ye[k];
    }
    PolyShiftFit(DataX, DataY, n_R, OrderF, RlansF_pol);

    //PAUL
    #if 0
    /*----- Find LM points for Kalman Filter -----*/
    if (distL >= distR) /* More data from the left LM */
    {
        /*-----*/
        /* Take Ykf_far[1] & Ykf_far[4] from the left side */
        /* and Ykf_far[2] & Ykf_far[3] from the right side */
        /*-----*/
        Ykf_far[0] = EvalRightM(X_pf[0]);
        Ykf_far[1] = EvalLeftF(X_pf[1], 1);
    }

```



```

//PAUL (multiple lane operation disabled for now)
LanePosM = 1;
LanePos = 1;
return;
}

/*----- FUNCTION Get_KFinputM -----*/
void Get_KFinputM(void)
{
    int j, k, m;
    float LocX[MaxScArrayM], LocY[MaxScArrayM];
    double PolyCoef[2];

    if (meas.LanePosM == 1) /* Right roadedge is reference */
    {
        /*---- Fit data in case raw LM data is missing ----*/
        k = 0;
        m = 0;
        while ((meas.RlaneM_xe[k] < r_nearE) && (k < meas.n_rightM))
        {
            if (meas.RlaneM_xe[k] > r_nearB)
            {
                LocX[m] = meas.RlaneM_xe[k]; /* Scan up to r_nearE m ahead of vehicle */
                LocY[m] = meas.RlaneM_ye[k];
                m++;
            }
            k++;
        }
        PolyFit(LocX, LocY, m, 1, PolyCoef);
        /*---- Get Kalman Filter LMS input from RAW !! data (linear interpolation) ----*/
        k = 0;
        for (j = 0; j < nkfo_near; j++)
        {
            while ((meas.RlaneM_xe[k] < X_pn[j]) && (k < meas.n_rightM))
            {
                if (k == 0)
                    Ykf_near[j] = (PolyCoef[1] + PolyCoef[0] * X_pn[j]) * cos(atan(PolyCoef[0]));
                else
                {
                    if (k == meas.n_rightM)
                        Ykf_near[j] = (PolyCoef[1] + PolyCoef[0] * X_pn[j]) * cos(atan(PolyCoef[0]));
                    else
                        Ykf_near[j] = ((meas.RlaneM_xe[k] - X_pn[j]) * meas.RlaneM_ye[k-1] +
                                      (X_pn[j] - meas.RlaneM_xe[k-1]) * meas.RlaneM_ye[k]) /
                                      (meas.RlaneM_xe[k] - meas.RlaneM_xe[k-1]) *
                                      cos(HeadAngle_est);
                }
            }
        }
    }
    else /* Left roadedge is reference */
    {
        /*---- Fit data in case raw LM data is missing ----*/
        k = 0;
        m = 0;
        while ((meas.LlaneM_xe[k] < r_nearE) && (k < meas.n_leftM))
        {
            if (meas.LlaneM_xe[k] > r_nearB)
            {
                LocX[m] = meas.LlaneM_xe[k]; /* Scan up to r_nearE m ahead of vehicle */
                LocY[m] = meas.LlaneM_ye[k];
                m++;
            }
            k++;
        }
        PolyFit(LocX, LocY, m, 1, PolyCoef);
        /*---- Get Kalman Filter LMS input from RAW !! data (linear interpolation) ----*/
        k = 0;
        for (j = 0; j < nkfo_near; j++)
        {
            while ((meas.LlaneM_xe[k] < X_pn[j]) && (k < meas.n_leftM))
            {
                if (k == 0)
                    Ykf_near[j] = (PolyCoef[1] + PolyCoef[0] * X_pn[j]) * cos(atan(PolyCoef[0]));
                else
                {
                    if (k == meas.n_leftM)
                        Ykf_near[j] = (PolyCoef[1] + PolyCoef[0] * X_pn[j]) * cos(atan(PolyCoef[0]));
                    else
                        Ykf_near[j] = ((meas.LlaneM_xe[k] - X_pn[j]) * meas.LlaneM_ye[k-1] +
                                      (X_pn[j] - meas.LlaneM_xe[k-1]) * meas.LlaneM_ye[k]) /
                                      (meas.LlaneM_xe[k] - meas.LlaneM_xe[k-1]) *
                                      cos(HeadAngle_est);
                }
            }
        }
    }
}

```

```

}
return;
}

/*----- FUNCTION Init_stateKalmanM -----*/
void Init_stateKalmanM(void)
{
    int i, k;
    double PolyCoef[2];
    float LocX[MaxScArrayM], LocY[MaxScArrayM];

    /*---- Assumption: vehicle is standing still on a straight piece of road ----*/
    k = 0;
    if (meas.LanePosM == 1) /* Right roadedge is reference */
    {
        while ((meas.RlaneM_xe[k] < r_nearE) && (k < meas.n_rightM))
        {
            LocX[k] = meas.RlaneM_xe[k]; /* Take data up to r_nearE m ahead of vehicle */
            LocY[k] = meas.RlaneM_ye[k];
            k++;
        }
    }
    else
    {
        while ((meas.LlaneM_xe[k] < r_nearE) && (k < meas.n_leftM))
        {
            LocX[k] = meas.LlaneM_xe[k]; /* Take data up to r_nearE m ahead of vehicle */
            LocY[k] = meas.LlaneM_ye[k];
            k++;
        }
    }
    PolyFit(LocX, LocY, k, 1, PolyCoef);
    for (i = 0; i < nkf_near; i++) /* Initial condition for Kalman Filter */
    {
        Xkf_near[i] = 0.0;
        dXkf_near[i] = 0.0;
    }
    Xkf_near[3] = -atan(PolyCoef[0]); /* initial heading angle */
    Xkf_near[2] = PolyCoef[1] * cos(Xkf_near[3]); /* initial lateral deviation */
    for (i = 0; i < nkfo_near; i++) /* Initialize LM input for Kalman Filter */
        Ykf_near[i] = 0.0;
    return;
}

/*----- FUNCTION Get_KFgainsM -----*/
void Get_KFgainsM(void)
{
    int i, j;

    /*---- Calculate the Kalman Filter feedback gains as a function of vehicle speed ----*/
    for (i = 0; i < nkf_near-2; i++)
        for (j = 0; j < nkfo_near; j++)
            Lkf_near[i][j] = Pkf_near[i][j][3] +
                            Pkf_near[i][j][2] * meas.u_car +
                            Pkf_near[i][j][1] * meas.u_car * meas.u_car +
                            Pkf_near[i][j][0] * meas.u_car * meas.u_car * meas.u_car;

    return;
}

/*----- FUNCTION Init_KalmanM -----*/
void Init_KalmanM(void)
{
    int i, j;

    Init_stateKalmanM(); /* Find initial conditions for Kalman Filter */
    /*---- At these longitudinal distances a LM value is expected ----*/
    X_pn[0] = 5.0; /* distance from c.g. of vehicle in [m] */
    X_pn[1] = 7.5;
    X_pn[2] = 10.0;
    X_pn[3] = 12.5;
    X_pn[4] = 15.0;

    /* These are the Polynomial Coefficients of the Kalman Filter feedback gains */
    /* as a function of the vehicle speed (m/s) (3rd-order fit) */
    Pkf_near[0][0][0] = -8.8658e-07;
}

```

```

PKT_near[0][0][1] = -5.4881e-05;
PKT_near[0][0][2] = -9.3365e-04;
PKT_near[0][0][3] = 2.9222e-03;
PKT_near[0][1][0] = -5.5512e-07;
PKT_near[0][1][1] = 4.5124e-05;
PKT_near[0][1][2] = 2.6095e-03;
PKT_near[0][1][3] = -3.7704e-07;
PKT_near[0][2][0] = -5.1594e-04;
PKT_near[0][2][1] = 1.4018e-02;
PKT_near[0][2][2] = -3.9824e-04;
PKT_near[0][2][3] = 2.7272e-05;
PKT_near[0][3][0] = -3.9884e-04;
PKT_near[0][3][1] = -1.7424e-07;
PKT_near[0][3][2] = 2.9622e-07;
PKT_near[0][3][3] = 5.2932e-04;
PKT_near[1][0][0] = -2.3368e-08;
PKT_near[1][0][1] = -1.9732e-04;
PKT_near[1][0][2] = 1.3184e-04;
PKT_near[1][0][3] = 2.3343e-06;
PKT_near[1][1][0] = -1.5345e-04;
PKT_near[1][1][1] = 6.2529e-04;
PKT_near[1][1][2] = -7.5988e-09;
PKT_near[1][1][3] = -1.3380e-04;
PKT_near[1][2][0] = -3.3338e-04;
PKT_near[1][2][1] = -3.8932e-09;
PKT_near[1][2][2] = 1.3011e-06;
PKT_near[1][2][3] = -1.1818e-04;
PKT_near[1][3][0] = -1.1078e-09;
PKT_near[1][3][1] = 9.8024e-07;
PKT_near[1][3][2] = -1.0872e-04;
PKT_near[1][3][3] = 4.1174e-04;
PKT_near[2][0][0] = -1.2054e-05;
PKT_near[2][0][1] = 1.8968e-02;
PKT_near[2][0][2] = 7.6344e-03;
PKT_near[2][0][3] = -3.2480e-07;
PKT_near[2][1][0] = 3.1407e-05;
PKT_near[2][1][1] = -1.8092e-02;
PKT_near[2][1][2] = 4.6823e-07;
PKT_near[2][1][3] = -4.4573e-05;
PKT_near[2][2][0] = 1.0925e-02;
PKT_near[2][2][1] = -5.5442e-05;
PKT_near[2][2][2] = 5.7837e-07;
PKT_near[2][2][3] = 9.3133e-03;
PKT_near[2][3][0] = -3.3429e-02;
PKT_near[2][3][1] = 6.5112e-05;
PKT_near[2][3][2] = 8.2373e-03;
PKT_near[2][3][3] = -4.0141e-02;
PKT_near[3][0][0] = -1.1398e-07;
PKT_near[3][0][1] = 2.1407e-05;
PKT_near[3][0][2] = -2.1407e-05;
PKT_near[3][0][3] = -1.0070e-03;
PKT_near[3][1][0] = -1.6118e-07;
PKT_near[3][1][1] = 2.6744e-05;
PKT_near[3][1][2] = -2.3222e-03;
PKT_near[3][1][3] = -2.4874e-05;
PKT_near[3][2][0] = -2.4874e-05;
PKT_near[3][2][1] = 3.0961e-05;
PKT_near[3][2][2] = -2.3670e-03;
PKT_near[3][2][3] = -3.6530e-03;
PKT_near[3][3][0] = -2.2672e-07;
PKT_near[3][3][1] = 2.4304e-03;
PKT_near[3][3][2] = -4.6732e-03;
PKT_near[3][3][3] = -2.5468e-07;
PKT_near[4][0][0] = 3.8285e-05;
PKT_near[4][0][1] = 3.8285e-05;
PKT_near[4][0][2] = 2.5526e-03;
PKT_near[4][0][3] = 1.4501e-07;
PKT_near[4][1][0] = -5.3682e-07;
PKT_near[4][1][1] = 6.6249e-05;
PKT_near[4][1][2] = 1.5377e-03;
PKT_near[4][1][3] = 2.7895e-09;
PKT_near[4][2][0] = 4.3375e-05;
PKT_near[4][2][1] = 2.7511e-03;
PKT_near[4][2][2] = 2.1385e-09;
PKT_near[4][2][3] = -3.3545e-07;
PKT_near[4][3][0] = 2.0249e-05;
PKT_near[4][3][1] = 7.0722e-10;
PKT_near[4][3][2] = -6.0725e-08;
PKT_near[4][3][3] = -6.0725e-08;

```

```

PKT_near[4][3][2] = -2.8493e-06;
PKT_near[4][3][3] = 4.8069e-03;
PKT_near[4][4][0] = -1.1179e-09;
PKT_near[4][4][1] = 2.5369e-07;
PKT_near[4][4][2] = -7.7292e-03;
PKT_near[4][4][3] = 5.7729e-03;

for (i = 0; i < nkt_near; i++)
    for (j = 0; j < nhto_near; j++)
        LKT_near[i][j] = 0.0;

/* Initialize Kalman Filter Feedback Matrix */
/* Compute Kalman Filter Feedback Matrix */
Get KFgainHT();
OidVolo = mess.u_cari;
get KFInputHT();
return;
}

```

KALMAN.H

```

/*
 * CAPC Controller Code: Kalman Filter
 * Last Update: 29 March 1995 (PV)
 */

/*
 * Order of Kalman filter for near-field IMS
 * Number of Kalman filter inputs for near-field IMS
 */
#define nkt_near 7
#define nhto_near 5
#define nkto_near 5

/*
 * STRUCTURES
 */
/*
 * GLOBALS
 */

EXT double LKT_near[nkt_near][nkto_near];
EXT double Xp[nkto_near][nkto_near];
EXT double Xp[nkto_near][nkto_near];
EXT double Xp[nkto_near][nkto_near];

/*
 * PROTOTYPES
 */

void KP_equatLonat(double dkt[], double XKF[]);
void KalmanFilter(void);
void KalmanFilter(void);
void KalmanFilter(void);
void Init_Kalman(void);
void Init_stateKalman(void);
void Get_KFgainHT(void);

#endif

```

IMS.C

```

/*
 * CAPC Controller Code: Lane Marker Sensor
 * Last Update: 20 June 1995 (PV)
 */

```

```

#include "hardWare.h" // hardware global
#include "globals.h" // LMS serial defn
#include "MathSupport.h"
#include "Kalmann.h"
#include "Supervisor.h"
#include "math.h"

/----- FUNCTION: Init_LMS -----*/
void Init_LMS(void)
{
    int i;

    for (i = 0; i < MaxSArrayF; i++)
    {
        mean_LlaneF_xe[i] = 0.0;
        mean_LlaneF_ye[i] = 0.0;
        mean_RlaneF_xe[i] = 0.0;
        mean_RlaneF_ye[i] = 0.0;
    }

    for (i = 0; i < MaxSArrayM; i++)
    {
        mean_LlaneM_xe[i] = 0.0;
        mean_LlaneM_ye[i] = 0.0;
        mean_RlaneM_xe[i] = 0.0;
        mean_RlaneM_ye[i] = 0.0;
    }

    for (i = 0; i < OrderF; i++)
        laneF_pol[i] = 0.0;

    for (i = 0; i < OrderF; i++)
        laneM_pol[i] = 0.0;

    laneF_solIDL = true;
    laneM_solIDL = true;
    Order_SolIDR = false;
    LMSready = false;
}

/----- FUNCTION: Fit_fitdata -----*/
void Fit_fitdata(void)
{
    float k, n_L, n_R;

    /----- Combine Near-field and Far-field Data Points -----*/
    n_L = 0;
    while ((mean_LlaneF_xe[n_L] < mean_LlaneF_xe[0]) || (n_L < mean_n_leftF))
    {
        DataX[n_L] = mean_LlaneF_xe[n_L];
        DataY[n_L] = mean_LlaneF_ye[n_L];
        n_L++;
    }

    for (k = 0; k < mean_n_leftF; k++)
    {
        DataX[n_L] = mean_LlaneF_xe[k];
        DataY[n_L] = mean_LlaneF_ye[k];
        n_L++;
    }

    /----- Fit Left Lane Marker Arrays -----*/
    PolyShifFit(DataX, DataY, n_L, OrderF, laneF_pol);

    /----- Combine Near-field and Far-field Data Points -----*/
    n_R = 0;
    while ((mean_RlaneF_xe[n_R] < mean_RlaneF_xe[0]) || (n_R < mean_n_rightF))
    {
        DataX[n_R] = mean_RlaneF_xe[n_R];
        DataY[n_R] = mean_RlaneF_ye[n_R];
        n_R++;
    }

    for (k = 0; k < mean_n_rightF; k++)
    {
        DataX[n_R] = mean_RlaneF_xe[k];
        DataY[n_R] = mean_RlaneF_ye[k];
        n_R++;
    }
}

```

```

/----- File Right Lane Marker Arrays -----*/
PolyShifFit(DataX, DataY, n_R, OrderF, laneR_pol);
return;

/----- FUNCTION: FitLaneGeof -----*/
void FitLaneGeof(void)
{
    float DataX[nkfo_far], DataY[nkfo_far], DataX[nkfo_far], DataY[nkfo_far];
    int i, j, k, m;

    get_RFInputF();

    for (j = 0; j < nkfo_far; j++)
    {
        DataX[j] = X_pf(j);
        DataY[j] = YH_far(j);
    }

    /----- Fit Linear Polynomial -----*/
    PolyFit(DataX, DataY, nkfo_far, OrderF, laneF_pol);

    for (k = 0; k < nkfo_far; k++)
    {
        DataX[k] = X_pf(k);
        DataY[k] = 0.0;
        for (m = 0; m < OrderF; m++)
            DataY[k] += laneF_pol[OrderF-m] * Power(X_pf(k), m);
        derivat = 0.0;
        for (m = 1; m < OrderF; m++)
            derivat += m * laneF_pol[OrderF-m] * Power(X_pf(k), m-1);
        if (mean_SolIDL)
        {
            DataX[k] = DataX[k] + (laneWidth - extra_offset) * sin(tan(derivat));
            DataY[k] = DataY[k] - (laneWidth - extra_offset) * cos(tan(derivat));
        }
        else
        {
            DataX[k] = DataX[k] + (2.0 * laneWidth - extra_offset) * sin(tan(derivat));
            DataY[k] = DataY[k] - (2.0 * laneWidth - extra_offset) * cos(tan(derivat));
        }
    }

    /----- OVERVERSION -----*/
    float DataX[2*MaxSArrayF], DataY[2*MaxSArrayF], DataX[2*MaxSArrayF];
    double L_xe[MaxSArrayF*MaxSArrayM], L_ye[MaxSArrayF*MaxSArrayM];
    double R_xe[MaxSArrayF*MaxSArrayM], R_ye[MaxSArrayF*MaxSArrayM];
    int k, n_L, n_R, temp_no, temp_yes;

    /----- Combine Near-field and Far-field Data Points -----*/
    n_L = 0;
    while ((mean_LlaneF_xe[n_L] < mean_LlaneF_xe[0]) || (n_L < mean_n_leftF))
    {
        L_xe[n_L] = mean_LlaneF_xe[n_L];
        L_ye[n_L] = mean_LlaneF_ye[n_L];
        n_L++;
    }

    for (k = 0; k < mean_n_leftF; k++)
    {
        L_xe[n_L] = mean_LlaneF_xe[k];
        L_ye[n_L] = mean_LlaneF_ye[k];
        n_L++;
    }

    n_R = 0;
    while ((mean_RlaneF_xe[n_R] < mean_RlaneF_xe[0]) || (n_R < mean_n_rightF))

```

```

R_xe[n_R] = mass_RlaneR_xe[n_R]
R_ye[n_R] = mass_RlaneR_ye[n_R]
for (k = 0; k < mass_n_right; k++)
{
  R_xe[n_R] = mass_RlaneR_xe[k];
  R_ye[n_R] = mass_RlaneR_ye[k];
}
}
distL = L_xe[n_L - 1] - L_xe[0];
distR = R_xe[n_R - 1] - R_xe[0];
/----- Fit Left and Right Lane Marker Arrays -----*/
for (k = 0; k < n_L; k++)
{
  DataX[k] = L_xe[k];
  DataY[k] = L_ye[k];
}
PolynomialFit(DataX, DataY, n_L, OrderF, LaneF_pol);
for (k = 0; k < n_R; k++)
{
  DataX[k] = R_xe[k];
  DataY[k] = R_ye[k];
}
PolynomialFit(DataX, DataY, n_R, OrderF, LaneF_pol);
/----- Combine Left and Right Lane Markers -----*/
j = 0;
n = 0;
for (k = 0; k < n_L; k++)
{
  derivat = 0.0;
  for (m = 1; m <= OrderF; m++)
    derivat += m * LaneF_pol[OrderF-m] * Power(L_xe[k],m-1);
  L_ye[k] = 0.0;
  for (m = 0; m <= OrderF; m++)
    L_ye[k] += LaneF_pol[OrderF-m] * Power(L_xe[k],m);
  temp_xe = L_xe[k] - LaneWidth * sin(atan(derivat));
  temp_ye = L_ye[k] + LaneWidth * cos(atan(derivat));
  if (temp_xe < R_xe[j])
  {
    DataX[n] = temp_xe;
    DataY[n] = temp_ye;
    n++;
  }
  else
  {
    while ((temp_xe > R_xe[j]) && (j < n_R))
    {
      DataX[j] = R_xe[j];
      DataY[j] = R_ye[j];
      j++;
    }
    DataX[n] = temp_xe;
    DataY[n] = temp_ye;
    n++;
  }
}
/----- Fit New Right Lane Marker Array with Polynomial Function -----*/
for (k = 0; k < n; k++)
  /* original DataX will be destroyed */
  cDataX[k] = DataX[k];
PolynomialFit(DataX, DataY, n, OrderF, LaneF_pol);
for (k = 0; k < n; k++)
  /* refill DataX array with copied data */
  DataX[k] = cDataX[k];
/----- Determine New Right Lane Marker Array -----*/
if (mass_SolidR == false)
  for (k = 0; k < (n_L + n_R); k++)
  {
    derivat = 0.0;
    for (m = 1; m <= OrderF; m++)
      derivat += m * RlaneF_pol[OrderF-m] * Power(DataX[k],m-1);
    DataX[k] = 0.0;
    for (m = 0; m <= OrderF; m++)
      DataY[k] += RlaneF_pol[OrderF-m] * Power(DataX[k],m);
    DataX[k] -= LaneWidth * sin(atan(derivat));
  }
}
DataY[k] += LaneWidth * cos(atan(derivat));
}
/----- Fit New Right Lane Marker Array with Polynomial Function -----*/
PolynomialFit(DataX, DataY, n, OrderF, RlaneF_pol);
for (k = 0; k < n; k++)
  /* refill DataX array with copied data */
  DataX[k] = cDataX[k];
}
/----- Determine New Left Lane Marker Array -----*/
for (k = 0; k < n; k++)
{
  derivat = 0.0;
  for (m = 1; m <= OrderF; m++)
    derivat += m * LlaneF_pol[OrderF-m] * Power(DataX[k],m-1);
  DataY[k] = 0.0;
  for (m = 0; m <= OrderF; m++)
    DataY[k] += LlaneF_pol[OrderF-m] * Power(DataX[k],m);
  if (mass_SolidL == true)
  {
    DataX[k] += LaneWidth * sin(atan(derivat));
    DataY[k] -= LaneWidth * cos(atan(derivat));
  }
  else
  {
    DataX[k] += 2.0 * LaneWidth * sin(atan(derivat));
    DataY[k] -= 2.0 * LaneWidth * cos(atan(derivat));
  }
  /* Shift 2 laneWidths to the left */
  DataX[k] += 2.0 * LaneWidth * sin(atan(derivat));
  DataY[k] -= 2.0 * LaneWidth * cos(atan(derivat));
}
}
}
/----- Fit New Left Lane Marker Array with Polynomial Function -----*/
PolynomialFit(DataX, DataY, n, OrderF, LlaneF_pol);
}
#endif
/----- FUNCTION: fitlanegeom -----*/
void fitlanegeom(void)
{
  float   Lock[MaxScArrayM], Lock[MaxScArrayM];
  double  derivat;
  int     j;
  get_RInputer();
  for (j = 0; j < nkfo_near; j++)
  {
    Lock[j] = k_pn[j];
    Lock[j] = k_t_pn[j];
  }
}
/----- Fit Linear Polynomial -----*/
PolynomialFit(Lock, Lock, nkfo_near, OrderM, laneF_pol);
/----- These are Heading Angle and Lateral Deviation in the Vehicle C.G. -----*/
derivat = 0.0;
for (j = 1; j <= OrderM; j++)
  derivat += j * laneF_pol[OrderM-j] * Power((0.0,j)-1);
HeadingAngle_est = -tan(derivat);
oldLatDev_est = latDev_est;
latDev_est = laneF_pol[OrderM] * cos(HeadingAngle_est);
return;
}

```



```

//-----
// GLOBALS
//
//
EXT double LaneF_pol[Order+1], LaneF_pol[Order+1];
EXT double LaneF_pol[Order+1];
EXT double HeadAngle_est, LatDev_est, oldLatDev_est;
EXT Boolean OnShoulder, ImSteady;

EXT Int LaneFom, LaneFom;

//-----
// PROTOTYPES
//
//
void Init_LatF(void);
void FitLaneDev(void);
void Fit_FitData(void);
void GetFitData(void);
#endif

```

MATHSUPPORT.CPP

```

// CAPC Controller Code: Math Support
//
// Last Update: 8 February 1995 (RV)
//
#include "MathSupport.h"
#include "Math.h"
//-----
//----- FUNCTION: Inverse
void Inverse(int Dimen, double Data[MaxMat], double Inv[MaxMat][MaxMat])
{
    double Mult, Divisor, DummyFom[MaxMat];
    Int Error, Row, Row, Row, Row, i, j;

    Error = 0;
    for (i = 0; i < Dimen; i++)
        for (j = 0; j < Dimen; j++)
            Inv[i][j] = 0.0;
            Inv[i][i] = 1.0;
    }

    while ((Error == 0) && (Row < Dimen))
    {
        if (fabs(Data[Row][Row]) < REALTY_ZERO)
            Error = 1;
        while ((Error > 0) && (Row < Dimen))
        {
            Row = Row + 1;
            if (fabs(Data[Row][Row]) > REALTY_ZERO)
            {
                for (i = 0; i < Dimen; i++)
                {
                    DummyFom[i] = Data[Row][i];
                    Data[Row][i] = Data[Row][i];
                    Data[Row][i] = DummyFom[i];
                    Data[Row][i] = Inv[Row][i];
                    Data[Row][i] = Inv[Row][i];
                    Inv[Row][i] = DummyFom[i];
                }
                Error = 0;
            }
        }
    }

    if (Error == 0)
        Divisor = Data[Row][Row];
    for (i = 0; i < Dimen; i++)
        Data[Row][i] /= Divisor;
        Inv[Row][i] /= Divisor;
    }
    for (Row = 0; Row < Dimen; Row++)
        if ((Row != Row) && (fabs(Data[Row][Row]) > REALTY_ZERO))

```

```

{
    Mult = - Data[Row][Row] / Data[Row][Row];
    for (i = 0; i < Dimen; i++)
    {
        Data[Row][i] += Mult * Data[Row][i];
        Inv[Row][i] += Mult * Inv[Row][i];
    }
    Row = Row + 1;
}
return;
}

//----- FUNCTION: PolyFit
void PolyFit(float Data[], float Data[], int Mpoint, int Order, double Poly[])
{
    double Sum, Sum;
    Int i, j, k;

    for (i = 0; i < Mpoint; i++)
        for (j = 0; j < Order; j++)
            Q[i][j] = Power(Data[i], (Order - j));

    for (i = 0; i < Order; i++)
    {
        for (k = 0; k < Order; k++)
        {
            Sum = 0.0;
            for (j = 0; j < Mpoint; j++)
                Q[i][k] = Sum;
        }
    }

    Inverse(Order+1, Q, Inv_Q);
    for (i = 0; i < Order; i++)
    {
        Sum2 = 0.0;
        for (k = 0; k < Mpoint; k++)
        {
            Sum = 0.0;
            for (j = 0; j < Order; j++)
                Sum += Sum * Inv_Q[i][j] * Q[k][j];
            Sum2 += Sum * Data[k];
        }
        Poly[i] = Sum2;
    }
    return;
}

//----- FUNCTION: PolyShiftFit
void PolyShiftFit(float Data[], float Data[], int Mpoint, int Order, double Poly[])
{
    // CAUTION: Maximum Order of Polynomial Fit = 3
    //
    Int i, j, ShiftX, ShiftY;
    float ShiftX, ShiftY;

    //----- Subtract begin point to avoid ill-conditioned matrices -----
    ShiftX = Data[0];
    ShiftY = Data[0];
    for (k = 0; k < Mpoint; k++)
    {
        Data[k] -= ShiftX;
        Data[k] -= ShiftY;
    }

    PolyFit(Data, Data, Mpoint, Order, Poly); // Fit polynomial function
    switch(Order)
    {
        case (1):
            Poly[0] += - 1.0 * Poly[0] * ShiftX + ShiftY;
            break;
        case (2):
            Poly[2] += - 1.0 * Poly[1] * ShiftX + 1.0 * Poly[0] * ShiftX * ShiftX + ShiftY;
            Poly[1] += - 2.0 * Poly[0] * ShiftX;
            break;
        case (3):
            Poly[3] += - 1.0 * Poly[2] * ShiftX + 1.0 * Poly[1] * ShiftX * ShiftX
                - 1.0 * Poly[0] * ShiftX * ShiftX * ShiftX + ShiftY;
    }
    // Correct coefficients for shifts
}

```



```

X_pth[1] = DT_PTM * (Fy_1 * a_v - Fy_2 * b_v) / Lsv;
X_pth[2] = DT_PTM * X_pth[1];
X_pth[3] = DT_PTM * (mass * cos(X_pth[2]) - X_pth[0] * sin(X_pth[2]));
X_pth[4] = DT_PTM * (mass * cos(X_pth[2]) + X_pth[0] * cos(X_pth[2]));
return;
}

/*----- FUNCTION: PathCurvature -----*/
double PathCurvature(void)
{
float
double
double
int
int
for (i = 0; i < 10; i++)
{
float h = X_ndl[n_prd-1-i];
float l = Y_ndl[n_prd-1-i];
}
PolyhFitf(Loadf, Loadf, 10, 2, PolyCoef);
Curvat = 2.0 * PolyCoef[0];
// Curvat = X_pth[1] / mass * v_cars;
return(Curvat);
}

```

PathPrediction.h

```

/* CAB Controller Codes Path Prediction */
/* Last Update: 29 March 1993 (PV) */
/*-----*/
#ifndef CAB_PATH
#define CAB_PATH
#endif
/*-----*/
/* order of vehicle model */
#define ORDER_VEH 5
/* maximum number of path prediction intervals */
#define MAX_PTM 200
/* integration time step */
#define DT_PTM 5.000E-2
/* maximum path prediction time [sec] */
#define MAX_PTM 4.000E+0
/*-----*/
/* STRUCTURES */
/*-----*/
/* GLOBALS */
/*-----*/
/* PROTOTYPES */
/*-----*/
void Init_Path_Prediction(void);
double Path_Prediction(double X_pth[]);
#endif

```

STRINGUTILS.C

```

#include <string.h>
#include <ctype.h>
#include <string.h>
#include <stringutil.h>
StringPtr GetSubstr(const String (const char*, const char*, const char*)
StringKeenIndl i, n;
SHORT StringPtr pi
if (stringIndex > 0) {
h = (StringKeenIndl) GetSource (readOnly, read);
if (h == NULL) {
return NULL;
}
}
return pi;
}
return NULL;
}
Boolean PTRTOULong (str235 str, long num)
{
// Return TRUE if it's a valid number, otherwise FALSE
unsigned char l, c, *pi;
unsigned char len = strlen(str);
unsigned char maxl = '\0299999999';
// Check for too-long numbers
if (len > 10)
return FALSE;
// Check for max-length, overly large numbers
if (len > 0)
{
for (i = 1; i <= len; i++)
{
if (c = str[i] < '0' || c > '9')
return FALSE;
}
return TRUE;
}
}
void SmartCopyStr (char *p1, char *p2)
{
if (p1 == NULL)
return;
if (p2 == NULL)
return;
// If the destination pointer is NULL,
// just copy the source pointer
// Otherwise, if the two pointers
aren't equal CopyStr (p1, p2) go ahead and call copystr
}
void CopyStr (str235 t, str235 s)
{
int i;
for (i = 1; i <= strlen(s); i++)
t[i] = s[i];
t[i] = '\0';
}
void AppendStr (str235 t, str235 s)
{
short i, j;
} = (short)t[i];
for (i = (short)strlen(t); i > 0; i--)
t[i+1] = t[i];
t[0] = (char)(t + (short)strlen(t));
}

```

```

)
short CompareStrings (StringPtr str1, StringPtr str2)
{
short len1, len2, i, ch;
short minLen, lenDiff;
unsigned char *p1, *p2;

len1 = strlen(0);
len2 = strlen(0);
lenDiff = len1 - len2;
minLen = (lenDiff < 0 ? len1 : len2);
for (i = minLen; p1 = str1, p2 = str2; i++)
if (ch = *p1++ != *p2++)
return lenDiff;

return lenDiff;
}

void ProcStrCopy(char *t, Str255 s)
{
int i;
for(i=1; i<=a[0];i++)
t[i-1] = a[i];
t[i-1] = '\0';
}

void CrossCopy(Str255 t, char *s)
{
int i;
int m;
char *p;

m = 0;
p = s;
while (*p++)
m = strlen(p);
for(i=0; i<=m;i++)
t[i] = s[i];
}

unsigned short ReverseCopy2CStr (unsigned char *pas, char *o)
{
short i, j, len = *pas++;
for (i = len, a = j; i > 0; i--)
a = *pas++;
a = '\0';
return len;
}

void ReverseCStr (char *str)
{
short i, j, n = 0;
char t, *p1, *p2;

while (*str)
p1 = p2 = str;
while (*p2++)
p2--;
for (i = n; i > 0; i--) {
t = *--p2;
*p2 = *--p1;
*p1 = t;
}
}

Boolean GetResString(int strID, int pos, Str255 pStr)
{
Handle hHandle;
Handle hStrTable;
hStrTable = GetResource('STR', strID);
if (hStrTable == nil)
return false;

// lock dereferenced handle
Lock(hStrTable);

GetIndString(pStr, strID, pos);
Unlock(hStrTable);
ReleaseResource(hStrTable); // free up memory when ALL done
return true;
}

Boolean GetResString(int strID, int pos, char *cStr)
{
Handle hHandle;
Handle hStrTable;
Str255 pStr;
hStrTable = GetResource('STR', strID);
if (hStrTable == nil)
return false;

// lock dereferenced handle
Lock(hStrTable);

GetIndString(pStr, strID, pos);
ProcStrCopy(cStr, pStr);
ReleaseResource(hStrTable); // free up memory when ALL done
return true;
}

void StrToLower(char *s)
{
char *p;
p = s;
while(*p)
if (islower(*p))
*p++ = *p-32;
}

void StrToUpper(char *s)
{
char *p;
p = s;
while(*p)
if (isupper(*p))
*p++ = *p+32;
}

void TrimTrailingWhiteSpace(char *str)
{
int i;
i = strlen(str) - 1;
while (str[i] == ' ')
i--;
str[i+1] = '\0';
}

void TrimLeadingSpace(char *s)
{
char *p;
p = s;
while (isspace(*p))
p++;
strcpy(s, p);
}

/*
-----
InchString.
-----
Scan the string set for the token.
Enters: tok - token to search for.
set - character set to scan.
-----
Exit:
return - true if in set, else false.
-----
Boolean InchString(char tok, char *set)
{
int i;
setLen;
Boolean InSet;

setLen = strlen(set);
InSet = false;
for (i=0; i<setLen; i++)
if (tok == set[i])
}
}

```



```

setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// ----- second good
// C = CAPCI.d, L = CAPCI.d, L = CAPCI.d
XYMin: ticsU[zz] = 2.0f // y-grid spacing
XYMin: absU[zz] = -4.0f // min graph value
XYMin: absU[zz] = 4.0f // max graph value
XYMin: absU[zz] = 1 // max graph value
XYMin: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
XYMin: ticsU[zz] = 2.0f // y-grid spacing
XYMin: absU[zz] = -4.0f // min graph value
XYMin: absU[zz] = 4.0f // max graph value
XYMin: absU[zz] = 1 // max graph value
XYMin: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// ----- PILOTS 6 -----
// C = last good
// C = last good
XYMin: ticsU[zz] = 2.0f // y-grid spacing
XYMin: absU[zz] = -4.0f // min graph value
XYMin: absU[zz] = 4.0f // max graph value
XYMin: absU[zz] = 1 // max graph value
XYMin: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
XYMin: ticsU[zz] = 2.0f // y-grid spacing
XYMin: absU[zz] = -4.0f // min graph value
XYMin: absU[zz] = 4.0f // max graph value
XYMin: absU[zz] = 1 // max graph value
XYMin: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

```

BGMInSetup.c

```

// this is the Bar-Graph window setup include.
// a rather poor rendition of an edit-able setup.c

define
setInternalTextVars
{ BGMIn:shortName[zz] = (Pfr){(a+)}->shortName }
{ BGMIn:unitName[zz] = (Pfr){(a+)}->unitName }
{ BGMIn:dFormat [zz] = (Pfr){(a+)}->dFormat }
{ BGMIn:chardata[zz] = (Pfr){(a+)}->chardata }
}

setInternalTextVars
{ BGMIn:shortName[zz] = (Pfr){(a+)}->shortName }
{ BGMIn:unitName[zz] = (Pfr){(a+)}->unitName }
{ BGMIn:dFormat [zz] = (Pfr){(a+)}->dFormat }
{ BGMIn:chardata[zz] = (Pfr){(a+)}->chardata }
}

// ----- set the number of channels
BGMIn: numChannels = 6;
zz = 0;

// --- steer string
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 2.0f // tic spacing
BGMIn: absU[zz] = -100.0f // min graph value
BGMIn: absU[zz] = 100.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- steer: 1vck (zack)
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 2.0f // tic spacing
BGMIn: absU[zz] = -6.5f // min graph value
BGMIn: absU[zz] = 6.5f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- steer: 1vck, 1R
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 23.0f // tic spacing

```

#if 0

```

BGMIn: absU[zz] = -135.0f // min graph value
BGMIn: absU[zz] = 135.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- steer: 1vck, 1R
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 23.0f // tic spacing
BGMIn: absU[zz] = -135.0f // min graph value
BGMIn: absU[zz] = 135.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- steer: 1vck, 1R
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 15.0f // tic spacing
BGMIn: absU[zz] = -70.0f // min graph value
BGMIn: absU[zz] = 70.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- steer: 1vck, 1R
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 15.0f // tic spacing
BGMIn: absU[zz] = -70.0f // min graph value
BGMIn: absU[zz] = 70.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setBackColorVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- Internal - 1d, 1v
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 50.0f // tic spacing
BGMIn: absU[zz] = -100.0f // min graph value
BGMIn: absU[zz] = 100.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setInternalTextVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- Internal - 1d, 1v
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 1.0f // tic spacing
BGMIn: absU[zz] = -4.0f // min graph value
BGMIn: absU[zz] = 4.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setInternalTextVars) setInternalTextVars) z++))
XYMin: numChannels))

// --- Internal - 1d, 1v
j = CAPCI.d, L = CAPCI.d, L = CAPCI.d
BGMIn: ticsU[zz] = 1.0f // tic spacing
BGMIn: absU[zz] = -4.0f // min graph value
BGMIn: absU[zz] = 4.0f // max graph value
BGMIn: absU[zz] = 1 // max graph value
BGMIn: absU[zz] = 1 // max graph value
setInternalTextVars) setInternalTextVars) z++))
XYMin: numChannels))
#endif

```

BrakePressure.c

```

// CAPC Controller Code: Brake Pressure servo I/O

// Last Update: April 1993 (GJ)

#include "servo.h"
#include "bluetooth.h"
#include "sysutil.h"
#include "brakePressure.h"

uchar bpdochkaum(uchar *bytearray, uint nbytes);
uchar bpdochkaum(uchar *bytearray, uint nbytes);
uchar testsum;

for (i=0; i<nbytes; i++)
testsum += (*bytearray++);

return(testsum);

void ExecuteBrakePressure( int wheel, int pressure ) {
uchar COMarr;
short doboth = 0;
}

```



```

    case IMSErr_BUF_OVERFLOW,
        strcpy(txt, "IMS ERROR - Buffer Overflow."); break;
    case IMSErr_IMS_ERROR:
        strcpy(txt, "IMS ERROR - Error from IMSI."); break;
    default:
        strcpy(txt, "IMS ERROR - unknown error."); break;
}

// -----
int TransmitPVTN( tType *pitch, tType *roll, tType *veloc, uchar Fn ) {
    IMSErr
    int
    uchar
    // build the data buffer (to Ntlo Ntlo Ntlo)
    if (pitch[0] < 0) pitch+=0.0001; else *pitch+=0.0001;
    if (roll[0] < 0) roll+=0.0001; else *roll+=0.0001;
    if (veloc[0] < 0) veloc+=0.0001; else *veloc+=0.0001;
    scaledPitch = (int) floor( *pitch / PITCH_COM_GAIN );
    scaledRoll = (int) floor( *roll / ROLL_COM_GAIN );
    scaledVeloc = (int) floor( *veloc / VELOC_COM_GAIN );
    // the frame count
    pdate[0] = (uchar) (Fn);
    // the pitch data
    pdate[1] = (uchar) ( scaledPitch & 0xFF );
    pdate[2] = (uchar) ( scaledPitch > 0xFF );
    // the roll data
    pdate[3] = (uchar) ( scaledRoll & 0xFF );
    pdate[4] = (uchar) ( scaledRoll > 0xFF );
    // the veloc data
    pdate[5] = (uchar) ( scaledVeloc & 0xFF );
    pdate[6] = (uchar) ( scaledVeloc > 0xFF );
    // send the packet
    IMSErr = SendIMSPacket( PTYPE_PVTN, pdate );
    return( IMSErr );
}

int ReceiveGeometry() {
    #define MAX_RETRY 8 // max number of times to try to rx lane geometry
    int IMSErr;
    int retries;
    char strMsg[100];
    // try a few times to get full IG packet
    retries = 0;
    IMSErr = ParseIMSPacket( 0 );
    while( IMSErr==IMSErr_TIMEOUT || (retries < MAX_RETRY) ) {
        return( IMSErr );
    }
}

// -----
uchar dochhsam(uchar *bytearray, uint nbytes);
short GlamaIntFromChar(uchar *mb, uchar *lab);
short GlamaIntToChar(uchar *mb, uchar *lab);
uint AddDoubleSize(uchar *data, uint len);
uchar dochhsam(uchar *bytearray, uint nbytes);
uchar testsum;
for (int i=0; i<nbytes; i++)
    testsum += *(bytearray+i);
return( testsum );
}

short GlamaIntFromChar(uchar *mb, uchar *lab) {
    return ( (( *mb ) << 8) & 0xFF00) | ( *lab );
}
unsigned short GlamaIntFromChar(uchar *mb, uchar *lab) {
    return ( (( *mb ) << 8) & 0xFF00) | ( *lab );
}

uint AddDoubleSize(uchar *data, uint len) {
    uint i, newcnt;
    uchar
    // copy old data
    for (i=0; i<len; i++) oldbuff[i]=data[i];
    // build new data
    for (i=0; newcnt=0; i<len; i++) {
        if ( oldbuff[i] != PCKCRF_10 )
            else( data[i + newcnt] = oldbuff[i] );
        data[i + newcnt] = oldbuff[i];
        newcnt++;
    }
    return( newcnt );
}

// -----
int ParseIMSPacket( bool InIt ) {
    OSTR
    bool found;
    int loc;
    int
    bool
    uchar
    uchar
    int16
    short
    unsigned short
    // incoming data buffering
    static uchar Incoming[IMAX_IMBYTES + 10];
    static int Incount;
    static int whichlane;
    // the base packet description
    IMSPacket
    LgpacketData Lgdata;
    LgLaneDataRecord Lgldr;
    LgDataPointRecord Lgdr;
    // initialize circular buffer
    if ( InIt ) {
        EmptySerialBuff( Incount );
        Incount = 0;
        Instart = 0;
        Inend = 0;
        whichlane = 0;
        return 0;
    }
    // initialize circular buffer
    if ( !Incount ) return IMSErr_NOT_ALIVE;
    // this is the slow version
    // first see if any more chars in serial buffer (MUST be < MAX_IMBYTES, or will overflow)
    while ( Incount < MAX_IMBYTES ) {
        if ( !Incount ) continue;
        if ( !Incount ) continue;
        if ( Incount >= MAX_IMBYTES-1 ) {
            Incount = 0;
            Instart = 0;
        }
    }
}

```



```

        InCount      = 0;
        InStart      = 0;
        PID_have_one = 0;
        WhichLane    = 0;
        break;
    }
    #endif // LMS_LOOP_THROUGH

    default:
        InCount      = 0;
        InStart      = 0;
        PID_have_one = 0;
        WhichLane    = 0;
        return LMSerr_BAD_PKTERR;
        break;
}

// ---- packet parsed, no reset indexes
InCount      = 0;
InStart      = 0;
PID_have_one = 0;
WhichLane    = 0;
return 0;
}

}

//-----
int sendLMSpacket( int packetType, uchar *data ) {
    uchar outgoing[ MAX_OUTBYTES ];
    uchar cHeader  sendBytes;
    long  kHeader  nBytes;

    if (!lan_active) return LMSerr_NOT_ACTIVE;

    // build the packet
    outgoing[0] = PACKET_ID;
    outgoing[1] = (uchar) packetType;
    switch(packetType) {
        case PTYPE_PVBN:
            sendBytes = 12;
            outgoing[2] = 0x00; // fixed data length
            outgoing[3] = 0x00; // frame count (0..255)
            outgoing[4] = data[0]; // pitch value (signed int)
            outgoing[5] = data[1];
            outgoing[6] = data[2];
            outgoing[7] = data[3];
            outgoing[8] = data[4];
            outgoing[9] = data[5];
            outgoing[10] = data[6]; // value value (unsigned int)
            outgoing[11] = docHeader( outgoing, (uint)sendBytes-1 );
            sendBytes += (long)kHeader;
            break;

        case PTYPE_CAL:
            long k; // 151,thout(100);
            uchar *p; // 151,thout(100);
            for (k=0; k<sendBytes; k++){
                printf( "0x%03d ", outgoing[k] );
                if (k%16 == 15) printf( "\n" );
            }
            break;

        case PTYPE_INTERRUPT:
            ParamLMSpacket( 1 ); // re-init pointers & counters
            sendBytes[2] = 0x00; // no data
            sendBytes[3] = 0x00;
            outgoing[4] = docHeader( outgoing, (uint)sendBytes-1 );
            kHeader = AddDoublePID( outgoing, (uint)sendBytes-2 );
            sendBytes += (long)kHeader;
            break;

        case PTYPE_STOP:
            sendBytes = 0;
            outgoing[2] = 0x00; // fixed length
            outgoing[3] = 0x00;
            break;
    }

    outgoing[4] = data[0]; // restore use 0x00 to execute cal
    outgoing[5] = docHeader( outgoing, (uint)sendBytes-1 );
    kHeader = AddDoublePID( outgoing, (uint)sendBytes-2 );
    sendBytes += (long)kHeader;
    break;
}

// keep a copy in the global buffer
for ( LMS_Out_nbytes=0; LMS_Out_nbytes<sendBytes; LMS_Out_nbytes++)
    LMS_Outgoing_buff[LMS_Out_nbytes] = outgoing[LMS_Out_nbytes];

// send the packet
if (sendBytes) return 0;
CHeader = FAST_write( lan_refout, (char*)outgoing, sendBytes );
if (CHeader) return LMSerr_BAD_WRITE;
else return 0;

}

void updateLaneGeometry( int detapt, uint xLoc, int yLoc ) {
    static lane = 0;

    // if init this lane line, xLoc is the type, yLoc is the ID
    if (detapt<0) {
        lane = (-detapt-1);
        if (lane>MAX_LANE_LINES) return;
        gic::solid [ lane ] = (xLoc);
        gic::id [ lane ] = (yLoc);
        return;
    }

    if (detapt==MAX_DATA_PTS) return;

    // store the values
    gic::solid [ lane ] = detapt+1;
    // callibrate
    gic::x [ lane ] = detapt; // (yLoc) * LANE_X_CON_GAIN;
    gic::y [ lane ] = (yLoc);
    return;
}

void FakeLMSdata( ) {
    int i;

    // Paul V.: please modify your "fake" code below.
    // For now, you should decode the data I've put
    // into my own global buffer.

    gic::id [n] = n; // int, ID (0..255) of n'th lane line, n=0 or 1
    gic::solid [n] = n; // int, 1 if n'th line is solid, 0 if dashed;
    gic::x [n] = n; // int, number of x'y coord in n'th lane line
    gic::y [n] = n; // int, number of x'y coord in n'th lane line
    gic::kate[n] = n; // double, 1'th y coord (n) in n'th lane line
    gic::kate[n] = gic::kate[n]; // double, 1'th y coord (n) in n'th lane line

    // Fake some LMS data:
    gic::solid[0] = false; // left LMS are not solid //
    gic::solid[1] = true; // right LMS are solid //
    gic::kate[0] = 20;
    gic::kate[1] = gic::kate[0];
    for (i = 0; i < gic::kate[0]; i++)
        gic::x[i][1] = 5.0 * i + 15.0;
}

```

CAPC Prototype control Code Listing


```

gLG.y0[i] = -0.5 * LaneWidth;
gLG.x1[i] = 5.0 * i + 15.0;
gLG.y1[i] = 0.5 * LaneWidth;

```

LMSserial.h

/* CAPC Prototype - LMSserial.h */

```

#ifndef CAPC_LMS_serial
#define CAPC_LMS_serial

```

```

#ifndef EXT
#define EXT extern
#endif

```

```

/*-----*/
/*          */
/*  CONSTANTS  */
/*          */
/*-----*/

```

```

#define NUM_LANE_LINES      2
#define MAX_DATA_PTS      30

```

```

#define MAX_OUTBYTES      1024
#define MAX_INBYTES      1024

```

```

// Packet types
#define PACKET_ID          ((uchar)0xAA)

```

```

#define PTTYPE_INIT        ((uchar)0x01) // misc. to the LMS
#define PTTYPE_CAL        ((uchar)0x02)
#define PTTYPE_PROBE      ((uchar)0x03)
#define PTTYPE_START      ((uchar)0x04)
#define PTTYPE_STOP       ((uchar)0x05)

```

```

#define PTTYPE_PRVFn      ((uchar)0x10) // P,R,V,Fn to the LMS

```

```

#define PTTYPE_STATUS     ((uchar)0x81) // lane geom. from the LMS
#define PTTYPE_LG        ((uchar)0x91)

```

// LMS error types

```

enum{
  LMSerr_NOT_ALIVE      = -1,
  LMSerr_BAD_WRITE     = -2,
  LMSerr_BAD_READ      = -3,
  LMSerr_BAD_CHKSUM    = -4,
  LMSerr_BAD_PTTYPE    = -5,
  LMSerr_TIMEOUT       = -6,
  LMSerr_BUFF_OVERFLOW = -7,
  LMSerr_LMS_ERROR     = -8
}

```

// LMS communication gains (in units/bit)
// multiply incoming (or divide outgoing) values by these

```

#define ROLL_COM_GAIN    0.01000 // 0.01 degrees/bit
#define PITCH_COM_GAIN  0.01000 // 0.01 degrees/bit
#define VELOC_COM_GAIN  0.50000  // 0.05 m/s/bit
#define LANE_X_COM_GAIN  0.01000 // 0.01 m/bit
#define LANE_Y_COM_GAIN  0.01000 // 0.01 m/bit

```

```

/*-----*/
/*          */
/*  STRUCTURES  */
/*          */
/*-----*/

```

```

EXT struct LaneMarkerSystem{
  bool  isAlive; // if the LMS is alive and responding
  short refIn; // input ref num of LMS port.
  short refOut; // output ref num of LMS port.

  uchar initialized; // status bytes
  uchar calibrated;
  uchar cameraHealth;
}

```

```

uchar frameCount;
uchar opMode;
uchar lmsError;
} lms;

```

```

typedef struct globalLaneGeometry{
  bool solid; NUM_LANE_LINES };
  bool id; [ NUM_LANE_LINES ];
  int nData; NUM_LANE_LINES };
  ftype x [ NUM_LANE_LINES ][ MAX_DATA_PTS ];
  ftype y [ NUM_LANE_LINES ][ MAX_DATA_PTS ];
} globalLaneGeometry;

```

// ---- packet structure descriptions

```

typedef struct LMSpacket{
  uchar *pmarker; // pointer to the packet marker byte
  uchar *pctype; // pointer to the packet type byte
  uchar *plength; // pointer to LSB of packet length byte-pair
  uchar *pdata; // pointer to 1st byte of packet data
  uchar *pchksm; // pointer to the packet checksum byte

  unsigned short dataBytes;
  //uint dataBytes;
} LMSpacket;

```

```

typedef struct LGpacketData{
  uint nRecords; // number of lane records ( = *LMS.pdata[0] )
  uint frameCount; // frame count for lane data ( = *LMS.pdata[1] )
} LGpacketData;

```

```

typedef struct LGLaneDataRecord{
  uchar lanetype; // type of lane this record is for
  uchar laneID; // a number for the lane's ID (0..255)
  uchar nDpRecs; // number of data-point records
  uchar *dpRec; // first data-point record
} LGLaneDataRecord;

```

```

typedef struct LGdataPointRecord{
  uchar *xloc; // x data location
  uchar *yloc; // y data location
} LGdataPointRecord;

```

```

/*-----*/
/*          */
/*  GLOBALS  */
/*          */
/*-----*/

```

```

EXT uchar LMS_Outgoing_Buff[ MAX_OUTBYTES ];
EXT long LMS_Out_numbytes;
EXT uchar LMS_Incoming_Buff[ MAX_INBYTES ];
EXT long LMS_In_numbytes;
EXT globalLaneGeometry gLG;

```

```

/*-----*/
/*          */
/*  PROTOTYPES  */
/*          */
/*-----*/

```

```

int TransmitPRVFn( ftype *pitch, ftype *roll, ftype *veloc, uchar Fn );
int ReceiveLaneGeometry( void );
int ParseLMSpacket( bool init );
int SendLMSpacket( int packetType, uchar *data );
void UpdateLaneGeometry( int dataPt, uint xloc, int yloc );
void GiveLMSerrText( int LMSerr, char *txt );

```

// temporary prototype
void FakeLMSdata(void);

#endif

LMSToolsDlg.cp

```

/*-----*/
/* CAPC Controller Code: LMS Tools Dialog */
/*          */
/* Last Update: 5 March 1995 (GJ) */
/*-----*/

```

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

```



```

#undef CHARS_PER_ROW
// -----
void LMSToolsDlg::LMSStatusOnScreen() {
    long k;
    char theout[50] = "";

    MyFontInfo saveFontInfo;
    MyFontInfo newFontInfo;

    MySetFont(fWindowP,saveFontInfo); // get the current window font information

    newFontInfo.font = monaco;
    newFontInfo.size = 9;
    newFontInfo.face = 0;
    newFontInfo.mode = srcCopy;
    MySetFont(fWindowP,newFontInfo); // set the new window font information

#define STATUS_XLOC 200
#define STATUS_YLOC 20
#define STATUS_IMCR 14

    sprintf(theout, "LMS is alive %d ", (int) lms.isAlive );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 0*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

    sprintf(theout, "Initialized %02X ", (uint) lms.initialized );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 1*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

    sprintf(theout, "Calibrated %02X ", (uint) lms.calibrated );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 2*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

    sprintf(theout, "Camera Health %02X ", (uint) lms.cameraHealth );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 3*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

    sprintf(theout, "Frame Count %02X ", (uint) lms.frameCount );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 4*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

    sprintf(theout, "LMS Error %02X ", (uint) lms.lmsError );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 5*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

    sprintf(theout, "Run Mode %02X ", (uint) lms.opMode );
    MoveTo( STATUS_XLOC, STATUS_YLOC + 6*STATUS_IMCR );
    DrawString(CtoPtr(theout)); PtoCatr((uchar*)theout);

#undef STATUS_XLOC
#undef STATUS_YLOC
#undef STATUS_IMCR

    MySetFont(fWindowP,saveFontInfo); // set the new window font information
}

/* -----
Constructor & Destructor
----- */
LMSToolsDlg::LMSToolsDlg(short resID)
{
    #ifdef MW_12
        fState = new State("LMSToolsDlg class");
    #endif
    #ifdef THINK_7
        fState = new State("LMSToolsDlg class");
    #endif

    if (fState)
        MyError err(kClassAllocErr);

    DoInit(resID);

    if (fState->Get() != kNoErr)
        fState->Display();
}

LMSToolsDlg::~LMSToolsDlg(void)
{
    DisposeWindow(fWindowP);
    delete fState;
}

```

```

/* -----
DoInit
----- */
void LMSToolsDlg::DoInit(Int16 resID)
{
    if (!ModalDialog::DoInit(resID))
        fState->Set(kClassAllocErr);
}

/* -----
DoClose.
----- */
void LMSToolsDlg::DoClose(void)
{
    HideWindow(fWindowP);
}

void LMSToolsDlg::DoSelect(void)
{
    ShowWindow(fWindowP);
    DoProcess();
}

void DumpLMSData( int mode );
void DumpLMSData( int mode ){
    static FILE *fTXT0,*fTXT1;
    int i;

    switch(mode){
        case 1: // open file
            fTXT0 = fopen("LMS_Text_Out0","w");
            fTXT1 = fopen("LMS_Text_Out1","w");
            fprintf(fTXT0, "Fn\tID\tType\tMpts\t(X,Y) Data Points...\n");
            fprintf(fTXT1, "Fn\tID\tType\tMpts\t(X,Y) Data Points...\n");
            break;

        case 0: // write data
            if (fTXT0==NULL) break;
            if (fTXT1==NULL) break;

            // the header data
            fprintf(fTXT0, "%d\t%d\t%d\t%d", (int)lms.frameCount, (int)gLg.Id[0], (int)gLg.solid[0], (int)gLg.Mdata[0] );
            fprintf(fTXT1, "%d\t%d\t%d\t%d", (int)lms.frameCount, (int)gLg.Id[1], (int)gLg.solid[1], (int)gLg.Mdata[1] );

            // the xy coords
            for (i=0; i<(int)gLg.Mdata[0]; i++)
                fprintf(fTXT0, "(%.2f,%.2f)\t", gLG.x[0][i], gLG.y[0][i] );
            for (i=0; i<(int)gLg.Mdata[1]; i++)
                fprintf(fTXT1, "(%.2f,%.2f)\t", gLG.x[1][i], gLG.y[1][i] );

            // end of line
            fprintf(fTXT0, "\n");
            fprintf(fTXT1, "\n");

            break;

        case -1: // close file
            if (fTXT0==NULL) break;
            if (fTXT1==NULL) break;
            fclose( fTXT0 );
            fclose( fTXT1 );
            break;
    }
}

/* -----
DoClose.
----- */
short LMSToolsDlg::DoProcess(void)
{
    GrafPtr oldPort;
    short it=Hit;
    Boolean done = false;
    Rect theRect;
    Handle hHandle;
    short iType;
    fType pitchVal = 0.0,
}

```



```

    &err ) );
    dRF = AD2EU ( hcr[CAPCI.def1_RF], analog_input( (as+CAPCI.def1_RF)->ADchan,
    uRF = AD2EU ( hcr[CAPCI.u_RF], analog_input( (as+CAPCI.u_RF)->ADchan, &err ) );
    uLF = AD2EU ( hcr[CAPCI.u_LF], analog_input( (as+CAPCI.u_LF)->ADchan, &err ) );
    velocVal = 0.5 * (uRF + uLF);
    pitchVal = 5.0E-4 * R2D * (dLR - dLF + dRR - dRF) / Wb;
    rollVal = 5.0E-4 * R2D * ( (dRR - dLF) / (1.167 * Tw1) * (1.0 + RollRatio1) +
    * Tw2) * (1.0 + RollRatio2));
    printf(tmpstr, "%4.2f", pitchVal);
    SetDialogString(fWindowP, kPitchET, tmpstr);
    printf(tmpstr, "%4.2f", rollVal);
    printf(tmpstr, "%5.1f", velocVal);
    SetDialogString(fWindowP, kRollET, tmpstr);
    SetDialogString(fWindowP, kVelocET, tmpstr);
}
TTLoutState = !TTLoutState;
digital_output( CAPC_DOUT_TTL, TTLoutState, &err16 );
SetControlState( fWindowP, kFrankClkCB, TTLoutState );
LMSerr = TransmitPRVFn( &pitchVal, &rollVal, &velocVal, 0 );
LMSMessageOnScreen("Transmitted PAR");
showOut = 1;
break;
case kRx8:
    LMSerr = ParseLMSPacket( 0 );
    LMSMessageOnScreen("Received Packet");
    showLG = GetControlState(fWindowP, kDecipherCB);
    showIn = 1;
    EmptySerialBuff( lms.refIn );
    break;
case kEmptyB:
    EmptySerialBuff( lms.refIn );
    break;
case kRunB:
    // if option key held, do about 2hz, else do about the control freq
    atcopy( runmsg, "Running @ 10Hz (click mouse to stop)..." );
    runDelay = (ulong)( 60 / max.CtrlFreq ) - 2;
    runDelay *= 1;
    if (OptionKeyHeld){
        runDelay *= 2;
        atcopy( runmsg, "Running @ 2Hz (click mouse to stop)..." );
    }
    if (ShiftKeyHeld){
        runDelay *= 4;
        atcopy( runmsg, "Running @ 1/2 Hz (click mouse to stop)..." );
    }
    LMSMessageOnScreen( runmsg );
    // check for saving data file
    if ( GetControlState(fWindowP, kSaveLMSDataCB) ){
        doSave = 1;
        DumpLMSData( 1 );
    }
    else doSave=0;
    // init some stuff
    showLG = GetControlState(fWindowP, kDecipherCB);
    digital_output( CAPC_DOUT_TTL, TTLoutState, &err16 );
    SetControlState(fWindowP, kFrankClkCB, TTLoutState);
    Delay(30, &finalTick);
    EmptySerialBuff( lms.refIn );
    wasRunErr = 0;
    runFn = 0;
    mydataptr = (uchar) 0;
    LMSerr = SendLMSPacket( PTYPE_START, &mydataptr );
    do{
        if ( OptionKeyHeld ){
            GetDialogString(fWindowP, kPitchET, pStr); PtoCStrCopy(astringVal, pStr);
            pitchVal = atof( stringVal );
            GetDialogString(fWindowP, kRollET, pStr);
            PtoCStrCopy(astringVal, pStr);
            rollVal = atof( stringVal );
            GetDialogString(fWindowP, kVelocET, pStr); PtoCStrCopy(astringVal, pStr);
            velocVal = atof( stringVal );
        }
        else{
            ftype dLR, dRR, dLF, dRF, uRF, uLF;
            char tmpstr[20];
            dLR = AD2EU ( hcr[CAPCI.def1_LR], analog_input( (as+CAPCI.def1_LR)
            >ADchan, &err ) );
            dRR = AD2EU ( hcr[CAPCI.def1_RR], analog_input( (as+CAPCI.def1_RR)

```

```

            >ADchan, &err ) );
            dLF = AD2EU ( hcr[CAPCI.def1_LF], analog_input( (as+CAPCI.def1_LF)
            >ADchan, &err ) );
            uRF = AD2EU ( hcr[CAPCI.u_RF], analog_input( (as+CAPCI.u_RF)->ADchan,
            &err ) );
            uLF = AD2EU ( hcr[CAPCI.u_LF], analog_input( (as+CAPCI.u_LF)->ADchan,
            &err ) );
            velocVal = 0.5 * (uRF + uLF);
            pitchVal = 5.0E-4 * R2D * (dLR - dLF + dRR - dRF) / Wb;
            rollVal = 5.0E-4 * R2D * ( (dRR - dLF) / (1.167 * Tw1) * (1.0 +
            RollRatio1) +
            (1.104 * Tw2) * (1.0 + RollRatio2));
            printf(tmpstr, "%4.2f", pitchVal);
            SetDialogString(fWindowP, kPitchET, tmpstr);
            printf(tmpstr, "%4.2f", rollVal);
            SetDialogString(fWindowP, kRollET, tmpstr);
            printf(tmpstr, "%5.1f", velocVal);
            SetDialogString(fWindowP, kVelocET, tmpstr);
        }
        runFn++;
        // toggle TTL line,
        TTLoutState = !TTLoutState;
        digital_output( CAPC_DOUT_TTL, TTLoutState, &err16 );
        SetControlState(fWindowP, kFrankClkCB, TTLoutState);
        // send the pitch and roll, show it, add some delay
        TransmitPRVFn( &pitchVal, &rollVal, &velocVal, runFn );
        LMSOutBufferOnScreen();
        Delay(1, &finalTick);
        // try many times to get full LG packet
        retries = 0;
        do{
            LMSerr = ParseLMSPacket( 0 );
            retries++;
        } while( (LMSerr<0) && (LMSerr==LMSerr_TIMEOUT) && (retries < 1000) );
        // check for saving data file
        if (LMSerr==0)
            if (doSave) DumpLMSData( 0 );
        if (LMSerr<0){
            wasRunErr = 1;
            LMS_in_nbytes = 0;
            GimmelMSERText( LMSerr, errMsg );
            LMSMessageOnScreen( errMsg );
            LMSerr = ParseLMSPacket( 1 ); // re-init pointers & counters!
        }
        else if (wasRunErr){
            LMSMessageOnScreen( runmsg );
            wasRunErr = 0;
        }
        if ( showLG )
            LMSLaneGeomOnScreen( );
        else
            LMSInBufferOnScreen( );
        // wait approx cycle time
        Delay(runDelay, &finalTick);
        // check for abort
        GetNextEvent( everyEvent, &myEvent );
    } while (myEvent.what != mouseDown);
    if (doSave) DumpLMSData( -1 );
    mydataptr = (uchar) 0;
    LMSerr = SendLMSPacket( PTYPE_STOP, &mydataptr );
    LMSMessageOnScreen("Running stopped.");
    break;
case kWantReplyCB:
    ToggleControl(fWindowP, kWantReplyCB);
    EmptySerialBuff( lms.refIn );
    break;
case kFrankClkCB:
    TTLoutState = !TTLoutState;
    digital_output( CAPC_DOUT_TTL, TTLoutState, &err16 );
    SetControlState(fWindowP, kFrankClkCB, TTLoutState);
    break;
case kDecipherCB:
    ToggleControl(fWindowP, kDecipherCB);

```

CAPC Prototype Control Code Listing

```

        break;
    case kSaveLMSTextCB:
        ToggleControl( fWindowP, kSaveLMSTextCB );
        break;
    }

    // check for any errors
    if ( LMSerr < 0 ) {
        GMessageErrText( LMSerr, errMag );
        LMSMessageOnScreen( errMag );
    }

    // always update the status
    LMSStatusOnScreen( );

    // check for display of out-going buffer
    if ( showOut )
        LMSOutBufferOnScreen();

    // check for display of in-coming buffer (clear if not)
    if ( showIn ) {
        if ( showLG )
            LMSLaneGeomOnScreen( );
        else
            LMSInBufferOnScreen( );
    }
    else {
        LMS_In_numbytes = 0;
        LMSInBufferOnScreen( );
    }

} // end of while(!done)

DoClose();
SetPort(oldPort);
return itemHit;
}

```

LMSToolsDlg.hp

```

#ifndef LMSTOOLS_DLG_
#define LMSTOOLS_DLG_

#include "ModalDialog.hp"
#include "State.hp"
#include "MyAppDS.h"

class LMSToolsDlg : public ModalDialog
{
private:
    State *fState;

public:
    LMSToolsDlg(short resID);
    ~LMSToolsDlg(void);

    void DoInit(Int16 resID);
    void DoClose(void);
    void DoSelect(void);
    short DoProcess(void);

    void LMSMessageOnScreen( char *thetext );
    void LMSStatusOnScreen( void );

    void BufferOnScreenEngine( char *buf, int Yloc );
    void LMSInBufferOnScreen( void );
    void LMSOutBufferOnScreen( void );
    void LMSLaneGeomOnScreen( void );
};
#endif

```

Appendix E

Computer code listing:
Lane-Mark-Sensor (LMS) Software Code

960 Part 4 . Rel 0 G:\LMS\APP\CAPC.SRC

```
0 \ polyFORTH ISD-4 PolyForth Extensions glg ( 03 Apr 1994)
1 \ ^^OPTION Copyright 1994 (C) by Forth, Inc. & ERIM^
2
3 2 OPTION" CAPC Loads CAPC Source."
4
5 INSTALL CAPC_COM OMIT COMCLOCK
6 INSTALL FRAMEGRABBER
7 INSTALL CAMERA INSTALL INVERTED
8 INSTALL DIAGNOSTICS OMIT IMGSEQ
9 OMIT FAKEDATA
10
11
12
13
14
15
```

961 Part 4 Rel 1 G:\LMS\APP\CAPC.SRC

```
0 ?CAPC Displays these CAPC Emulator instructions.
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

962 Part 4 Rel 2 G:\LMS\APP\CAPC.SRC

```
0 ( CAPC Load Block glg 15 Feb 1995)
1 -1 +B H: ?CAPC ; ?CAPC
2 TIMERS LOAD
3 5 +P LOAD ( TIMERS Background Task )
4
5 [+ FRAMEGRABBER PCI LOAD MUTECH LOAD +]
6
7 1 +B LOAD
8 60 +P LOAD
9 [+ DIAGNOSTICS 102 +P 108 +P THRU
10 [+ CAMERA 114 +P 116 +P THRU ( Exposure Setting )
11 117 +P 118 +P THRU ( Fading Memory Filter ) +] +]
12 2 +B LOAD
13
14
15
```

963 Part 4 Rel 3 G:\LMS\APP\CAPC.SRC

```

0 ( CAPC Load Block glg 15 Feb 1995)
1
2
3 [+ CAPC_COM
4
5 8 +P LOAD ( UART Facility Variable )
6 9 +P 14 +P THRU ( GetPacket, SendPacket )
7 6 +P LOAD ( OBUF )
8 15 +P 32 +P THRU ( Commands, DisplayPacket )
9
10 +)
11
12
13
14
15

```

964 Part 4 Rel 4 G:\LMS\APP\CAPC.SRC

```

0 ( CAPC Load Block glg 15 Feb 1995)
1 33 +P 36 +P THRU ( Lane Geometry Structure )
2 81 +P 89 +P THRU ( Lane Geometry Extraction )
3 42 +P LOAD ( Tracking Logic )
4 90 +P 101 +P THRU ( Lane Geometry Extraction )
5
6 37 +P 39 +P THRU ( Lane Geometry Processing )
7
8 [+ FAKEDATA
9 42 IN ..\APP\CAPC_EM 45 IN ..\APP\CAPC_EM THRU
10 48 IN ..\APP\CAPC_EM 49 IN ..\APP\CAPC_EM THRU ( Lane Geo)
11 +)
12 50 +P 53 +P THRU ( MainLoop )
13 111 +P 113 +P THRU ( CAPC Init )
14 47 +P 49 +P THRU ( Initialization )
15

```

965 Part 4 Rel 5 G:\LMS\APP\CAPC.SRC

```

0 ( TIMERS Initialization glg 18 Mar 1994 )
1
2 ( u n s r )
3 112 0 1024 512 BACKGROUND WATCHER
4 WATCHER BUILD
5
6 : WATCHING WATCHER WATCHES ;
7
8 : -WATCHING WATCHER HALT ;
9
10 0 INITIALIZES WATCHING
11
12
13
14
15

```

The WATCHER background task monitors the timers. The TIMERS utility uses CATCH which requires the USER variable CATCHER. The TIMERS utility cannot be TURNKEY or TARGET compiled in its present state.

966 Part 4. Rel 6 G:\LMS\APP\CAPC.SRC

```

0 ( OBUF      glg      03 Mar 1995)
1
2 : ?#in NEW-IN @ OLD-IN @ - DUP
3      0< IF MAX-IN @ 1+ + THEN ;
4
5 : .IBUF BUF-IN OLD-IN @ + ?#in DUMP ;
6
7 : OBUF 0 DUP OLD-IN ! NEW-IN ! ;
8
9 : ?IBUF ?#in ?DUP IF COUNTER
10      BEGIN DUP uKEY . 5 SPACES TIMER AGAIN
11      THEN DROP OBUF ;
12 : ?IBUFS BEGIN ?IBUF ?ESC AGAIN ;
13
14
15

```

967 Part 4 Rel 7 G:\LMS\APP\CAPC.SRC

```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```

968 Part 4 Rel 8 G:\LMS\APP\CAPC.SRC

```

0 ( UART Access Control  glg  01 Apr 1994)
1 FACILITY UART
2
3 : [U UART GET ; : U] UART RELEASE ;
4
5 : FREE ( a - ) 0 SWAP ! ;
6
7 VARIABLE 'xt  VARIABLE xTIME
8
9 : remoteEXECUTE
10 SECOND ACTIVATE UART GRAB COUNTER >R 'xt @EXECUTE
11      COUNTER R> - xTIME ! U] NOD ;
12
13
14
15

```

UART is the facility variable for the UART receiver.
 It is used to sequence task execution between the OPERATOR task and the REMOTE (RS-232) task.

'xt is an execution token used by the REMOTE task. Its value is set by the OPERATOR task.

xTIME contains the execution time of the vectored 'xt behavior.

It is essential that the REMOTE task GRAB the facility variable and not first traverse the multiprogramming loop as GET does.

969 Part 4. Rel 9 G:\LMS\APP\CAPC.SRC

```

0 ( uKEY, uEMIT, u?KEY   glg   07 Mar 1995)
1
2 \ 54 +P 55 +P THRU   ( Diagnostic defs )
3 56 +P LOAD ( uEMIT )
4
5 : uKEY KEY ;
6
7 : u?KEY ?KEY ;
8
9
10
11
12
13
14
15

```

970 Part 4 Rel 10 G:\LMS\APP\CAPC.SRC

```

0 ( pKEY, pEMIT   glg   03 Mar 1995)
1 HEX
2 VARIABLE dbl
3
4 : pKEY uKEY dbl @ IF
5     ( AA = NOT IF bad packet EXIT )
6     DROP 0 dbl ! uKEY
7     ELSE DUP AA = IF 1 dbl ! THEN
8     THEN ;
9
10 : pEMIT ( c - ) DUP AA = IF AA uEMIT THEN uEMIT ;
11
12
13
14
15

```

971 Part 4 Rel 11 G:\LMS\APP\CAPC.SRC

```

0 ( [sync], *Len   glg   15 Feb 1995)
1 HEX
2 VARIABLE PacketBuffer 401 ALLOT VARIABLE PacketLen
3 VARIABLE crcError VARIABLE BadPacket
4
5 : [sync] ( - cmd ) BEGIN u?KEY ?DUP IF AA = IF
6     uKEY DUP AA = IF DROP ELSE EXIT THEN THEN THEN
7     AGAIN ;
8
9 : @Len ( crc - crc n ) 0 dbl ! pKEY pKEY OVER >< OVER +
10 0 MAX 400 MIN >R ++ R> ;
11
12 : >PB ( - a ) PacketBuffer PacketLen " + .
13
14
15

```

972 Part 4 Rel 12 G:\LMS\APP\CAPC.SRC

```

0 ( GetPacket      glg   15 Feb 1995)
1 HEX
2 : GetPacket ( - cmd ) [sync] DUP AA + ( crc )
3 @Len DUP PacketLen !
4 ?DUP IF 0 DO pKEY DUP
5
6         PacketBuffer I + C! +
7         LOOP
8     THEN
9
10    FF AND pKEY DUP >PB C!
11        - NOT IF 1 crcError +! DROP AA THEN
12 dbl @ IF uKEY DROP THEN ;
13
14
15

```

973 Part 4 Rel 13 G:\LMS\APP\CAPC.SRC

```

0 ( Listen, ?Packet      glg       07 Mar 1995)
1 VARIABLE PacketReady VARIABLE PacketType
2 : (Listen)  UART GRAB GetPacket PacketType !
3         1 PacketReady ! U] ;
4
5 : listen  UART @ NOT IF
6         SECOND ACTIVATE (Listen) STOP THEN ;
7 : Listen listen ;
8
9 : ?Packet ( - cmd ) PacketReady @ IF 0 PacketReady !
10        PacketType @ ELSE 0 THEN ;
11
12 : -Listen SECOND HALT UART FREE ;
13
14
15

```

974 Part 4 Rel 14 G:\LMS\APP\CAPC.SRC

```

0 ( SendPacket      glg   16 Feb 1995)
1 HEX VARIABLE SendStruct CELL ALLOT
2 : pSend OVER CONSTANT + DOES> @ SendStruct + ;
3 0 1 pSend SendCmd 2 pSend SendLen CELL pSend SendAddr
4 1 pSend SendCRC CONSTANT |SendStruct|
5 : [H] ( crc n - crc ) DUP >< DUP pEMIT + DUP pEMIT + ;
6 : [B] ( crc c - crc ) DUP pEMIT + ;
7
8 : (SendPacket) ( - ) AA DUP uEMIT ( crc )
9 SendCmd C@ DUP uEMIT + SendLen H@ [H] SendLen H@
10 ?DUP IF 0 DO SendAddr @ I + C@ [B] LOOP THEN
11 DUP SendCRC C! pEMIT ;
12
13 : SendPacket ( cmd a n - ) SendLen H! SendAddr ! SendCmd C!
14 (SendPacket) ;
15

```

975 Part 4. Rel 15 G:\LMS\APP\CAPC.SRC

```
0 ( Command Vectors glg 15 Feb 1995)
1 HEX
2 VARIABLE UCMS 3FC ALLOT
3
4 : uInit UCMS 400 ERASE ;
5 0 INITIALIZES uInit
6
7 : UCMD ( n ) : LAST @ @ CFA 4+ SWAP 4* UCMS + ! ;
8
9 : ExecuteCmd ( n ) 4* UCMS + @EXECUTE ;
10
11
12
13
14
15
```

976 Part 4 Rel 16 G:\LMS\APP\CAPC.SRC

```
0 ( Misc. Packet Commands glg 15 Feb 1995)
1 HEX
2 AA : UCMD bell 1 BadPacket +! BELL ;
3
4 07 : UCMD Ptype PacketBuffer PacketLen @ TYPE ;
5
6
7
8
9
10
11
12
13
14
15
```

977 Part 4 Rel 17 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

978 Part 4 Rel 18 G:\LMS\APP\CAPC.SRC

```
0 ( STATUS glg 03 Mar 1995)
1 HEX
2 VARIABLE LMSstatus 2 ALLOT
3
4 : pStat OVER CONSTANT + DOES> @ LMSstatus + ;
5
6 0 1 pStat ?Initialized
7 1 pStat ?Calibrated
8 1 pStat ?CameraReady
9 1 pStat FrameCount
10 1 pStat Error
11 1 pStat LMSmode CONSTANT |LMSstatus|
12
13
14
15
```

979 Part 4 Rel 19 G:\LMS\APP\CAPC.SRC

```
0 ( PROBE glg 03 Mar 1995)
1 HEX
2
3 03 :UCMD status 81 LMSstatus |LMSstatus| SendPacket ;
4
5
6
7
8
9
10
11
12
13
14
15
```

980 Part 4 Rel 20 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

981 Part 4. Rel 21 G:\LMS\APP\CAPC.SRC

```

0 ( Default Clocking      glg          07 Mar 1995)
1
2 VARIABLE Armed
3 : ?FrameClock ( - f ) Armed @ ;
4
5 [- FRAMEGRABBER [- COMCLOCK
6 VARIABLE ClockPeriod
7
8 : Clocking BEGIN ClockPeriod @ MS
9   LMSmode C@ IF 1 FrameCount C+! 1 Armed ! THEN AGAIN ;
10
11 : ClockInit 100 ClockPeriod ! ;
12 0 INITIALIZES ClockInit
13
14 -) -]
15

```

982 Part 4 Rel 22 G:\LMS\APP\CAPC.SRC

```

0 ( COM Port FrameClock 15 Feb 1995)
1 HEX
2 [+ COMCLOCK
3
4 : (?FrameClock) ( - f ) MSR INPUT 11 AND .11 XOR NOT ;
5
6 : Clocking COM2 MSR INPUT DROP
7   BEGIN (?FrameClock) LMSmode C@ AND IF -RTS
8     1 FrameCount C+! 1 Armed ! THEN PAUSE AGAIN ;
9 +]
10
11 : +RTS MCR INPUT 2 OR MCR OUTPUT ;
12 : -RTS MCR INPUT FD AND MCR OUTPUT ;
13 : -RTS MCR INPUT 2 XOR MCR OUTPUT ;
14
15

```

983 Part 4 Rel 23 G:\LMS\APP\CAPC.SRC

```

0 ( FrameGrabber Clocking  glg          07 Mar 1995)
1
2 [+ FRAMEGRABBER
3
4 : Clocking [capture BEGIN capture] LMSmode C@ IF
5   -RTS 1 FrameCount C+! 1 Armed ! THEN [capture AGAIN ;
6 +]
7
8
9
10
11
12
13
14
15

```


984 Part 4 Rel 24 G:\LMS\APP\CAPC.SRC

```
0 ( FrameClock BACKGROUND task glg          07 Mar 1995)
1
2 ( u n s r )
3 #USER 0 512 96 BACKGROUND FrameClock
4 FrameClock BUILD
5
6 : +FrameClock FrameClock ACTIVATE Clocking ;
7
8 : -FrameClock FrameClock HALT ;
9
10
11
12
13
14
15
```

985 Part 4 Rel 25 G:\LMS\APP\CAPC.SRC

```
0 ( Display Packet glg 17 Feb 1995)
1
2 : DisplayPacket ( cmd - cmd ) C# 2@ 2>R
3 CORNER @ >R TOP 2@ 2>R [Corner] 2C@ CORNER 2C! 0. TOP 2C!
4 4 0 TAB ." Packet Type " DUP 4 .R CR
5 ( PacketBuffer PacketLen @ 1+ DUMP )
6 2R> TOP 2! R> CORNER ! 2R> C# 2! ;
7
8 : .FrameCount C# 2@ 2>R
9 CORNER @ TOP 2@ [Corner] 2C@ CORNER 2C! 0. TOP 2C!
10 0 0 TAB ." Frame Count " FrameCount C@ 4 .R 21 SPACES
11 TOP 2! CORNER ! 2R> C# 2! ;
12
13
14
15
```

986 Part 4 Rel 26 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5                               Should a pending clock transition be processed?
6                               Should Armed variable be zeroed here?
7
8
9
10
11
12
13
14
15
```

987 Part 4 . Rel 27 G:\LMS\APP\CAPC.SRC

```
0 ( START, STOP          glg  06 Mar 1995)
1 HEX
2 VARIABLE 'start VARIABLE 'stop
3
4 04 :UCMD StartLMS
5          0 FrameCount C! 'start @EXECUTE          Should a pending clock transition be processed?
6          1 LMSmode C! +FrameClock ;              Should Armed variable be zeroed here?
7
8 05 :UCMD StopLMS 'stop @EXECUTE 0 LMSmode C! -FrameClock
9          0 ?Initialized C! ;
10
11
12
13
14
15
```

988 Part 4 Rel 28 G:\LMS\APP\CAPC.SRC

```
0 ( CALIBRATE          glg  16 Feb 1995)
1 HEX
2
3 02 :UCMD calibrate 1 ?Calibrated C! ;
4
5
6
7
8
9
10
11
12
13
14
15
```

989 Part 4 Rel 29 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

990 Part 4 . Rel 30 G:\LMS\APP\CAPC.SRC

```

0 ( Pitch & Roll      glg      07 Mar 1995)          Pitch and Roll units are 0.01 degree
1
2 VARIABLE P&Rdata  1 ALLOT
3
4 : p&rItem OVER CONSTANT + DOES> @ P&Rdata + ;
5
6 0  1 p&rItem  P&Rcount
7    2 p&rItem  Pitch
8    2 p&rItem  Roll
9
10 CONSTANT |P&Rdata|
11
12
13
14
15

```

991 Part 4 Rel 31 G:\LMS\APP\CAPC.SRC

```

0 ( .P&R , @Pitch>Rad      glg      12 May 1995)          Pitch and roll in 100 ths of a degree are converted to
1                                                              microradians using a rational approximation for PI/180.
2 : .P&R C# 2@
3     CORNER @ TOP 2@ [Corner] 2C@ CORNER 2C!  0. TOP 2C!
4     Pitch H@  1 0 TAB  ." Pitch "  10 .R
5     Roll H@    CR  ." Roll "  10 .R
6     P&Rcount C@ CR  ." P&R count "  10 .R
7     TOP 2! CORNER ! C# 2! ;
8
9 : @Pitch>Rad ( - n ) Pitch H@ 698131701 4000000 */ ;
10
11 : @Roll>Rad ( - n ) Roll H@ 698131701 4000000 */ ;
12
13 : .LMSdata .FrameCount .P&R ;
14
15

```

992 Part 4 Rel 32 G:\LMS\APP\CAPC.SRC

```

0 ( Pitch & Roll      glg      07 Mar 1995)          Live pitch and roll data is stored only if the camera is enabled
1
2 CODE ><H@ ( a - h)  W POP  h W ) 0 MOV
3     0 hi 0 XCHG  h 0 1 0 MOV SX  0 PUSH  NEXT
4
5 HEX
6
7 10 :UCMD p&r (+ CAMERA
8     PacketBuffer  C@  P&Rcount C!
9     PacketBuffer 1+ ><H@ Pitch H!
10    PacketBuffer 3 + ><H@ Roll H! .P&R +) ;
11
12
13
14
15

```

993 Part 4- Rel 33 G:\LMS\APP\CAPC.SRC

```

0 ( LG Structure      glg      10 Mar 1995)
1 VARIABLE 'LgBuffer
2
3 : pLg OVER CONSTANT + DOES> @ 'LgBuffer @ + ;
4
5 0 1 pLg LgFrameCount      1 pLg #LaneRecords
6   0 pLg ldr                CONSTANT |LgHeader|
7
8 VARIABLE ldrPTR
9 : pLDR OVER CONSTANT + DOES> @ ldrPTR @ + ldr + ;
10
11 0 1 pLDR LaneType         1 pLDR LaneID
12   1 pLDR #DataPairs       0 pLDR DataPairs
13 CONSTANT |LDRheader|
14
15

```

994 Part 4 Rel 34 G:\LMS\APP\CAPC.SRC

```

0 ( LG Buffers      glg      10 Mar 1995)
1 1025 CONSTANT |LgBuffer|
2 VARIABLE (LgBuffer) |LgBuffer| ALLOT
3
4 : LgBuffer ( n - a ) 2 MOD
5   [ |LgBuffer| 2/ ] LITERAL * (LgBuffer) + ;
6
7 VARIABLE (LG) CELL ALLOT
8
9 : LgLen ( n ) 2 MOD 4* (LG) + ;
10
11 : #Buffer FrameCount C@ 01 AND ;
12
13 : DataLen [ |LgBuffer| |LgHeader| - |LDRheader| - ] LITERAL ;
14
15

```

The lane geometry buffer is sized for 256 data pairs.

Currently a max of 16 data pairs per lane are produced.

The number of pairs is limited by the RS-232 transfer time.

995 Part 4 Rel 35 G:\LMS\APP\CAPC.SRC

```

0 ( LgBuffer Init   glg      16 Mar 1995)
1
2 : LgInit
3   0 LgBuffer |LgBuffer| ERASE
4   0 LgLen 2 CELLS ERASE
5   #Buffer LgBuffer 'LgBuffer !
6   |LgHeader| #Buffer LgLen ! ;
7
8 : NewBuffer #Buffer LgBuffer 'LgBuffer ! ;
9
10
11
12
13
14
15

```

996 Part 4 - Rel 36 G:\LMS\APP\CAPC.SRC

```

0 ( #!          glg    13 Mar 1995)
1
2 VARIABLE PairPTR
3
4 : #! ( n ) >< PairPTR @ SWAP OVER DataPairs + H!
5         2+ 1023 AND PairPTR ! ;
6
7 : !Pair ( cross down ) #! #! ;
8
9 : @Pair ( - cross down ) PairPTR @ DataPairs + 2- DUP
10         ><H@ SWAP 2- ><H@ ;
11
12
13
14
15

```

997 Part 4 Rel 37 G:\LMS\APP\CAPC.SRC

```

0 ( @LaneData  glg    13 Mar 1995)
1 [- FAKEDATA
2 : @LeftLaneData
3     0 PairPTR !
4     (+ FRAMEGRABBER GrabTimeout @ NOT IF +)
5     FindLeft
6     (+ FRAMEGRABBER THEN +)
7     PairPTR @ 4/ #DataPairs C! ;
8 : @RightLaneData
9     0 PairPTR !
10    (+ FRAMEGRABBER GrabTimeout @ NOT IF +)
11    FindRight
12    (+ FRAMEGRABBER THEN +)
13    PairPTR @ 4/ #DataPairs C! ;
14 -]
15

```

998 Part 4 Rel 38 G:\LMS\APP\CAPC.SRC

```

0 ( GetLaneData      glg    16 Feb 1995)
1 VARIABLE [RightLaneType] VARIABLE [LeftLaneType]
2 [- FAKEDATA
3 : GetLaneData      0 ldrPTR ! 0 #LaneRecords C!
4     FrameCount C@ LgFrameCount C!
5     [RightLaneType] @ LaneType C! 0 LaneID C!
6 ( -RTS )          @RightLaneData
7 PairPTR @ ?DUP IF [LDRheader] + ldrPTR +!
8                 1 #LaneRecords C+! THEN
9     [LeftLaneType] @ LaneType C! 1 LaneID C!
10 ( -RTS )          @LeftLaneData
11 PairPTR @ ?DUP IF [LDRheader] + ldrPTR +!
12                 1 #LaneRecords C+! THEN
13 ;
14 ]
15

```

999 Part 4 - Rel 39 G:\LMS\APP\CAPC.SRC

```
0 ( Process Lane Geometry      glg   16 Feb 1995)
1 HEX
2 \ : ProcessLG ;
3
4 : ProcessLG
5         GetLaneData
6 ( -RTS )
7     ldrPTR @ |LgHeader| + #Buffer LgLen ! ;
8
9
10
11
12
13
14
15
```

1000 Part 4 Rel 40 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1001 Part 4 Rel 41 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1002 Part 4 Rel 42 G:\LMS\APP\CAPC.SRC

```
0 ( Search Geometry Reset      glg      10 Aug 1995)
1
2 VARIABLE {NearTracking}
3
4 : RG 0 tanHeading ! LaneWidth @ 2/ LatDev ! ;
5
6 : ?NearTracking {NearTracking} @ ?DUP IF 1-
7     {NearTracking} !
8     ELSE RG THEN ;
9
10
11
12
13
14
15
```

1003 Part 4 Rel 43 G:\LMS\APP\CAPC.SRC

```
0 ( Search Geometry Reset      glg      10 Aug 1995)
1
2 VARIABLE {FarTracking}
3
4
5 : ?FarTracking {FarTracking} @ ?DUP IF 1- {FarTracking} !
6     ELSE ZERO_Deltas THEN ;
7
8 : ?Tracking ?NearTracking ?FarTracking ;
9
10
11
12
13
14
15
```

1004 Part 4 Rel 44 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1005 Part 4 Rel 45 G:\LMS\APP\CAPC.SRC

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1006 Part 4 Rel 46 G:\LMS\APP\CAPC.SRC

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1007 Part 4 Rel 47 G:\LMS\APP\CAPC.SRC

```
0 ( LMS Initialization 17 Feb 1995)
1
2 : InitLMS -Listen -FrameClock
3     PacketBuffer 400 ERASE
4     P&Rdata |P&Rdata| ERASE
5     0 crcError ! 0 BadPacket !
6     LMSstatus |LMSstatus| ERASE
7     LgInit ( Assumes FrameCounter zeroed )
8     COM2 MSR INPUT DROP OBUF
9     [+ FAKEDATA 0 >PTR ! +]
10    [+ CAMERA 1600 thSEC +]
11    'start IS SetupLMS
12    'testProcess IS ?Tracking ;
13
14 0 INITIALIZES InitLMS
15
```


1008 Part 4 Rel 48 G:\LMS\APP\CAPC.SRC

```
0 ( LMS INIT 17 Feb 1995)
1
2 01 :UCMD initialize InitLMS 1 ?Initialized C! ;
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1009 Part 4 Rel 49 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1010 Part 4 Rel 50 G:\LMS\APP\CAPC.SRC

```
0 ( Sender TERMINAL task glg 09 Mar 1995)
1 ( per dev n )
2 \ ' BIOS DEVICE # TASK-SIZE TERMINAL Sender Sender CONSTRUCT
3 HEX
4 : (SendLG)
5 91 #Buffer 1+ DUP LgBuffer SWAP LgLen # SendPacket ;
6
7 \ : SendLG Sender ACTIVATE (SendLG) STOP ; RECOVER
8
9 : SendLG (SendLG) ;
10
11
12
13
14
15
```

1011 Part 4 Rel 51 G:\LMS\APP\CAPC.SRC

```

0 ( Main TERMINAL task glg          09 Mar 1995)
1 ( per dev n )
2 ' SVGA-GRAPHICS
3 DEVICE @ TASK-SIZE TERMINAL Main  Main CONSTRUCT
4
5 'TYPE @ Main 'TYPE HIS !
6
7
8
9
10
11
12
13
14
15

```

1012 Part 4 Rel 52 G:\LMS\APP\CAPC.SRC

```

0 ( BusyLoop glg 15 Feb 1995)
1
2 VARIABLE 'testProcess VARIABLE 'LMSprocess VARIABLE 'LMSsend
3
4 : BusyLoop Listen
5
6 ?FrameClock IF [+ FRAMEGRABBER ?Timeout +]
7     NewBuffer 'LMSsend @EXECUTE
8
9     [+ DIAGNOSTICS @IMAGE IMAGE 'testProcess @EXECUTE +]
10
11     'LMSprocess @EXECUTE ( .FrameCount ) 0 Armed ! THEN
12
13 ?Packet ?DUP IF DisplayPacket ExecuteCmd THEN ;
14
15

```

1013 Part 4 Rel 53 G:\LMS\APP\CAPC.SRC

```

0 ( MainLoop glg 15 Feb 1995)
1
2 : MainLoop UART FREE OBUF -RTS
3 ?Half IF HalfScale 0 50 [Corner] 2C!
4     ELSE FullScale 31 55 [Corner] 2C! THEN
5 BEGIN BusyLoop
6 PAUSE ( ?ESC ) AGAIN ;
7
8 : GO PAGE 15 0 TAB Main ACTIVATE MainLoop STOP ;
9
10 : -GO Main HALT ;
11
12
13
14
15

```

1014 Part 4 Rel 54 G:\LMS\APP\CAPC.SRC

```

0 ( CollectLoop glg 15 Feb 1995)
1
2
3 : CollectLoop UART FREE OBUF -RTS
4 ?Half IF HalfScale ELSE FullScale THEN
5 ZERO-FRAME-COUNT 0 'LMSprocess !
6 BEGIN BusyLoop
7 PAUSE [img#] @ 78 > UNTIL ;
8
9 : ZC CollectLoop ;
10
11 : SS Save-sequence ;
12
13
14
15

```

1015 Part 4 Rel 55 G:\LMS\APP\CAPC.SRC

```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```

1016 Part 4 Rel 56 G:\LMS\APP\CAPC.SRC

```

0 ( UEXPECT, UTYPE glg 15 Feb 1995)
1 ASSEMBLER
2 CREATE execute W W OR 0= NOT IF CELL # W SUB W ) LIP
3 THEN NEXT
4
5 CODE UTYPE ( a n) S ) 1 MOV 1NZ IF ( 1 C# U) ADD ) SECOND
6 'TYPE HIS W MOV execute JMP THEN 2 CELLS # S ADD NEXT
7 : uEMIT ( c) 'S 1 UTYPE DROP ;
8 \\
9 CODE UEXPECT ( a n) 0 , HERE USE CELL W) PUSH
10 SECOND 'EXPECT HIS W MOV execute JMP
11 CODE USTRAIGHT ( a n) -1 , ' UEXPECT CELL+ USE
12
13 : u?KEY ( - t) 0 0 UEXPECT ;
14 : uKEY ( - n) 0 'S 1 USTRAIGHT ;
15

```

1020 Part 4. Rel 60 G:\LMS\APP\CAPC.SRC

0 (02 May 1995)
1 MARKER PULNIX
2
3 (Vector Arithmetic) FORTH 51 IN MATH 52 IN MATH THRU
4
5 63 +P 75 +P THRU
6
7
8
9
10
11
12
13
14
15

1021 Part 4 Rel 61 G:\LMS\APP\CAPC.SRC

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1022 Part 4 Rel 62 G:\LMS\APP\CAPC.SRC

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1023 Part 4 Rel 63 G:\LMS\APP\CAPC.SRC

```

0 ( Camera Data Structure   glg      02 May 1995)           efl unit is microns
1 VARIABLE 'CamData           FPA dimension units are microns
2                               Angle units are microradians
3 : camItem ( n ) OVER CONSTANT + DOES> @ 'CamData @ + ;
4 0                               Values through FPA_Row_0 are input quantities, others are
5 CELL camItem efl                               derived quantities.
6 CELL camItem FPA_H          CELL camItem FPA_W
7 CELL camItem FPA_H#        CELL camItem FPA_W#           H_0 is the camera height (cm) above the road surface.
8 CELL camItem FPA_Row_0     CELL camItem FPA_Col_0
9 CELL camItem Phi/Pixel     CELL camItem Theta/Pixel      X_0 and Y_0 are camera coordinates (cm) in the vehicle
10 CELL camItem FPA_Addr_0                                         (CG) coordinate system. The camera is in front of, and
11 CELL camItem ALPHA_0      CELL camItem DELTA_0           to the right of the CG.
12 CELL camItem H_0          CELL camItem Y_0
13 CELL camItem X_0
14 CONSTANT |CamData|
15

```

1024 Part 4 Rel 64 G:\LMS\APP\CAPC.SRC

```

0 ( Camera Data   glg      02 May 1995)           TM9700 Data Sheet indicates 11.6 micron x 13.6 micron cell size.
1
2 CREATE Cam_Data_1           A 13.64 micron pitch follows from a 6.60 mm vertical dimension
3     12500 ,                 and 484 cells.
4     6600 , 8900 ,
5     484 , 768 ,           An 11.6 micron pitch would require a horizontal size of 8.91 mm.
6     242 , 384 ,
7     0 , 0 ,
8     0 ,
9     0 , 10751 ,
10    1245 , 15 ,
11    14 ,
12
13
14
15

```

1025 Part 4 Rel 65 G:\LMS\APP\CAPC.SRC

```

0 ( Camera Data   glg      02 May 1995)
1
2 : Camera_1 'CamData IS Cam_Data_1 ;
3
4 : CamDataInit
5     FPA_Row_0 = FPA_W# @ * FPA_Col_0 @ + FPA_Addr_0 !
6     FPA_H = 1000000 efl @ FPA_H# @ * */ Phi/Pixel !
7     FPA_W = 1000000 efl @ FPA_W# @ * */ Theta/Pixel ! ;
8
9 : CamInit Camera_1 CamDataInit ;
10 0 INITIALIZES CamInit
11
12
13
14
15

```

1026 Part 4 Rel 66 G:\LMS\APP\CAPC.SRC

```
0 ( Camera r,c to image offset conversion glg 24 Apr 1995) The image address is confined to one MB for the MuTech FrameGrab
1 HEX
2 VARIABLE '>IA
3 : rc>ImgOffset '>IA @EXECUTE ;
4
5 : rc>IA ( r c - a ) >R NEGATE FPA_W# @ *
6           R> + FPA_Addr_0 @ +
7           0 MAX 100000 MIN ;
8
9 : -rc>IA ( r c - a ) >R FPA_W# @ *
10          R> NEGATE + FPA_Addr_0 @ +
11          0 MAX 100000 MIN ;
12
13
14
15
```

1027 Part 4 Rel 67 G:\LMS\APP\CAPC.SRC

```
0 ( Pixel:Angle Conversion glg 02 May 1995) PHI and THETA units are microradians.
1
2 : PHI ( r - n ) Phi/Pixel @ 10 */ ; Pixel row and column units are tenths of a pixel.
3
4 : THETA ( c - n ) Theta/Pixel @ 10 */ ;
5
6 : ROW ( n - r ) 10 Phi/Pixel @ */ ;
7
8 : COL ( n - r ) 10 Theta/Pixel @ */ ;
9
10
11
12
13
14
15
```

1028 Part 4 Rel 68 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1029 Part 4 Rel 69 G:\LMS\APP\CAPC.SRC

```

0 ( Trig Approximations      glg      10 May 1995)           Scaling for Trig functions is 1E6.
1 MARKER -TRIG
2
3 : tan ( n - n ) DUP DUP 3000000 */ OVER 1000000 */ + ;
4
5 : sin ( n - n ) DUP DUP 6000000 */ OVER 1000000 */ - ;
6
7 : cos ( n - n ) 1000000 SWAP DUP 2000000 */ - ;
8
9 : atan ( n - n ) DUP DUP 3000000 */ OVER 1000000 */ - ;
10
11
12
13
14
15

```

1030 Part 4 Rel 70 G:\LMS\APP\CAPC.SRC

```

0 ( Roll Compensation      glg      12 May 1995)           Row and Column are adjusted by the roll value received
1                                           from the CAPC data acquisition system.
2 : Rollcomp ( r c - r' c' ) 2DUP
3     @Roll>Rad DUP sin >R cos 1000000 V*/
4     2SWAP R> 1000000 V*/ NEGATE SWAP V+ ;
5
6 : -Rollcomp ( r c - r' c' ) 2DUP
7     @Roll>Rad NEGATE DUP sin >R cos 1000000 V*/
8     2SWAP R> 1000000 V*/ NEGATE SWAP V+ ;
9
10
11
12
13
14
15

```

1031 Part 4 Rel 71 G:\LMS\APP\CAPC.SRC

```

0                                           Here, row and column units are pixels.
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```

1032 Part 4 . Rel 72 G:\LMS\APP\CAPC.SRC

```

0 ( Angle to Surface Range Conversion      glg      11 May 1995)  Surface range and cross range units are cm.
1
2 : DownRange ( phi - n ) DELTA_0 @ @Pitch>Rad - SWAP- tan  Depression angle is adjusted by the pitch delta received from
3   DUP 0> IF >R H_0 @ 100000 R> */ X_0 @ +  the CAPC data acquisition system.
4     ELSE DROP 0 THEN ;
5
6 : CrossRange ( theta phi - n ) DownRange SWAP ALPHA_0 @ -
7     1000000 */ Y_0 @ + ;
8
9 : SurfaceRange ( theta phi - cross down ) DownRange DUP >R
10    SWAP ALPHA_0 @ - 1000000 */ Y_0 @ + R> ;
11
12
13
14
15

```

1033 Part 4 Rel 73 G:\LMS\APP\CAPC.SRC

```

0 ( Camera Pixel to Surface Range Conversion  glg  18 May 1995)
1
2 : DRange ( r c - n ) Rollcomp DROP PHI DownRange ;
3
4 : CRange ( r c - n ) Rollcomp THETA SWAP PHI CrossRange ;
5
6 : SRRange ( r c - cross down )
7     Rollcomp THETA SWAP PHI SurfaceRange ;
8
9 : dc>dCross ( r c - n ) THETA SWAP PHI DownRange
10    SWAP 1000000 */ ;
11
12
13
14
15

```

1034 Part 4 Rel 74 G:\LMS\APP\CAPC.SRC

```

0 ( Surface Range to Angle Conversion      glg      23 May 1995)
1
2 : phi ( down - phi ) X_0 @ - >R H_0 @ 100000 R> */
3   atan DELTA_0 @ @Pitch>Rad - SWAP- ;
4
5 : theta ( cross down - theta )
6     SWAP Y_0 @ - SWAP
7     1000000 SWAP */
8     ALPHA_0 @ + ;
9
10 : -SR ( cross down - theta phi ) DUP >R theta R> phi ;
11
12
13
14
15

```


1035 Part 4 . Rel 75 G:\LMS\APP\CAPC.SRC

0 (Surface Range to Camera Coordinate Conversion glg 23 May 1995)

1
2 : cd>rc (cross down - r c) DUP phi >R theta
3 COL R> ROW SWAP -Rollcomp ;
4
5
6
7
8
9
10
11
12
13
14
15

1036 Part 4 Rel 76 G:\LMS\APP\CAPC.SRC

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1037 Part 4 Rel 77 G:\LMS\APP\CAPC.SRC

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

1038 Part 4 Rel 78 G:\LMS\APP\CAPC.SRC

```
0 ( Surface Range / Angle Display      glg      18 May 1995)
1
2 : '.' 46 HOLD ;
3
4 : G.2 ( n ) DUP ABS 0 <# # # '.' #S SIGN #> WRITE 2 SPCS ;
5
6
7
8
9
10
11
12
13
14
15
```

1039 Part 4 Rel 79 G:\LMS\APP\CAPC.SRC

```
0 ( Surface Range / Angle Display      glg      18 May 1995)
1 [+ DIAGNOSTICS
2
3 : .RANGES ( r c - r c ) 2DUP 10 10 V* SRange
4   @CP 2DUP 2>R 20 140 V- AT
5   NORM 18 SCRUB XOR'D 2 SPCS G.2 G.2
6   2R> AT ;
7
8 : .ANGLES ( r c - r c ) 2DUP 10 10 V* Rollcomp
9   THETA SWAP PHI @CP 2DUP 2>R 40 140 V- AT
10  NORM 18 SCRUB XOR'D 9 G.R 9 G.R
11  2R> AT ;
12
13 : .GEOMETRY 'userInfo ASSIGN .RANGES .ANGLES ;
14 +]
15
```

1040 Part 4 Rel 80 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1041 Part 4 . Rel 81 G:\LMS\APP\CAPC.SRC

```
0 ( Roadway Data Structure   glg    24 May 1995)           Roadway dimension units are cm
1 VARIABLE 'RoadData
2
3 : roadItem ( n ) OVER CONSTANT + DOES> @ 'RoadData @ + ;
4 0
5 CELL roadItem tanHeading
6 CELL roadItem LatDev      CELL roadItem LaneWidth
7 CELL roadItem SurfaceVals
8
9 CONSTANT |RoadData|
10
11
12 : <Y> ( down - cross ) tanHeading @ 1000000 */
13           LatDev @ + ;
14
15
```

1042 Part 4 Rel 82 G:\LMS\APP\CAPC.SRC

```
0 ( LaneMark Data Structure   glg    24 May 1995)           Lane mark dimension units are cm.
1 VARIABLE 'LMdata
2
3 : lmItem ( n ) OVER CONSTANT + DOES> @ 'LMdata @ + ;
4 0
5 CELL lmItem LMwidth
6 CELL lmItem LMlen
7 CELL lmItem LMval
8
9 CONSTANT |LMdata|
10
11
12
13
14
15
```

1043 Part 4 Rel 83 G:\LMS\APP\CAPC.SRC

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

1044 Part 4 . Rel 84 G:\LMS\APP\CAPC.SRC

```

0 ( Roadway Data      glg      24 May 1995)      Ford TEST track width is 3.50 m
1
2 CREATE RoadData |RoadData| ALLOT
3
4 : RoadInit RoadData |RoadData| ERASE
5         'RoadData IS RoadData ;
6
7 0 INITIALIZES RoadInit
8
9
10
11
12
13
14
15

```

1045 Part 4 Rel 85 G:\LMS\APP\CAPC.SRC

```

0 ( Lane Mark Data      glg      24 May 1995)
1
2 CREATE LMdata |LMdata| ALLOT
3
4 : LMdataInit LMdata |LMdata| ERASE
5         'LMdata IS LMdata
6         10 LMwidth ! ;
7
8 0 INITIALIZES LMdataInit
9
10
11
12
13
14
15

```

1046 Part 4 Rel 86 G:\LMS\APP\CAPC.SRC

```

0 ( Lane Mark Intensity glg  15 Jun 1995)
1
2 VARIABLE S_I#  VARIABLE S_I
3
4 : CLEAR_I 0 S_I# ! 0 S_I ! ;
5
6 : I+ ( n ) 1 S_I# +! S_I +! ;
7
8 : <I> ( n ) S_I# 4 ?DUP IF DUP 2/ S_I * + SWAP /
9         ELSE 0 THEN ;
10
11
12
13
14
15

```

1047 Part 4 . Rel 87 G:\LMS\APP\CAPC.SRC

```

0 ( Least Squares Line Data Structure   glg   04 Jun 1995)
1 VARIABLE 'LSQdata
2
3 : lsqItem ( n ) OVER CONSTANT + DOES> @ 'LSQdata @ + ;
4 0
5 CELL lsqItem S_#
6 CELL lsqItem S_X
7 CELL lsqItem S_Y
8 CELL lsqItem S_X2
9 CELL lsqItem S_Y2
10 CELL lsqItem S_XY
11
12 CONSTANT |LSQdata|
13
14
15

```

1048 Part 4 Rel 88 G:\LMS\APP\CAPC.SRC

```

0 ( Least Squares Line   glg   04 Jun 1995)
1 : CLEAR_S S_# |LSQdata| ERASE ;
2
3 : S+ ( y x - ) 2DUP S_X V+! 2DUP 2DUP V* S_X2 V+!
4           * S_XY +! 1 S_# +! ;
5
6 : denom S_# @ S_X2 @ * S_X @ DUP * - ;
7 : slope ( n ) S_# @ S_XY @ * S_X @ S_Y @ * -
8           1000000 denom */ ;
9
10 : intercept ( n ) S_X2 @ S_Y @ denom */
11       S_X @ S_XY @ denom */ - ;
12
13 : LinReg ( - slope intercept t | f ) S_# * 3 > IF slope
14       intercept 1 ELSE 0 THEN ;
15

```

1049 Part 4 Rel 89 G:\LMS\APP\CAPC.SRC

```

0 ( ?ORIENTATION   glg   04 Jun 1995)
1
2 CREATE RLWorkspace |LSQdata| ALLOT
3
4 'LSQdata IS RLWorkspace
5
6
7
8
9
10
11
12
13
14
15

```

1050 Part 4. Rel 90 G:\LMS\APP\CAPC.SRC

```

0 ( Line Following Stats      glg      13 Jun 1995)
1 VARIABLE 'DeltaData
2
3 : deltaItem ( n ) OVER CONSTANT + DOES> @ 'DeltaData @ + ;
4
5 0 CELL deltaItem S_D#
6 CELL deltaItem S_Delta
7 CELL deltaItem S_|Delta|
8 CELL deltaItem <Delta>
9 8 CELLS deltaItem RDeltas
10 8 CELLS deltaItem LDeltas
11
12 CONSTANT |DeltaData|
13
14
15

```

1051 Part 4 Rel 91 G:\LMS\APP\CAPC.SRC

```

0 ( Line Following Stats      glg      13 Jun 1995)
1
2 : ZERO_Deltas S_D# |DeltaData| ERASE ;
3
4 : Init_Delta_Sums S_D# 3 CELLS ERASE ;
5
6 : +!Delta ( n ) 1 S_D# +! DUP S_Delta +! ABS S_|Delta| +! ;
7
8 : UpdateRDeltas ( n n ) RDeltas + 2DUP H+!
9           2+ H@ + <Delta> +! ;
10
11 CREATE DeltaWorkSpace |DeltaData| ALLOT
12
13 'DeltaData IS DeltaWorkSpace
14 43 +P LOAD
15

```

1052 Part 4 Rel 92 G:\LMS\APP\CAPC.SRC

```

0 ( Gradient      glg      11 May 1995) HEX
1 CODE GRAD ( o d n ) 1 POP 40 # 1 CMP S> IF 40 # 1 MOV THEN
2 W POP S ) I XCHG R PUSH U PUSH PUSHF STD
3 DS 0 MOV { - IMGSEQ >MuVRAM 0 MOV - } 0 GS MOV
4 3 3 XOR 2 2 XOR 0 0 XOR R R XOR
5 1 b W ) MOV 1 W ADD 4 CELLS W) W LEA 1 I ADD I DEC
6 GS SEG I ) 0 hi MOV
7 BEGIN GS SEG b LODS h 0 8 # ROL 0 hi 0 SUB
8 CY IF 0 b NEG 0 b 3 CMP U< IF 0 b 3 MOV 1 b 3 hi MOV
9 THEN STC
10 ELSE 3 10 # ROR 0 b 3 CMP U< IF 0 b 3 MOV
11 1 b 3 hi MOV THEN 3 10 # ROR CLC
12 THEN R RCR 2 RCR b STOS
13 LOOP b LODS b STOS -3 W) W LEA
14 R 0 MOV STOS 2 0 MOV STOS 3 0 MOV STOS
15 3 I XCHG POPF U POP R POP S ) I XCHG NEXT

```

1053 Part 4 . Rel 93 G:\LMS\APP\CAPC.SRC

```

0 ( Line Following      glg      11 May 1995)                [ColDelta] is the pixel delta between the estimated line locatio
1 HEX                                                         n      and the measured line location
2 VARIABLE [ColDelta]
3
4 CODE LocMark ( r c n n - r c' height width )
5     W POP
6     2 POP  2 1 MOV  1 10 # ROR  b 1 0 1 MOVZX  b 2 0 ADD
7     0 # 0 hi ADC  h 0 SHR  0 PUSH
8     2 hi 0 1 MOVZX  1 hi 0 SUB  0 # 0 hi SBB  SXT  0 PUSH
9     2 hi 0 1 MOVZX  1 hi 0 ADD  0 # 0 hi ADC  0 SHR
10    (+ INVERTED  0 2 CELLS S) SUB  +)
11    (- INVERTED  0 2 CELLS S) ADD  -)
12    W SHR  W 0 SUB  (+ INVERTED  0 NEG  +)
13    0 [ColDelta] MOV  NEXT
14
15

```

1054 Part 4 Rel 94 G:\LMS\APP\CAPC.SRC

```

0 ( Line Following      glg      11 May 1995)
1 VARIABLE (Slope)  VARIABLE Collimit
2 : EdgeLimit ( r c n - r c' n ) >R
3     DUP 0< IF ABS -1 ELSE 1 THEN SWAP
4     Collimit @ R@ 2/ - MIN * R> ;
5
6 : ?LaneMark ( r c n - r c' f )
7     >R 0 R@ 2/
8     [- INVERTED  V- -] [+ INVERTED  V+ +]
9     2DUP rc>ImgOffset
10    [+ IMGSEQ imgAddr + +] [- IMGSEQ >vram CELL+ # + -]
11    PAD R> GRAD PAD C@ LocMark 2>R OVER R>
12    10 10 V* dc>dCross LMwidth @ 2* <
13    R> [Slope] # > AND IF 1 ELSE 0 THEN ;
14
15

```

1055 Part 4 Rel 95 G:\LMS\APP\CAPC.SRC

```

0 ( Lane Processing Marks  24 May 1995)
1 [+ DIAGNOSTICS
2
3 : LMdataMark
4     ( r c n ) >R ( 1 10 V*/ ) R@ 2/ 0 V+ rc>*RC
5     @CP 2SWAP 2DUP *AT R> 0 V- YELLOW *DRAW AT ;
6
7 : SearchMark ( r c n ) >R 0 R@ 2/ V- rc>*RC
8     @CP 2SWAP 2DUP *AT 0 R> V+ RED *DRAW AT ;
9
10 : LaneMark BLUE
11 600 DUP <Y> SWAP cd>rc 1 10 V*/ rc>*RC @CP 2>R *AT
12 2000 DUP <Y> SWAP cd>rc 1 10 V*/ rc>*RC
13 *DRAW 2R> AT ; +)
14
15

```

1056 Part 4. Rel 96 G:\LMS\APP\CAPC.SRC

```

0 ( LM Locate      glg      24 May 1995)
1 : FindNearRight  CLEAR_S  CLEAR_I
2 2200 600 DO I DUP <Y> SWAP
3      cd>rc
4      1 10 V*/ 64 EdgeLimit
5      [+ DIAGNOSTICS >R 2DUP R@ SearchMark R> +]
6      ?LaneMark
7      NOT IF NIP -242 SWAP ELSE
8      2DUP @ImgVal I+ 2DUP
9      10 10 V* SRange 2DUP S+ !Pair THEN
10     [+ DIAGNOSTICS 16 LMdataMark +]
11     [- DIAGNOSTICS 2DROP -]
12     200 +LOOP
13     LinReg IF LatDev ! tanHeading ! 10 [NearTracking] !
14     [+ DIAGNOSTICS LaneMark +] THEN ;
15

```

1057 Part 4 Rel 97 G:\LMS\APP\CAPC.SRC

```

0 ( LM Locate      glg      24 May 1995)
1
2 : FindNearLeft
3 2200 600 DO I <Y> LaneWidth @ - I
4      cd>rc
5      1 10 V*/ 64 EdgeLimit
6      [+ DIAGNOSTICS >R 2DUP R@ SearchMark R> +]
7      ?LaneMark
8      NOT IF NIP -242 SWAP ELSE 2DUP
9      10 10 V* SRange !Pair THEN
10     [+ DIAGNOSTICS 16 LMdataMark +]
11     [- DIAGNOSTICS 2DROP -]
12     200 +LOOP ;
13
14
15

```

1058 Part 4 Rel 98 G:\LMS\APP\CAPC.SRC

```

0 ( LM Locate      glg      24 May 1995)
1
2 : FindMark ( cross down n - f ) OVER >R >R
3      cd>rc
4      1 10 V*/ <Delta> @ + R> EdgeLimit
5      [+ DIAGNOSTICS >R 2DUP R@ SearchMark R> +]
6      ?LaneMark
7      NOT IF R> DROP NIP 0 SWAP -242 SWAP
8      ELSE 1 ROT ROT 2DUP
9      10 10 V* SRange
10     R> 100 / 1 AND NOT IF !Pair ELSE 2DROP THEN THEN
11     [- DIAGNOSTICS 2DROP -]
12     [+ DIAGNOSTICS 8 LMdataMark +] ;
13
14
15

```

Only even-valued downrange distance pairs are sent to CAPC
QUADRA.

1059 Part 4. Rel 99 G:\LMS\APP\CAPC.SRC

```

0 ( LM Locate      glg      24 May 1995)
1
2 : |Mark| ( - n ) S_D# @ 5 OVER < IF
3           S_|Delta| @ 10 ROT */ 4 MAX 16 MIN
4           ELSE DROP 16 THEN ;
5
6
7
8
9
10
11
12
13
14
15

```

The Deltas array must be large enough to hold all the values generated by the DO loop.

1060 Part 4 Rel 100 G:\LMS\APP\CAPC.SRC

```

0 ( LM Locate      glg      24 May 1995)
1 VARIABLE {#Rhits}
2 : FindFarRight  Init_Delta_Sums  RDeltas H# <Delta> !
3           0 {#Rhits} !
4           0 10500 2500 DO
5           I DUP <Y> SWAP |Mark| FindMark
6 IF {ColDelta} @ 1 {#Rhits} +!
7 ELSE |Mark| 2/ S_|Delta| +! 0
8 THEN
9 DUP +!Delta |Mark| MIN OVER UpdateRDeltas
10      2+ 500 +LOOP DROP
11 {#Rhits} @ 4 > IF 10 {FarTracking} ! THEN ;
12
13 : FindRight  FindNearRight  FindFarRight ;
14
15

```

1061 Part 4 Rel 101 G:\LMS\APP\CAPC.SRC

```

0 ( LM Locate      glg      24 May 1995)
1
2 : FindFarLeft  Init_Delta_Sums  LDeltas H# <Delta> !
3           0 10500 2500 DO I <Y> LaneWidth @ - I
4           |Mark| FindMark
5 IF {ColDelta} @
6 ELSE |Mark| 2/ S_|Delta| +! 0
7 THEN
8 DUP +!Delta OVER LDeltas + 2+ H# + <Delta> +!
9           500 +LOOP DROP ;
10
11 : FindLeft  FindNearLeft  FindFarLeft ;
12
13
14
15

```

1062 Part 4 - Rel 102 G:\LMS\APP\CAPC.SRC

```

0 ( VIEWING SUPPORT glg 06 Apr 1995)
1
2 1024 1024 * 29 * MALLOC ( Grab -80 frames of memory )
3
4 VIEWING LOAD
5 484 [imgH] ! 768 [imgW] ! 1 [B/P] ! imgInit
6
7 78 +P 79 +P THRU
8
9 : VAN 1588 H_0 ! 157552 DELTA_0 ! 20394 ALPHA_0 ! ;
10 : VAN2 1588 H_0 ! 94830 DELTA_0 ! 23175 ALPHA_0 ! ;
11 : TAURUS2 1241 H_0 ! 10751 DELTA_0 ! -3821 ALPHA_0 ! ; For TAURUS2: Pitch= -13 Roll=+16
12 : TAURUS 1241 H_0 ! 14000 DELTA_0 ! -2781 ALPHA_0 ! ; DELTA_0 = 20710 ( HIGHBAY ) 10751 ( TRACK )
13 0 INITIALIZES TAURUS2 ALPHA_0 = -2781 ( HIGHBAY ) -3821 ( TRACK )
14 \ IMAGE= E:\CAPC\IMAGES\DTB6.RAW
15

```

1063 Part 4 Rel 103 G:\LMS\APP\CAPC.SRC

```

0 ( @ImgVal glg 24 Apr 1995)
1
2 : @ImgVal ( r c - n ) rc>ImgOffset
3 [+ IMGSEQ imgAddr + IO 1- MIN C@ +]
4 [- IMGSEQ >R >MuVRAM @ >vram CELL+ @ R> +
5 EC@ -] ;
6
7 : FullScale 483 0 ORIGIN 599 799 SIZE ;
8 : HalfScale 241 0 ORIGIN 299 399 SIZE ;
9
10 [- IMGSEQ
11 : Direct 'imageDisplay ASSIGN DROP
12 >vram @ 0 'host>display @EXECUTE ; -]
13
14
15

```

1064 Part 4 Rel 104 G:\LMS\APP\CAPC.SRC

```

0 ( Display Row & Column / Camera Conversion 16 May 1995)
1
2 : RC>rc ( r c - r' c' )
3 NEGATE FPA_Row_0 @ NEGATE FPA_Col_0 @ V+ ;
4
5 : RC>rc ( r c - r' c' )
6 YNEGATE FPA_Row_0 @ FPA_Col_0 @ NEGATE V+ ;
7
8 : -rc>RC ( r c - r' c' )
9 NEGATE FPA_Row_0 @ FPA_Col_0 @ V+ ;
10
11 : rc>RC ( r c - r' c' )
12 YNEGATE FPA_Row_0 @ FPA_Col_0 @ V+ ;
13 : rc>*RC ( r c - r' c' )
14 FPA_Row_0 @ FPA_Col_0 @ V+ ;
15

```

Appendix F

"Road Friction Coefficient Estimation for Vehicle Path Prediction"

C.-S. Liu and H. Peng,
1995 LAVSD Symposium on Vehicles on Roads and Tracks

1065 Part 4 - Rel 105 G:\LMS\APP\CAPC.SRC

```

0 ( DispInit          glg          24 Apr 1995)
1
2 : DispInit B&W 'imageType IS sequential
3           0 [delay] ! HalfSize HalfScale
4 [+ IMGSEQ Plain +] [- IMGSEQ Direct -]
5 [+ INVERTED
6           'host>display IS /2BROT180
7           '>IA IS -rc>IA
8           'rcXform IS RC>rc +)
9 [- INVERTED
10          '>IA IS rc>IA
11          'rcXform IS RC>rc -]
12          '@ImageData IS @ImgVal ;
13
14 0 INITIALIZES DispInit
15

```

1066 Part 4 Rel 106 G:\LMS\APP\CAPC.SRC

```

0 ( Image Fetch      glg          07 Apr 1995)
1
2 [- CAMERA
3
4 : @IMAGE [+ IMGSEQ 1 [img#] +!
5
6 [+ CAPC_COM " P&RData " COUNT DROP 2@
7           imgAddr 2@ D- + NOT IF
8           imgAddr 9 + P&Rcount 5 MOVE ( .P&R ) THEN +) .
9
10          +] ; -]
11
12
13
14
15

```

1067 Part 4 Rel 107 G:\LMS\APP\CAPC.SRC

```

0 ( Image Fetch      glg          07 Apr 1995)
1
2 [+ CAMERA
3
4 : @IMAGE
5           [NextFrame#] @ [img#] !
6
7 [+ IMGSEQ >vram CELL+ @
8           imgAddr imgLen @image
9
10 [+ CAPC_COM " P&RData " COUNT imgAddr SWAP MOVE
11           FrameCount C@ imgAddr 8 + C!
12           P&Rcount imgAddr 9 + 5 MOVE +) +]
13
14           1 [NextFrame#] +! ; +)
15

```

1068 Part 4 - Rel 108 G:\LMS\APP\CAPC.SRC

```

0 ( Image Capture      glg      25 Mar 1994)
1 [+ CAMERA
2 : CAPTURE capture @IMAGE IMAGE ;
3
4 : [CAPTURE [capture @IMAGE IMAGE ;
5
6 : CAPTURES ( n - ) 0 MAX ?DUP IF capture
7 1- ?DUP IF 0 DO [CAPTURE [NextFrame#] @ . capture] ' LOOP
8     THEN @IMAGE IMAGE [NextFrame#] @ . THEN ;
9
10 : CAPTURING BEGIN [CAPTURE capture] ?ESC AGAIN ;
11
12 'vt IS CAPTURE
13 +)
14
15

```

1069 Part 4 Rel 109 G:\LMS\APP\CAPC.SRC

```

0 ( IMAGE STUFFING glg      23 Mar 1994) HEX
1 CODE MBSTUFIT ( s o - ) 0Banks CALL *A0 0 MOV 0 ES MOV
2     W POP S ) I XCHG >MuVRAM 0 MOV 0 GS MOV
3     b Bank 0 MOVZX 0 10 # ROL 0 W SUB
4 [winH] 1 MOV PUSHF CLD R PUSH Lookup R MOV
5 BEGIN 1 PUSH [winW] 1 MOV
6 BEGIN GS SEG b LODS [bias] 0 ADD
7     FF # 0 AND 7000 # 0 h TEST
8     0= NOT IF 0 h OD # SHR 1000 # 0 h OR THEN
9     0 R) 0 )l b 0 MOV b STOS
10 LOOP
11 PHYSICAL W ADD W INC [winW] W SUB
12 [imgW] I ADD [imgW] I ADD [winW] I SUB [winW] I SUB
13 1 POP LOOP
14 R POP POPF I POP SS_DATA # 0 MOV 0 DS MOV 0 ES MOV NEXT
15

```

1070 Part 4 Rel 110 G:\LMS\APP\CAPC.SRC

```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```

1071 Part 4 - Rel 111 G:\LMS\APP\CAPC.SRC

```

0 ( LMS Startup Sequence      glg      09 May 1995)
1
2 [+ CAMERA
3
4 : SetRoad  RoadVal  80  setexp  ;
5
6 : SetLine
7     FindNearRight
8     S_# @ 3 > IF
9         <I> ?DUP IF [DesiredVal] @
10    2DUP - ABS 10 < IF 2DROP EXIT THEN
11    setexp THEN THEN ;
12 +)
13
14
15

```

1072 Part 4 Rel 112 G:\LMS\APP\CAPC.SRC

```

0 ( LMS Startup Sequence      glg      09 May 1995)
1 VARIABLE [AutoExp]
2
3 : SetExposure [+ CAMERA      [AutoExp] @ IF
4   FrameCount C@ DUP 1 AND IF DROP EXIT THEN
5       DUP 4 < IF DROP EXIT THEN
6       DUP 10 < IF DROP SetRoad EXIT THEN
7       20 < IF SetLine EXIT THEN
8
9 THEN +) 'LMSprocess IS ProcessLG 'LMSsend IS SendLG ;
10
11 : SetupLMS  RG [+ CAMERA 1600 thSEC +]
12     'LMSprocess IS SetExposure 0 'LMSsend ! ;
13
14
15

```

1073 Part 4 Rel 113 G:\LMS\APP\CAPC.SRC

```

0 ( LMS Startup Sequence      glg      09 May 1995)
1
2 : InitCAPC 15 [Slope] ! 364 ColLimit ! COM2 57600 BAUD
3   366 LaneWidth ! RG
4   ZERO_Deltas 0 [RightLaneType] ! 1 [LeftLaneType] !
5   1 [AutoExp] !
6 'start IS SetupLMS
7 [+ DIAGNOSTICS [+ CAPC_COM 'imageLabel IS .LMSdata
8   0 50 [Corner] 2C! +) +) ;
9
10 0 INITIALIZES InitCAPC
11
12 : DEARBORN 350 LaneWidth ! 0 [LeftLaneType] ! ;
13 : UMTRI   311 LaneWidth ! 1 [LeftLaneType] ! ;
14 : HIGHWAY 366 LaneWidth ! 0 [LeftLaneType] ! ;
15

```

1074 Part 4- Rel 114 G:\LMS\APP\CAPC.SRC

```

0 ( Auto Exposure      glg      24 Apr 1995)
1
2 VARIABLE 'CurrentVal VARIABLE %Max VARIABLE {DesiredVal}
3
4 : CurrentVal ( n ) 'CurrentVal @EXECUTE ;
5 : CenterVal ( n ) 0 0 @ImgVal ;
6 : RoadVal ( n ) -100 0 @ImgVal ;
7
8 : NewExp ( n n - n ) >R DUP 255 < IF
9           Exposure @ R> */
10          ELSE R> DROP DROP Exposure @ 2* THEN ;
11
12 : .EXPOSURE
13 550 640 AT logical @ NORM @COLOR WHITE 10 SCRUB XOR'D
14 Exposure @ 10 G.R COLOR logical ! ;
15

```

1075 Part 4 Rel 115 G:\LMS\APP\CAPC.SRC

```

0 ( Auto Exposure      glg      24 Apr 1995)
1
2 : ?Exp ( n - n ) DUP
3 2196 > IF CR ." Excessive Illumination -- reduce iris "
4         CR THEN DUP
5 500 < IF CR ." Insufficient Illumination -- enlarge iris "
6         CR THEN ;
7
8
9
10
11
12
13
14
15

```

1076 Part 4 Rel 116 G:\LMS\APP\CAPC.SRC

```

0 ( Auto Exposure      glg      24 Apr 1995)
1
2 : setexp ( n n ) NewExp ?Exp thSEC .EXPOSURE ;
3
4 : SETEXP CurrentVal {DesiredVal} @ setexp ;
5
6 : InitExp 'CurrentVal IS CenterVal 85 %Max !
7         200 {DesiredVal} ! 1600 thSEC ;
8
9 0 INITIALIZES InitExp
10
11
12 [+ DIAGNOSTICS
13 : MouseVal ( n ) MOUSER MP HIS 2* @ImgVal ; +]
14
15

```


1077 Part 4 . Rel 117 G:\LMS\APP\CAPC.SRC

```

0 ( Auto Threshold glg 27 Apr 1995)
1 VARIABLE Gain VARIABLE DeltaT
2 VARIABLE {Velocity} VARIABLE {Position} VARIABLE {Eps}
3
4 : !Eps CurrentVal 1000 * {Position} @ - {Eps} ! ;
5
6 : PVel 1000 Gain @ - DUP * DeltaT @ /
7 {Eps} @ 1000 */ {Velocity} @ + ;
8
9 : PPos 1000 Gain @ - 2* {Eps} @ 1000 */
10 DeltaT @ {Velocity} @ 1000 */ + {Position} @ + ;
11
12 : UpdateFilter !Eps PPos {Position} ! PVel {Velocity} ! ;
13
14 : Fval {Position} @ 1000 / ;
15

```

1078 Part 4 Rel 118 G:\LMS\APP\CAPC.SRC

```

0 ( Filter Length glg 27 Apr 1995)
1
2 : FilterLen >R 3200 DeltaT @ R> */ 1000 SWAP- Gain ! ;
3
4 : FilterInit CurrentVal 1000 * {Position} ! -0 {Velocity} ! ;
5
6 : .FILTER
7 500 640 AT logical @ NORM @COLOR WHITE 10 SCRUB XOR'D
8 CurrentVal 4 G.R Fval 6 G.R COLOR logical ! ;
9
10 : FMFInit
11 0 {Velocity} ! 128 1000 * {Position} !
12 'value IS Fval 125 DeltaT ! 1000 FilterLen ;
13
14 0 INITIALIZES FMFInit
15

```

1079 Part 4 Rel 119 G:\LMS\APP\CAPC.SRC

```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

```


Road Friction Coefficient Estimation For Vehicle Path Prediction

CHIA-SHANG LIU¹ AND HUEI PENG²

1. INTRODUCTION

One of the most noticeable trends of the automotive industry in the 1990's is the emerging of active safety technologies. Instead of passive protection and crash worthiness, emphasis has been shifted toward accident prevention and damage reduction using active safety techniques. To develop effective active safety devices, it is necessary to estimate vehicle motions accurately, and reliable road friction estimation is one of the most important steps toward this goal.

This paper is part of an on-going research toward the development of a lane-departure warning system [1], one of the most technology-demanding active safety systems. The basic concept of the lane-departure warning system is to project vehicle trajectory, and compare with the perceived road geometry (obtained from vision systems) to calculate a performance metric termed "time to lane crossing" (TLC). When the calculated TLC is less than a threshold value, the control system will either issue warning signals or take intervention actions. To project the vehicle future trajectory accurately, we need to update vehicle parameters (speed, cornering stiffness, etc.) and estimate external disturbances (road super-elevation, wind gust, etc.). The estimation of external disturbances further depends on the vehicle parameters. Therefore, on-line estimation of road/tire characteristics is crucial for the TLC calculation and the overall lane-departure warning system. It is important to note that accurate road/tire friction estimation can also improve the performance of many other vehicle control/safety systems such as ABS, traction control, and 4WS systems.

Recently, various methods to identify the road friction coefficients have been developed. Dieckmann [2] developed a method which allows the exact measurement of wheel-slip in the order of 10^{-4} . From the information of measured wheel slip the road surface variation is detected. Eichhorn et al. [3] use optical sensor at the front-end of the car and stress and strain sensors inside the tire's tread to determine the road friction potential. Ito et al. [4] uses the applied traction force and the resulting wheel slip variations to estimate the road surface condition. Pal et al. [5] applied the neural-network based identification technique to predict the structural response of a non-linear system and applied it to forecast the frictional coefficient of the road. Pasterkamp and Pacejka [6] develops an on-line estimation method based on recognition of the pneumatic trail to estimate the friction between tire and road. In this paper, we have developed a disturbance observer to identify the road surface friction coefficient. The effect of the road surface condition on vehicle path prediction (measured in TLC) is the main performance evaluation metric. The vehicle path prediction is obtained based on the 2-DOF lateral dynamic model (bicycle model). To calculate the TLC the longitudinal tire force is first estimated from a single wheel model. The road friction

¹ Graduate student

² Assistant Professor, University of Michigan, Ann Arbor, MI 48109

coefficient and the tire lateral force are then calculated based on an anisotropic model. Based on the estimated lateral force, the cornering stiffness is then obtained. This updated cornering stiffness is then used in the bicycle model to compute the TLC. In the estimation scheme, the wheel speed and torque are assumed to be available (measured or calculated). From the measured wheel speed and the estimated vehicle velocity the tire slip ratio can be calculated. Vehicle velocity is derived by the undriven wheel under driving conditions or estimated during braking conditions.

The methods to estimate the road surface conditions are least square method and modified adaptive observer. The least square method used in this paper is a recursive least square method with a forgetting factor [8]. The modified adaptive observer applied in this paper is a traditional adaptive observer with a correcting term which improves the performance of the observer based on a linear relationship between the output and input signals. From simulation results we found that the inclusion of the road friction estimation scheme will improve the TLC accuracy significantly under several cases.

2. SYSTEM MODELS

The system models used to estimate the road surface condition consist of two parts: a single wheel model and an anisotropic brush model. The model used for the vehicle path prediction and TLC calculations is a 2 DOF lateral dynamic model.

2.1 Single Wheel Model

The single wheel model shown in Fig. 1 is used to calculate the tire longitudinal force. The dynamics of the system in the state-space form is

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \quad (1)$$

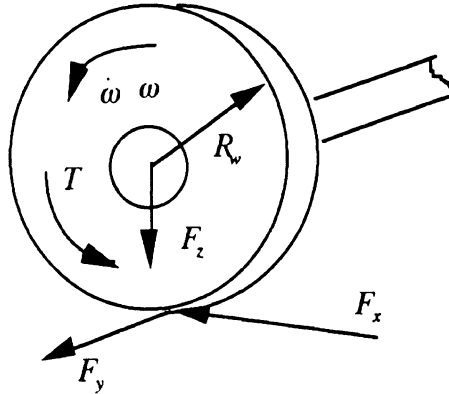


Fig. 1 Single Wheel Model

where x is the wheel angular velocity ω , y is the measurement (which is also the wheel speed), $A=0$, $B=1$ and $C=1$. The external disturbance signal u is

$$u = \frac{T}{I_w} - F_x \frac{R_w}{I_w} \quad (2)$$

where T is the tire traction torque (negative when braking). I_w is the wheel moment of inertia, F_x is the tire longitudinal force and R_w denotes the radius of the tire.

2.2 Anisotropic Brush Model

The anisotropic brush tire model represents the tire forces as functions of the tire slip ratio, slip angle and road surface condition. When the tire strain is not saturated, the longitudinal force F_x and lateral force F_y are

$$F_x = -\frac{k_x \sigma_x}{\sqrt{(k_x \sigma_x)^2 + (k_y \sigma_y)^2}} \left\{ 3c_o \sigma - 3 \frac{1}{\mu} \frac{(c_o \sigma)^2}{F_z} + \frac{1}{\mu^2} \frac{(c_o \sigma)^3}{F_z^2} \right\} \quad (3a)$$

$$F_y = -\frac{k_y \sigma_y}{\sqrt{(k_x \sigma_x)^2 + (k_y \sigma_y)^2}} \left\{ 3c_o \sigma - 3 \frac{1}{\mu} \frac{(c_o \sigma)^2}{F_z} + \frac{1}{\mu^2} \frac{(c_o \sigma)^3}{F_z^2} \right\} \quad (3b)$$

where $\sigma = \sqrt{(\sigma_x)^2 + (\sigma_y)^2}$ is the total strain of the tire. $\sigma_x = -\frac{\kappa}{1+\kappa}$ and $\sigma_y = -\frac{\tan(\alpha)}{1+\kappa}$ are the strain components in the x and y directions, respectively. k_x and k_y are the tire horizontal stiffness, α is the tire slip angle, κ is the tire slip ratio, and F_x is the tire normal force. μ is the maximum traction coefficient.

The main reason we chose the brush model is because it is a easier model in mathematical manipulations. The parameters of the brush model were tuned to match the force curves of the magic formula[7] used in the simulation program. Fig 2 compares the tire forces from both the brush model and the magic formula under different road surface conditions and tire slip ratio. It can be seen that some discrepancies still exist. However, when the slip ratio is smaller than the peak-force value, these two models are quite close.

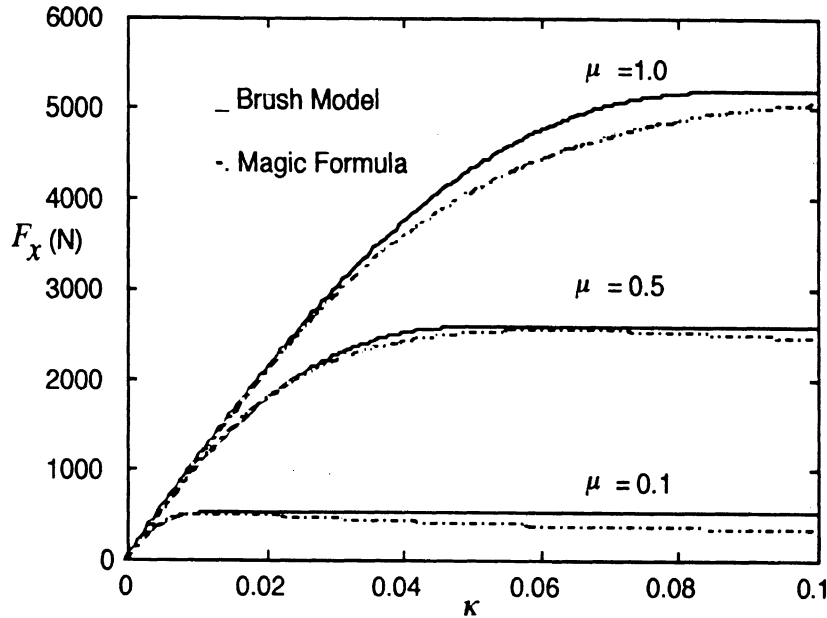


Fig.2 The Comparison between Brush Model and Magic Formula

2.3 Vehicle Lateral Dynamics

The vehicle lateral dynamics were found to be accurately described by the following 2 DOF bicycle model when the lateral acceleration is less than 0.3g

$$\begin{bmatrix} \dot{\mathbf{v}} \\ \dot{\mathbf{r}} \end{bmatrix} = \begin{bmatrix} \frac{1}{m} \left(\frac{2C_{sf} + 2C_{sr}}{\mathbf{u}} \right) & \frac{1}{m} \left(-m\mathbf{u} + \frac{2aC_{sf} - 2bC_{sr}}{\mathbf{u}} \right) \\ \frac{1}{I_z} \left(\frac{2aC_{sf} - 2bC_{sr}}{\mathbf{u}} \right) & \frac{1}{I_z} \left(\frac{2a^2C_{sf} + 2b^2C_{sr}}{\mathbf{u}} \right) \end{bmatrix} \begin{bmatrix} \mathbf{v} \\ \mathbf{r} \end{bmatrix} + \begin{bmatrix} \frac{-2C_{sf}}{m} \\ \frac{-2aC_{sf}}{I_z} \end{bmatrix} \delta_{fw} \quad (4)$$

where \mathbf{u} and \mathbf{v} are the vehicle longitudinal and lateral speeds, \mathbf{r} is the yaw rate, m is the vehicle mass, and I_z is the yaw moment of inertia. C_{sf} and C_{sr} are front and rear wheel cornering

stiffness, respectively. δ_{fw} is the front wheel steering angle, and a and b are the distance from vehicle center of gravity to front and rear axles, respectively.

3. ESTIMATION SCHEMES

3.1 Recursive Least Square Method

To estimate the longitudinal force from the single tire model, we can rewrite Eq.(1) into the discrete form as

$$\frac{\omega(k+1) - \omega(k)}{\Delta t} - \frac{T(k)}{I_w} = \frac{-F_x(k)R_w}{I_w} \quad (5)$$

where Δt is the sampling time, and k is the time index. For least-square estimation, we put the output signal in the following form

$$y(k) = \Phi^T(k)\Theta(k) \quad (6)$$

where for the single-wheel dynamics, we have $y(k) = \frac{\omega(k) - \omega(k-1)}{\Delta t} - \frac{T(k-1)}{I_w}$, $\Phi^T(k) = -\frac{R_w}{I_w}$

and $\Theta(k) = F_x(k)$

It is well-known that the recursive least-square estimation of the parameter vector $\hat{\Theta}(k)$ is

$$\hat{\Theta}(k+1) = \hat{\Theta}(k) + \Psi(k+1)\Gamma(k)\Phi(k+1)[y(k+1) - \Phi^T(k+1)\hat{\Theta}(k)]$$

and the two auxiliary matrices are updated in the following way

$$\Gamma(k+1) = \frac{1}{\lambda} [\Gamma(k) - \Psi(k+1)\Gamma(k)\Phi(k+1)\Phi^T(k+1)\Gamma(k)] \quad (7)$$

$$\Psi(k+1) = \frac{1}{[1 + \Phi^T(k+1)\Gamma(k)\Phi(k+1)]}$$

where $0 < \lambda < 1$ is the forgetting factor.

3.2 Enhanced Adaptive Observer

A standard Luenberger observer for the state-space model shown in Eq.(1) has the following form:

$$\begin{aligned} \hat{\dot{x}}(t) &= A\hat{x}(t) + B\hat{u}(t) + K(\hat{y} - y) \\ \hat{y}(t) &= C\hat{x}(t) \end{aligned} \quad (8)$$

where K is the observer gain vector selected in a way such that the eigenvalues of the closed-loop state matrix $A + KC$ are located at desired locations. The error dynamics of this observer can be obtained by subtracting Eq.(1) from Eq.(8):

$$\dot{e} = A_k e + B e_u \quad (9)$$

where $e = \hat{x} - x$, $e_u = \hat{u}(t) - u(t)$ and $A_k = A + KC$. The updating law for the estimation of the unknown input is chosen to be

$$\dot{\hat{u}}_o(t) = -B^T P e \quad (10)$$

$$\hat{u}(t) = \hat{u}_o(t) - K_o e \quad (11)$$

$$A_k^T P + P A_k = -Q \quad (12)$$

where $\hat{u}_o(t)$ is the estimated input before correction, $\hat{u}(t)$ is the estimated input after correction, K_o is the linear correction gain, and P and Q are positive definite matrices. The derivation of the updating law listed above will be shown in the next section.

3.3 Stability Analysis of the Enhanced Adaptive Observer

To avoid any confusion, the derivation of the estimation scheme is divided into two steps: pre-correction and after-correction.

a) Pre-Correction

The observer of the system can be written as

$$\dot{\hat{x}} = A\hat{x} + B\hat{u}_o + K(\hat{y} - y) \quad (13)$$

where $\hat{u}_o(t)$ is the estimated input before correction. Subtract Eq.(1) from (13), we have

$$\dot{e} = A_k e + B e_{u_o} \quad (14)$$

where $e_{u_o} = \hat{u}_o(t) - u(t)$.

The Lyapunov candidate function is chosen as

$$V = e^T P e + e_{u_o}^T e_{u_o} \quad (15)$$

Taking the derivative of the Lyapunov function (15) along the trajectory of the error dynamics, we have

$$\dot{V} = -e^T Q e + 2e^T P B e_{u_o} + 2\dot{e}_{u_o}^T e_{u_o} \quad (16)$$

Let

$$\dot{e}_{u_o} = -B^T P e \quad (17)$$

then Eq.(17) becomes

$$\dot{V} = -e^T Q e < 0. \quad (18)$$

In other words, the updating law shown in Eq.(10) will guarantee the convergence of the estimation error for both state and input signals, for the cases when the unknown input $u(t)$ is constant. The aforementioned assumption of the update law imposes a limit on our ability to track time-varying functions. In the following, we will introduce a modification to reduce the effect of this drawback.

b) Modification of the Up-date law

We assumed that the estimated input error e_{u_o} is proportional to the state estimation error e when e is small, i.e.

$$e_{u_o} = K_o e \quad (19)$$

In order to achieve the condition described in Eq.(18), Eq.(19) must be matched with Eq.(17). Take the derivative of Eq.(19) and substitute it into (17), we obtain the following relationship

$$K_o A_k + K_o B K_o + B^T P = 0 \quad (20)$$

From the above equation the proportional gain matrix K_o can be solved. Substituting the obtained matrix K_o into Eq.(19) and from the definition that $e_{u_o} = \hat{u}_o(t) - u(t)$, we have the modified estimation signal

$$\hat{u}(t) = \hat{u}_o(t) - e_{u_o} \quad (21)$$

Compare Eq.(19) with Eq.(21), Eq.(11) is obtained. The block diagram of the estimation scheme is shown in Fig. 3.

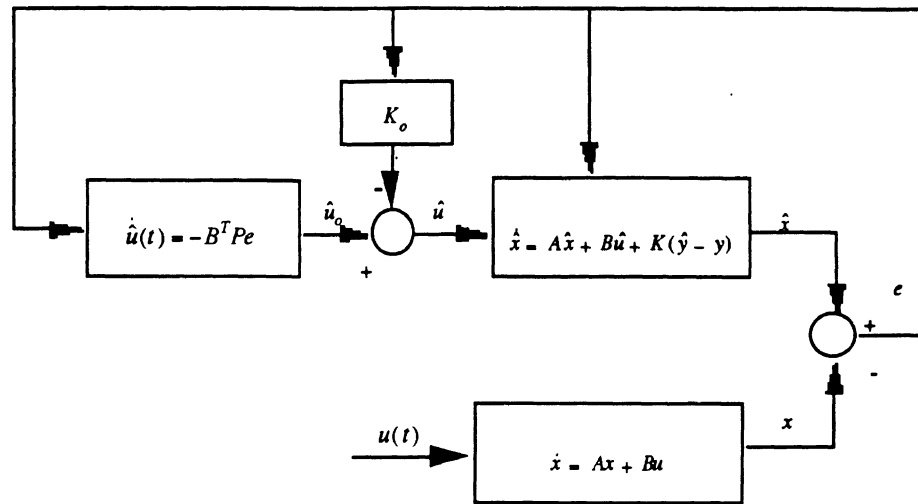


Fig.3 Schematic Representation of Estimation

3.4 Estimation of Road Surface Condition

The estimation of the road surface condition is based on the anisotropic brush model described in section 2.2. The idea is that it needs more tire slip to obtain the same tire-road interactive force when the road is slippery; thus road surface condition can be detected by observing tire shear force vs. tire slip. After estimating the tire longitudinal force, the tire-road friction coefficient μ can be obtained from (3a) and (3b):

$$\hat{\mu} = \frac{a_1 + \sqrt{a_1^2 - 4a_2 a_0}}{2a_2} \quad \text{if } \sigma \leq \sigma_m \quad (22a)$$

$$\hat{\mu} = \frac{\hat{F}}{F_z} \quad \text{if } \sigma > \sigma_m \quad (22b)$$

where

$$\begin{aligned} a_0 &= (\bar{\theta}\sigma)^3 \\ a_1 &= 3(\bar{\theta}\sigma)^2 F_z \\ a_2 &= (3\bar{\theta}\sigma - \hat{F})F_z^2 \\ \bar{\theta} &= \frac{4}{3} l_1^2 l_2 k \\ \hat{F} &= -\frac{\sqrt{(k_x \sigma_x)^2 + (k_y \sigma_y)^2}}{\sigma_x k_x} \hat{F}_x \end{aligned}$$

l_1 , l_2 and k are tire-road patch length, width and stiffness, respectively. l_1 and l_2 depend on the tire pressure and k varies with the tire slip ratio. σ_m is the maximum total strain allowed on the tire.

4. VEHICLE PATH PREDICTION AND TLC CALCULATION

4.1 Vehicle Path Prediction

Vehicle path is determined by the complex interaction between human driver and the vehicle dynamics. To predict the future trajectory of the vehicle, it is a common practice [1] to assume that the steering angle is fixed and the vehicle speed remain constant in this future period of time. This assumption, albeit not true in many cases, will be used in most of the simulation cases in this paper. The prediction of the vehicle future path will be obtained from the bicycle model described in section 2.3. Since the relative motion of the car (with respect to the road) is of interest, the lateral dynamics described in (4) will be augmented to include curved-road scenarios.

a) *Straight Road*

When the road is straight, the following bicycle model is used:

$$\begin{bmatrix} \dot{y} \\ \dot{v} \\ \dot{\phi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 & u & 0 \\ 2C_{sf} + 2C_{sr} & 0 & (-u + \frac{2aC_{sf} - 2bC_{sr}}{mu}) & 0 \\ mu & 0 & 1 & 0 \\ 2aC_{sf} - 2bC_{sr} & 0 & \frac{2a^2C_{sf} + 2b^2C_{sr}}{I_z u} & 0 \end{bmatrix} \begin{bmatrix} y \\ v \\ \phi \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ -2C_{sf} \\ m \\ -2aC_{sf} \\ I_z \end{bmatrix} \delta_{fw} \quad (23)$$

where y is the lateral displacement, v is the lateral speed, ϕ is the vehicle heading angle, and r is the yaw rate.

b) *Curved Road*

When the road is curved, the orientation of the road ϕ_d and the changing rate of the orientation $\dot{\phi}_d$ must be considered. Choosing \dot{y} and $\phi - \phi_d$ instead of v and ϕ as the state variables, the vehicle lateral dynamics in curved road will be

$$\frac{d}{dt} \begin{bmatrix} y \\ \dot{y} \\ \phi - \phi_d \\ r \end{bmatrix} = \begin{bmatrix} 0 & 1 & u & 0 \\ 2C_{sf} + 2C_{sr} & 0 & -\frac{2C_{sf} + 2C_{sr}}{m} & \frac{2aC_{sf} - 2bC_{sr}}{mu} \\ mu & 0 & 0 & 1 \\ 2aC_{sf} - 2bC_{sr} & 0 & -\frac{2aC_{sf} - 2bC_{sr}}{I_z} & \frac{2a^2C_{sf} + 2b^2C_{sr}}{I_z u} \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \phi - \phi_d \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ -2C_{sf} \\ m \\ -2aC_{sf} \\ I_z \end{bmatrix} \delta_{fw} + \begin{bmatrix} 0 \\ -u \\ -1 \\ 0 \end{bmatrix} \dot{\phi}_d \quad (24)$$

4.2 TLC Calculation

TLC (time to lane crossing) is defined as the time for the center of gravity of the vehicle to cross the roadway edges. When calculating the TLC, the steering angle and the vehicle speed are assumed to be constant. TLC has been found to be a good metric for active safety measures. Warning, intervention or control actions can be taken based on the value of TLC. The TLC is calculated based on the lateral motion of the vehicle. More precisely, given the current lateral displacement from the lane edges, and assuming a constant steering input, TLC is defined as the time necessary for the lateral displacement to diminish to zero. It is obvious that the vehicle path prediction and TLC are affected by the tire cornering stiffness and steering input. On the other hand, tire lateral force needs to be calculated based on an estimated cornering stiffness. An initial guess of the tire cornering stiffness is chosen to estimate the vehicle lateral speed and the tire slip angle, the estimated vehicle lateral speed and tire slip angle are then used to calculate the updated tire cornering stiffness. Since the calculation of tire lateral speed and cornering stiffness depends on each other, we must solve them iteratively. Due to this iterative process, the estimation of the tire cornering stiffness is not fast enough to track the variation of the cornering stiffness if the real cornering stiffness varies dramatically.

5. SIMULATION RESULTS

The aforementioned methods were implemented on a simulation program which was developed by the CAPC team of the University of Michigan Transportation Research Institute (UMTRI). A 14 DOF vehicle model [11] was used, which includes the six degree of freedom of the sprung mass, and one degree of freedom for each of the four suspension linear motion and tire rotational motion. The simulation program utilizes the combined-slip magic formula model to compute the

tire forces. In the simulations some variables are assumed to be measured. The wheel speed is measured, which is corrupted with .1% random white noise. This level of accuracy was shown to be achievable under moderate vehicle speeds[12]. The high accuracy of the wheel speed measurement is necessary, since we need the high accurate of the tire slip ratio to identify the road surface condition. The measurement of yaw rate is contaminated by a 0.1% white noise error. The engine torque and braking torque have 5% white noise error. The steering angle of the front wheel is assumed to be measured with a 0.1 degree white noise error.

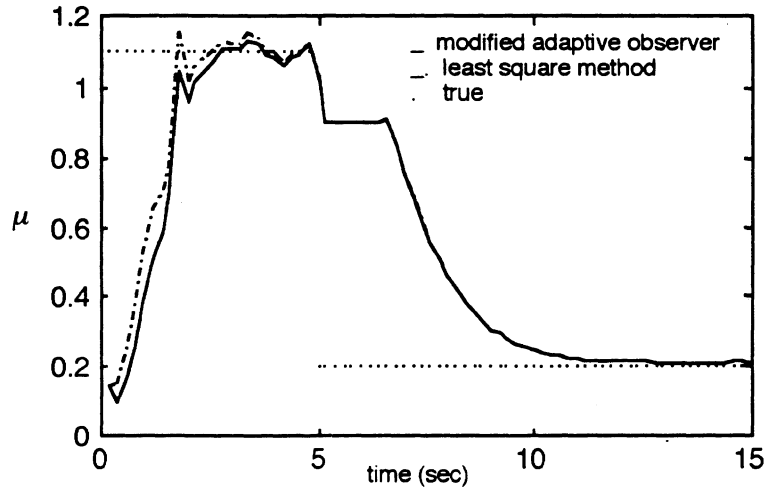


Fig. 4 Estimation of the Road Surface Condition

Both modified adaptive observer and least square algorithms are applied. It was found that both of these methods work well if designed properly. Fig. 4 shows the scenario when the road surface condition experienced a step-changes from $\mu = 1.1$ to $\mu = 0.2$. The vehicle driving torque is 440 N when $\mu = 1.1$ and is 220 N when $\mu = 0.2$. We can see that the fluctuation of the estimation result is large when μ is large. This is because it is difficult to distinguish μ when μ is large (see Figure 2).

Fig. 5 shows the estimated cornering stiffness for both front and rear wheels under three different road surface conditions. The steering wheel is assumed to be kept at 30 degree (=1 degree steering at tires). We can see that when $\mu = 1.1$ and $\mu = 0.5$ the cornering stiffness didn't deviate much from the initial guesses, but when $\mu = 0.2$ the cornering stiffness will reduce to about 4000 N/rad. Since the cornering stiffness will not vary much when μ is larger than 0.5, we will focus on the cases when the road surface is slippery. When μ is high, we would expect TLC to be very insensitive to the road surface condition. Fig. 6 shows the predicted path of the vehicle when the wheel steering angle is 20 degree (=0.67 degree steering at tires) and $\mu = 0.1$. The results show that the TLC based on up-dated cornering stiffness is much more accurate. We also expect that the improvement from updated cornering stiffness will be larger when TLC is higher. In other words, it makes more sense to use updated cornering stiffness for warning, rather than on intervention and control actions.

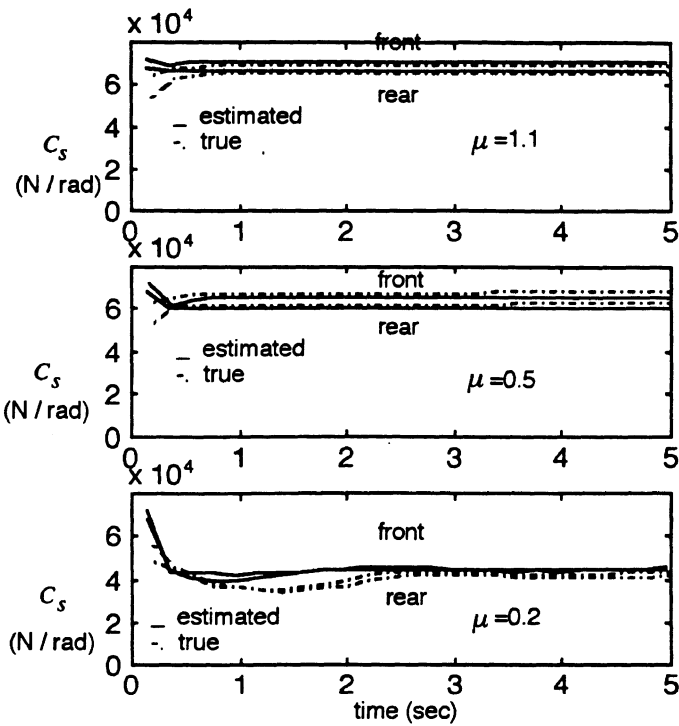


Fig: 5 Estimation of the Cornering Stiffness

In figure 7, the vehicle is assumed to have a 20 degree wheel steering angle at 18 m/sec. It is assumed that the road is straight and the distance between the c.g. of the vehicle and the road edge is 2m. The rms value of TLC estimation error is 0.02 second and 0.28 second for the updated and un-updated cases, respectively. Fig. 8 shows the calculated TLC under the same driving conditions except that the road has a radius of curvature = 1000m. The rms value of TLC estimation error is 0.187 second and 0.65 second for the updated and un-updated cases. It should be noted that the difference between the estimated TLC and the nominal TLC is larger in the in the curved-road case. The major reasons is that the TLC is larger in this case. Because the initial value of the cornering stiffness of the updated one is the same of that of the un-updated one, the initial calculated TLC will be the same in both cases.

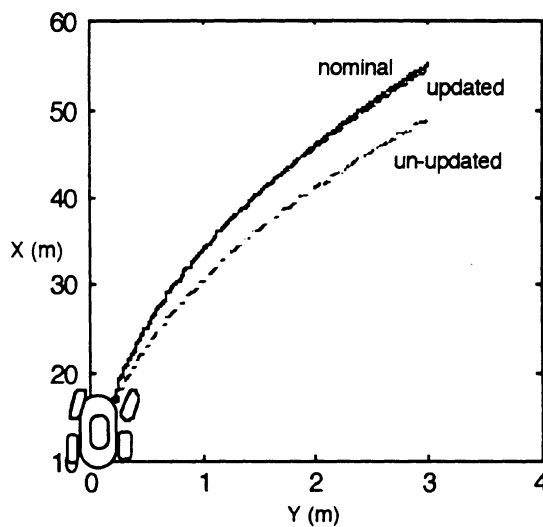


Fig. 6 Predicted Vehicle Path

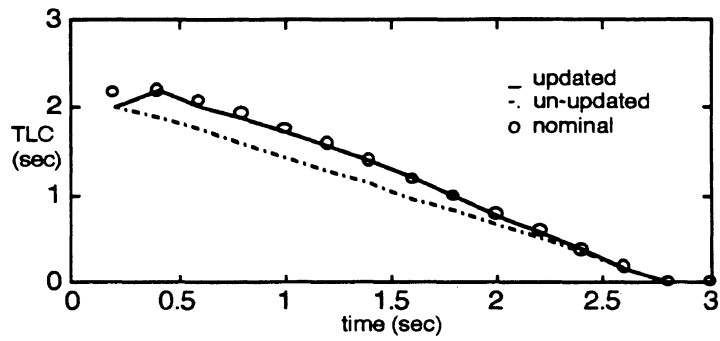


Fig. 7 Calculated TLC (Straight Road)

Fig. 9 shows the TLC of the vehicle with zero steering angle on a curved road. It can be seen that the 0.1 degree measurement noise of the front wheel steering angle has little effect on the TLC calculation. Only the results of the updated case is presented because the difference between the cornering stiffness of the updated and un-updated cases is small when tire slip angle is very small.

Fig. 10 shows the effect of vehicle load variation. A 200 kg luggage is assumed to be loaded in the trunk. Important parameters of the vehicle including c.g. location, vehicle mass and moment inertia are changed accordingly. The driving conditions and road geometry are the same as those of Fig. 8. The results show that the TLC based on updated information is still much more accurate than the un-updated case. The rms value of the estimation error is 0.09 second and 0.35 second, respectively.

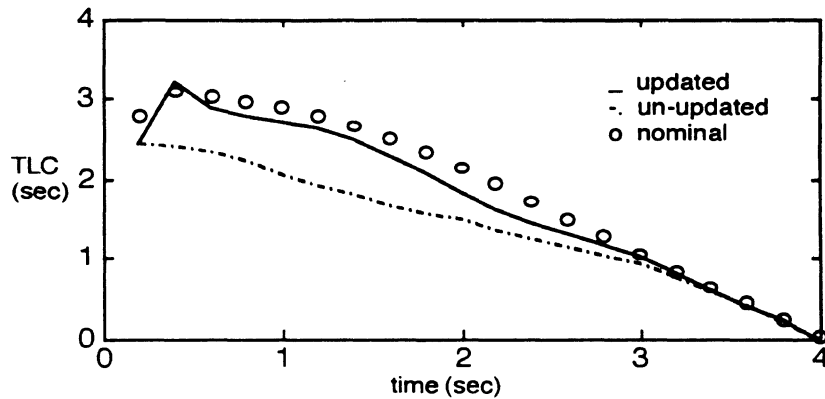


Fig. 8 Calculated TLC (Curved Road)

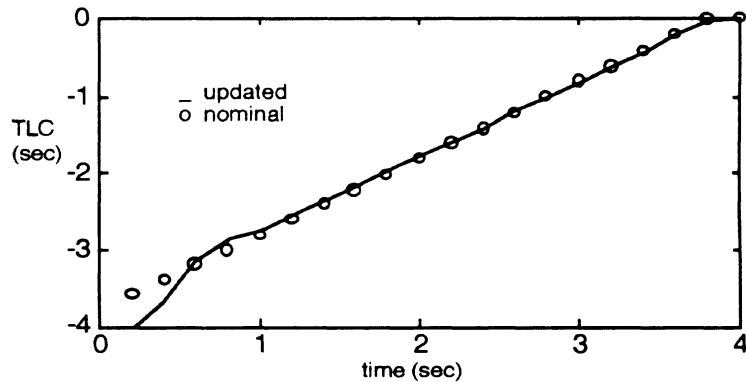


Fig. 9 Calculated TLC (curved road, zero steering angle)

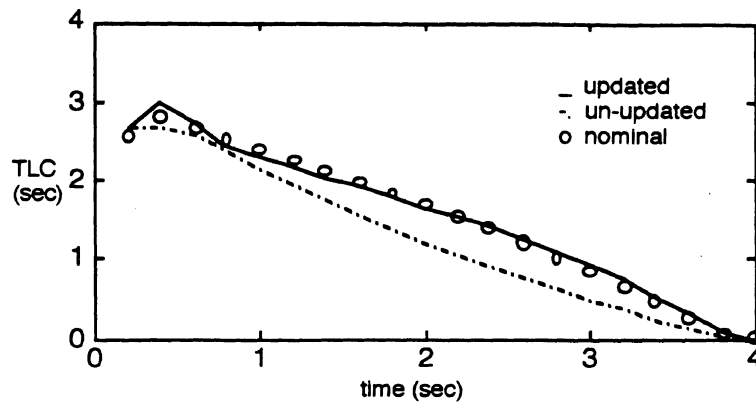


Fig. 10 Calculated TLC (curved road, extra load in trunk)

The effect of road gradient is shown in Fig. 11. The gradient of the road is assumed to be 0.05 rad (about 3%). All the other simulation parameters are the same as those of Figure 8. The rms value of TLC estimation error is 0.33 second and 0.78 second for the updated and un-updated cases, respectively. Under this circumstance, the identification schemes cannot provide satisfactory improvements.

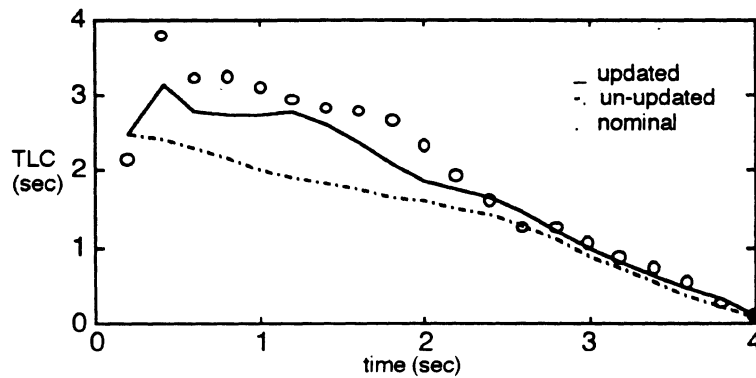


Fig. 11 Calculated TLC (curved road, 0.05 rad gradient)

The effect of road super-elevation is presented in Fig. 12. The results show that the TLC of the updated case are closer to the nominal value than the un-updated does. The rms values of TLC error are 0.037 second and 0.21 second. We can see that the rms values of TLC error are reduced significantly of the un-updated one as compared to the rms values of TLC error of the case shown in Fig. 8. It is because that when the road has a proper super-elevation the tire forces generated by the steering will be reduced, i.e. the tire slip angle will be reduced. Since the cornering stiffness will be insensitive to road surface conditions when the slip angle is small, it can be expected that the calculated TLC of un-updated one will be more close to the nominal one if the road super-elevation is more than 0.05 rad.

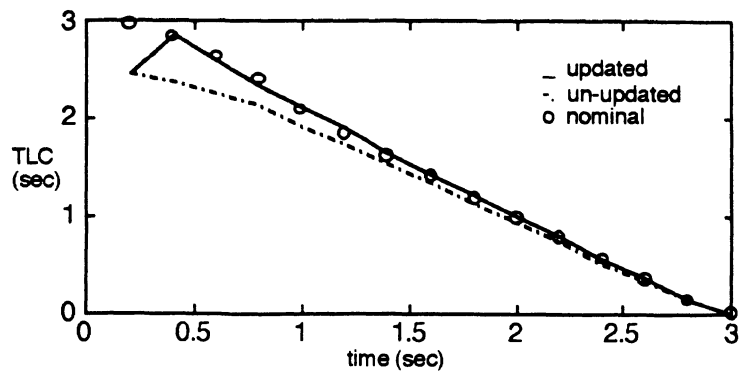


Fig. 12 Calculated TLC (Curved Road, 0.05 rad Super-elevation)

When the vehicle is steered by wiggling input (a rough approximation of a drunk driver), the TLC was only slightly improved in a certain region which can be viewed more clearly in Fig. 14. The main reason of this dismal improvement is because the estimation scheme is sluggish. In this simulation the steering input is sinusoidal which means that the effective cornering stiffness changes dramatically, thus the estimated cornering stiffness cannot follow the change of cornering stiffness well. The TLC calculation is improved more when the steering angle is large, i.e., when the cornering stiffness is not fast varying and effective cornering stiffness is small.

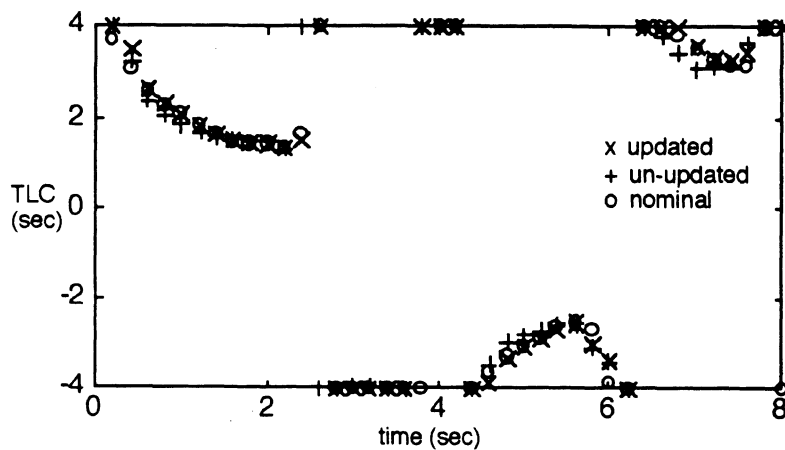


Fig. 13 Calculated TLC (straight road, sinusoidal steering angle input)

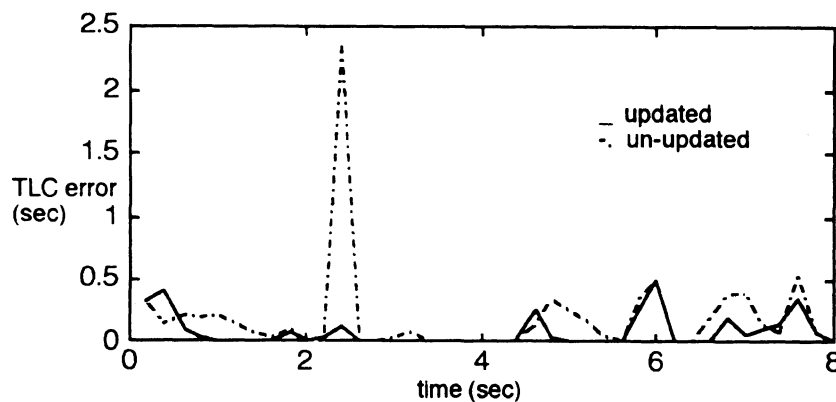


Fig. 14 TLC Error (straight road, sinusoidal steering angle input)

6. CONCLUSIONS

Both methods of modified adaptive observer and least square algorithm work well to estimate the road surface condition. The purpose of using the modified adaptive observer is that it can estimate the time variant nonlinear input without a mathematical form of the input. The effect of the vehicle parameters (tire cornering stiffness) on the TLC calculation is the main performance evaluation metric in this paper. It is found out that the accuracy of the TLC can be improved as large as 0.5 second if the road surface condition is considered, but for some cases the accuracy of the TLC calculation of the updated one is still not good enough even though it has better results than that of the un-updated one. The solution is that we should modify the scheme of the vehicle lateral speed estimation. It is also found that the TLC calculation error of the case of the cornering stiffness without updating will be reduced if the road has a proper super-elevation. For the case of sinusoidal steering input, the updated one can only have better performance on some regions. How to increase the response and accuracy of the cornering stiffness estimation is another challenge task. It is observed that the TLC can be improved more significant when TLC is large. It means that the update of the cornering stiffness is more useful for warning than Intervention/control.

7. ACKNOWLEDGMENT

The work is supported by the ITS Research Center of Excellence at the University of Michigan. The authors also wish to thank Prof. A.G. Ulsoy and other CAPC team members for their support.

REFERENCES

1. Lin, C.-F. and Ulsoy, A.G., "Calculation Of The Time To Lane Crossing And Analysis Of Its Frequency Distribution," American Control Conference, 1995.
2. Dieckmann, T., "Assessment of Road Grip by way of Measured Wheel Variables," Proceedings FISITA, London , June 1992.
3. Eichhorn, U. and Roth, J., "Prediction And Monitoring of Tyre/Road Friction," Proceedings FISITA, London , June 1992.
4. Ito, M., Yoshioka, K. and Saji, T., "Estimation of Road Surface Conditions Using Wheel Speed Behavior," SAE no. 9438826.
5. Pal, C., Hagiwara, I., Morishita, S. and Inoue, H., "Application of Neural Networks in Real Time Identification of Dynamic Structural Response and Prediction of Road-Friction Coefficient μ from Steady State Automobile Response," SAE no. 9438817.
6. Pasterkamp, W. R. and Pacejka, H. B., "On Line Estimation of Tyre Characteristics for Vehicle Control," American Control Conference, 1995.
7. Pacejka, H. B. and Sharp, R. S., "Shear Force Development by Pneumatic Tyres in Steady State Conditions: A Review of Modeling Aspects," Vehicle System Dynamics, Vol. 20, 1991, pp. 121-176.
8. Mendel, J. M., "Lessons in Estimation Theory for Signal Processing, Communication, and Control," Prentice Hall, Inc., Englewood Hill, New Jersey, 1995.
9. Messner, W., and Horowitz, R., "Identification of a Nonlinear Function in a Dynamical System," ASME Journal of Dynamic Systems, Measurement, and Control, Dec. 1993, Vol. 115, pp. 587-591.
10. Gillespie, T. D., "Fundamentals of vehicle dynamics," Warrendale, PA : Society of Automotive Engineers, 1992.
11. Venhovens P. J. Th., "Optimal Control of Vehicle Suspensions," Ph.D. Thesis, Delft University of Technology, Dutch, 1993.
12. Gustafsson F., "Slip-Based Estimation of Tire-Road Friction", Submitted to the IEEE Transactions on Control Systems Technology.

Appendix G

"On-Line Identification of Driver State for Lane-Keeping Tasks"

T.E. Pilutti and A. G. Ulsoy,
Proceedings of the 1995 American Control Conference

On-Line Identification of Driver State for Lane-Keeping Tasks

Tom Pilutti, Research Engineer, Ford Research Laboratories, Dearborn, MI, pilutti@fmsrlw.srl.ford.com
Galip Ulsoy, Professor of Mechanical Engineering, University of Michigan, Ann Arbor, MI, ulsoy@umich.edu

Abstract

On-line identification of driver state is a desirable element of many proposed active safety systems (e.g., collision detection and avoidance, automated highway and road departure warning systems). Here we consider driver state assessment in the context of a road departure warning and intervention system. A system identification approach, using vehicle lateral position as the input and steering wheel position as the output, is used to develop a model and to continually update its parameters during driving. Driving simulator results indicate that changes in the bandwidth and/or parameters of such a model may be useful indicators of driver fatigue.

Motivation

On-line identification of driver state is motivated by the increased interest in vehicle driver assistance features such as antilock brakes, automatic cruise control, collision warning and road departure warning. These driver assistance features are centered around the ability to perform a control-type task while at the same time keeping the driver in the control loop. Antilock brake systems (ABS) are a current example. ABS works in conjunction with the driver brake pedal input to prevent wheel lock. Automatic cruise control and the potential application of collision avoidance takes over the function of headway control, but it remains the duty of the driver to steer the vehicle. Road departure warning can be viewed as a further extension where an additional driver support system acts as a co-pilot to monitor lane keeping performance.

One way to realize such a lane keeping system is to approach it as an automatic steering control problem, which augments the driver's steering commands with steering commands generated by a controller algorithm that seeks to keep the vehicle centered in the lane. With this approach, warnings are not needed and intervention is essentially continuous. On the other hand, it also assumes that the driver desires to drive like a controller, i.e., tracking the lane center. Typically, though, most drivers view lateral lane position control with a pseudo-deadband approach. Some drivers have a large deadband while others keep the deadband tight - as far as the driver is

concerned, lateral lane position is satisfactory as long as the vehicle does not run off the road.

A variation of the automatic steering system which includes driver behavior may make it possible to accommodate different driving styles. If driver actions can be monitored, it would be possible to "personalize" the warning to adapt to different driving styles. In addition, it would also be possible to track changing driver parameters during a long drive. This would enable a driver assistance system to provide warning as a function of changing driver *state* (e.g., alertness or fatigue).

Background

Taken from the perspective of driver modeling, the literature is rich with examples. There is an optimal preview model [MacAdam 1981] which internalizes a vehicle model as part of the driver response to the upcoming road path, a multiple input system [McRuer and Weir 1969] which closes the loop on lateral position and heading angle errors, and a control theoretic model [Modjtahedzadeh and Hess 1993] which uses previewed lateral position as the feedback term with internal "driver" states. These models represent the starting point for driver model identification for this paper. Additionally, there are examples of statistics-based driver models [Knipling and Wierwille 1993] that seek to capture anomalies in driver behavior such as fatigue. And finally, there are neural network approaches to driver modeling for vehicle control [Thorpe *et al* 1992] where driver model steering angle output is mapped from vision-based road views.

Purpose and Scope

We seek a method to detect changes in driving patterns so that an assessment of driver alertness/performance can be made. This assessment will be used first as an input to a road departure warning system. Our hypothesis is that system identification techniques can be used to form a set of driver parameters that can be correlated with various levels of lane keeping performance. Variations in these key parameters will then permit us to monitor driver state.

Driver Models vs. Simulator Data

Driver models found in the literature are used primarily for vehicle dynamics studies for maneuvers such as lane changes, emergency maneuvers and general tracking. The function of the driver model is to approximate driver tracking performance so that vehicle dynamics models can be evaluated. They are “control” models, and as such, they assume the “driver” goal is to follow the road reference to the best of its ability. An example is seen in Figure 1 where an optimal driver model [MacAdam 1981], seeking to minimize the error between the future road trajectory and the predicted vehicle trajectory, controls the vehicle model to a standard deviation of approximately 0.001 m with steering wheel motions in the 1 hz region.

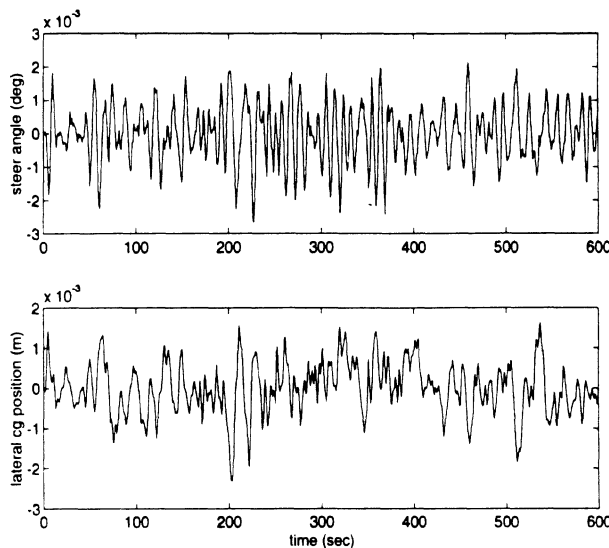


Figure 1 Optimal driver model on straight road (asphalt)

Data from a driving simulator with a similar road surface, but now with a real person at the steering wheel, is shown in Figure 2. The lane tracking performance is quite different; the lateral position standard deviation is 0.25 m with steering wheel motions under 0.25 hz.

One can observe a “complacency zone” in the steer angle data in Figure 2 which is not evident in Figure 1. The steering position remains constant while small lane deviation (and heading angle) errors build. Past some threshold the driver makes a correction. The complacency zone is a function of disturbance level, and will also depend on other factors such as traffic density and road curvature.

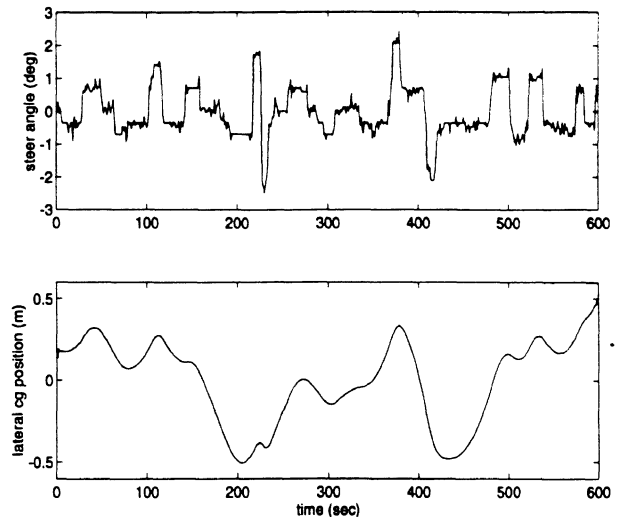


Figure 2 Driver simulator on straight road (asphalt)

The implications are that driver models for handling experiments do not replicate driving behavior seen in highway-type lane keeping tasks. There may be simpler structures that can capture changing driver behavior, and while the identification effort may not produce a driver model capable of stand-alone simulation, the main purpose here is to support driver state assessment.

Identification Approach to Driver State Assessment

We are currently pursuing a system identification approach (see Figure 1), and have assumed an ARX structure as the candidate model structure:

$$A(q)y(t) = B(q)u(t - nk) \quad (1)$$

where $y(t)$ is the driver model steering position output (δ) and $u(t-nk)$ is the delayed driver model input (in this case lateral vehicle position, y).

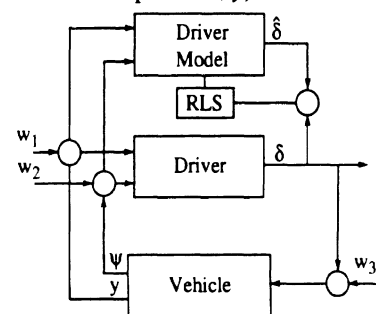


Figure 3 Driver Identification Framework

Figure 3 describes the closed loop system where the driver model parameters are estimated on-line in a recursive algorithm. Before this can be achieved, preliminary driving simulator data is to be tested for informative value (data collected in feedback loops should be handled with care). The data can then be

tentatively fitted to ARX/ARMAX models with delay to examine suitability with standard system identification techniques. Proposed inputs to the driver model are lateral deviation, y , and heading angle, ψ . Noise inputs, w_i allow the model to represent sensor uncertainties. Additional test data with driving subjects in various degrees of subjective alertness is then needed to examine driver parameter variations from test to test. Final validation tests will then occur with the on-line system with a comparison to subject alertness level (either subjective or measured). Indicators such as parameter values and pole/zero locations can be used to track changes in driver state. We will compare this approach with results shown in [Thorpe *et. al* 1992] to demonstrate the ability to detect driver fatigue.

Before proceeding further, the existence of persistent and sufficiently rich excitation must be considered. Since identification needs to occur during normal driving, we are not able to perturb the system to induce "interesting" driver behavior (or wait for high demand driving tasks). Rather, it is assumed that normal driver tasks will provide sufficient excitation for identification. This assumption requires further investigation. Figure 2 showed typical driver steer patterns, and the plots in Figure 4 show the effect of adding road surface noise to the simulator vehicle dynamics model. It shows improved steering power spectra in the 0.5 hz range. The advantage to adding disturbances through the road surface is that the surface is a realistic representation of vibration induced in the vehicle and is propagated through the vehicle dynamics model. The disadvantage is that road inputs require a vehicle dynamics model with sufficient degrees of freedom to allow tire displacement inputs. The simulator runs in the remainder of this paper use the asphalt road surface shown in Figure 4 (middle).

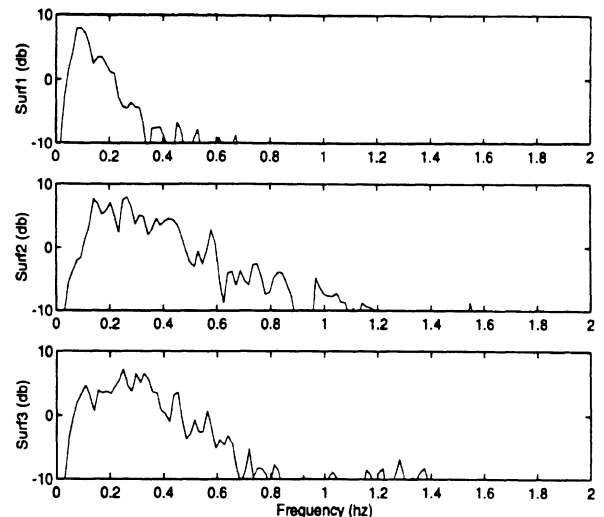


Figure 4 Driving Simulator steering position spectra (top) perfectly smooth road, (middle) and (bottom) asphalt and concrete road surfaces

In this paper, we report preliminary results from this proposed approach where only y (not y and ψ) in Figure 3 is fed back to the driver model, and the parameter estimates are obtained using off-line least square estimates from driver simulator data. An ARX(3,3,1) model (i.e., the order of polynomials A and B are both 3, and delay $n=1$ in Eq. 1) was determined to be adequate.

Results & Discussion

Using straight-line driving simulator data, we have been able to fit ARX models using lateral lane deviation as the input with steering position as the output. The standard deviation of y is observed to vary over the course of a one hour driving test as shown in Figures 5-7. Figure 5 shows the steering wheel angle, heading angle and lateral position for a portion of a straight-line driving test. The overall (for the entire test) trend that can be seen in lateral deviation variance (second trace in Figure 6) that are similar to variations found in [Thorpe *et. al* 1992]. No trends related to driver fatigue are evident from the mean value of y (the first trace in figure 6). Figure 7 is an example of an ARX(3,3,1) model fit to non-overlapping portions of the test run with lateral position as the input and steering position as the output.

The results in Figure 7 show a 3-step ahead prediction of the steering wheel angle using the ARX model, compared to actual steering wheel angular position. The model fit, and the output prediction are quite good.

Careful analysis of the poles and zeros of this ARX(3,3,1) model revealed that two of the poles and zeros in the model were related to the quantization error (evident in Figures 2 and 5) associated with the measurement of the angular position of the steering wheel. This is due to the limited resolution of the steering angle sensor used in the driving simulator. Consequently, the third (dominant) pole only was found to be directly related to driver steering behavior. Figure 8 shows the value of this dominant pole versus the simulation time. Initially the pole is at $z \approx 0.94$, and near the end of the simulation is at $z \approx 0.96$. This corresponds to a change in the effective "time constant" of the driver from $\tau \approx 0.8$ sec to $\tau \approx 1.2$ sec, and correlates well with the increased standard deviation of y as shown in the second trace of Figure 6.

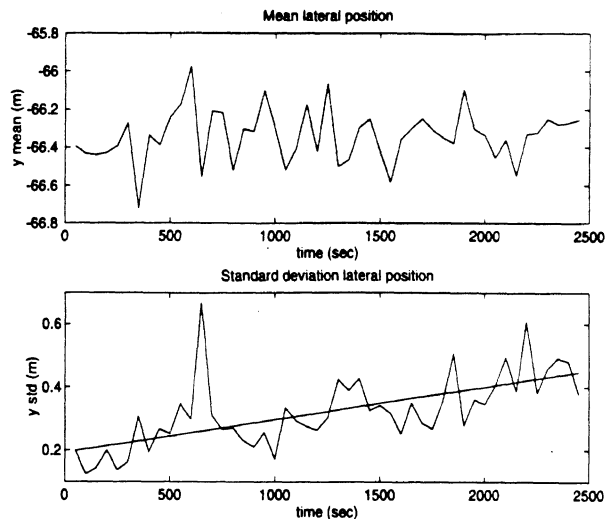


Figure 6 Lateral position mean and standard deviation for 1000 sample-wide non-overlapping windows

Concluding Remarks

A system identification approach to driver state assessment has been proposed and evaluated using

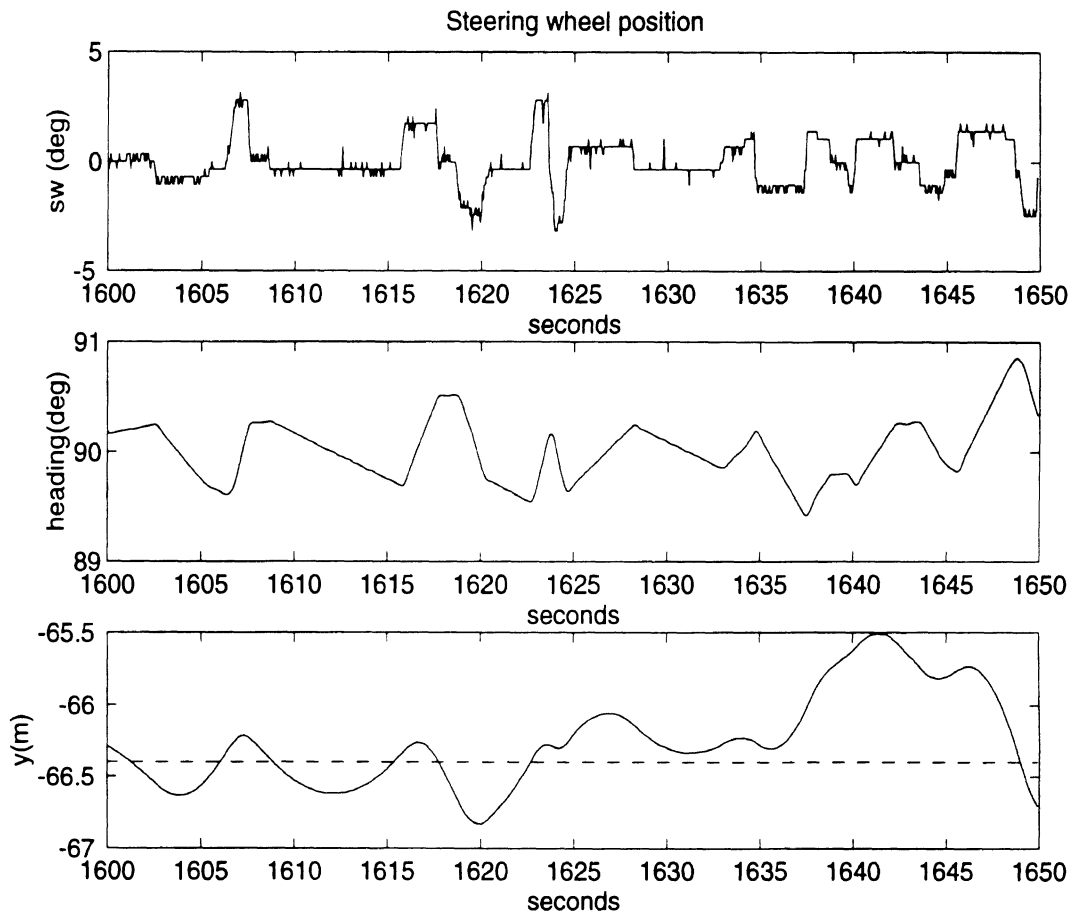


Figure 5 Portion of driving simulator data

driving simulator tests. The approach uses an ARX model to represent the relationship between vehicle lateral position y and the driver steering wheel angular position δ . The ARX model parameters are then continually estimated during the driving task. The results are encouraging in that good ARX model fit is obtained, and changes in driver behavior (i.e., increased standard deviation of y due to driver fatigue) can be detected from changes in the poles of the model.

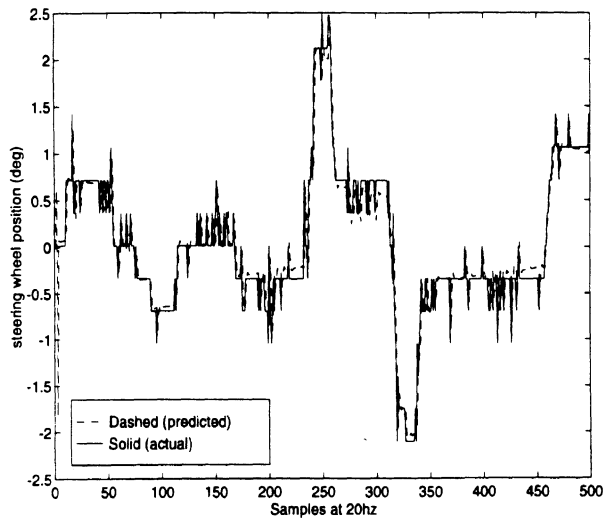


Figure 7 3-step ahead steering angle prediction

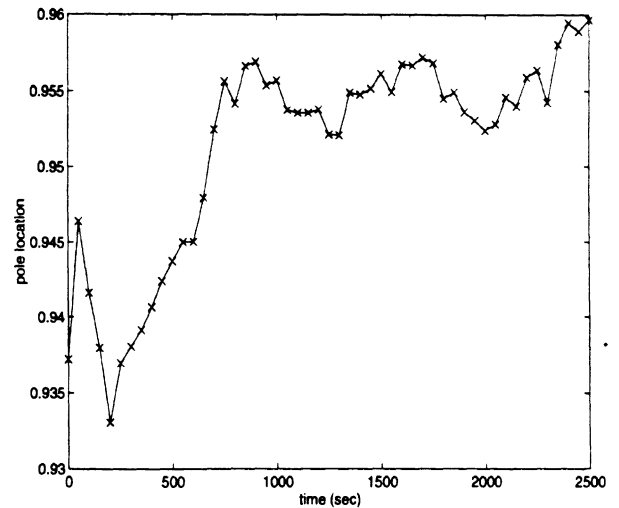


Figure 8 Dominant ARX pole locations

References

- MacAdam, C. C. "Application of an Optimal Preview Control for Simulation of Closed-Loop Automobile Driving." *IEEE Transactions on SMC*, 1981, 11(6).
- McRuer, D.; Weir, D. H. "Theory of Manual Vehicular Control." *Ergonomics*. 1969; 12(4).
- Modjtahedzadeh, A.; Hess, R. "A Model of Driver Steering Control Behavior for Use in Assessing Vehicle Handling Qualities." *J. of Dynamic Systems, Measurement and Control*, Sep, 1993.
- Knipling, R.; Wierwille, W. "U.S. IVHS Research: Vehicle-Based Drowsy Driver Detection". *Vigilance and Transport Conference*. INRETS/USTRB; 1993 Dec.
- Thorpe, C.; Hebert, M.; Kanade, T.; Shafer, C. "The New Generation System for the CMU Navlab". Vision-based Vehicle Guidance, Springer-Verlag; 1992.

Appendix H

"Vehicle Steering Intervention Through Differential Braking"

T.E. Pilutti, A. G. Ulsoy, and D. Hrovat
Proceedings of the 1995 American Control
Conference

Vehicle Steering Intervention Through Differential Braking

Tom Pilutti¹, Galip Ulsoy² and Davor Hrovat³

Abstract

This paper examines the usefulness of a brake steer system (BSS) which uses differential brake forces for steering intervention in the context of Intelligent Vehicle Highway Systems. The resulting moment on the vehicle affects yaw rate and lateral position, thereby providing a limited steering function. The steering function achieved through BSS can then be used to control lateral position in an unintended road departure system. Models for the vehicle and the brake system are presented. A state feedback regulator and PID controller are developed to explore BSS feasibility and capability. Computer simulation results are included. In addition, we define a region of allowable steady state differential braking forces for given ranges of steering angle, vehicle speed and road coefficient of friction. We use multiple simulations of a nonlinear vehicle model (nonlinear tire model) to determine stable control regions. We assume steady state steering and brake steer force inputs, and use non-constant yaw rate as the stability criterion. The resulting maximum steady state brake forces form a conservative upper bound on brake steer operational regions.

1 Introduction

The use of differential brake forces has been applied to increase handling performance during a combined braking and cornering maneuver [Nakazato *et al.* 1989]. Their results show that a decreased yaw moment in the direction of the turn can be achieved by altering the left/right brake distribution such that the brakes on the inside of the turn have less braking force than those on the outside of the turn. The distribution is controlled by hydraulic valves which operate as a function of vehicle roll. The concept has been extended to control the brake force distribution as a function of steering wheel input during the turn [Matsumoto *et al.* 1992]. A yaw rate model reference controller establishes the desired yaw rate given the steering wheel input, and then makes corrections to the anti-lock brake system pressure in order to track the reference yaw rate.

This paper explores the use of differential braking as a means to perform limited steering functions. The proposed brake steer system (BSS) is motivated by Intelligent Vehicle Highway Systems (IVHS) Advanced Vehicle Control Systems (AVCS) research into systems such as anti-road departure, collision avoidance and intelligent cruise control. In these systems it is feasible that limited steering capabilities would provide an enhanced degree of system performance. In anti-road departure and collision avoidance, for example, a driver over-rideable course correction could be performed after an appropriate warning in response to an impending dangerous condition such as under-driving a turn in the case of anti-road departure, or an oncoming vehicle in the case of collision avoidance. Intelligent cruise control may benefit from the ability to react to conditions where a vehicle cuts in front of the autonomous cruise control vehicle. Each of these examples show the usefulness of limited steering ability to intervene and alter vehicle course. There are numerous technical, legal and social issues involved in altering the course of the vehicle beyond that

¹Research Engineer, Ford Research Laboratories, Dearborn, MI and Student Member of ASME

²Professor of Mechanical Engineering and Applied Mechanics, University of Michigan, Ann Arbor, MI and Fellow of ASME

³Principal Staff Engineer, Ford Research Laboratories, Dearborn, MI and Member of ASME

specified by the driver. This paper assumes that the legal and social issues can be contained, and that some degree of steering intervention is deemed desirable.

The key idea with steering intervention is that the driver remains in the loop, and that any control efforts only augment those given by the driver. The driver provides the primary steering commands, with intervention providing additional assistive lateral control efforts. This level of control is contrasted with automatic lateral control where the driver is effectively out of the loop, and at most provides override input [Peng *et al.* 1992].

Intervention can be implemented in the “conventional” sense by viewing intervention as a subset of steer-by-wire actuation. Steer-by-wire enables additive augmentation of the driver steer command, and can be done using either a series or parallel approach [Peng *et al.* 1992] [Tran 1991]. The series approach performs the additive function after the driver, at the actuator, where the driver does not receive feedback regarding the augmenting commands. The parallel approach performs the additive function before the actuator, and provides feedback in the form of steering wheel movement, regarding the augmenting commands. The series and parallel approaches assume, however, that a steering-augmentation-type system is on board the vehicle. For applications such as Program on Advanced Technology for the Highway (PATH) [Shladover *et al.* 1991] where autonomous driving is the goal, such an assumption is valid. Systems seeking to perform at most intervention, on the other hand, permit other less expensive forms of actuation to be considered.

One such alternative is the use of differential braking to perform intervention steering maneuvers. Such a brake steer system (BSS) has an advantage over steer-by-wire implementations in that the steering system remains intact, while the required anti-lock brake system (ABS) modifications have been shown to be minimal [Matsumoto *et al.* 1992], and will be discussed in the section on control. Brake steer, however, is not as efficient as pure steering. The brakes actuated to induce the yaw moment will also serve to decelerate the vehicle. Despite the initial reaction to this side effect, it is possible that the effect will serve a purpose in the psychology of the road keeping problem, and motivate the driver to eliminate the condition prompting road departure intervention.

Sections 2 and 3 discuss the bicycle vehicle model with the brake steer input used in system design, and the simple ABS model that seeks to reflect dynamic as well as saturation effects. Section 4 uses the model to form PID-type and state variable feedback controllers, and discusses the issues in implementation. Two appendices provide ABS and traction control system (TCS) background.

2 Vehicle Model

The brake steer moment imposed on the linearized, 2 degree of freedom (DoF) vehicle model is implemented by adding the external moment, M_{BS} , to the vehicle diagram (Figure 1), and then summing forces in the Y direction and moments about the Z direction to yield [Segel 1992] [Asgari and Hrovat 1990]:

$$\sum F_Y : \quad m(\dot{v} + ru) = F_{1Y} + F_{2Y} \quad (1a)$$

$$\sum M_Z : \quad I_z \dot{r} = aF_{1Y} - bF_{2Y} + M_{BS} \quad (1b)$$

where

$m =$ vehicle mass (kg)

$v =$ relative lateral velocity (m/s)

$r =$ yaw rate (rad/s)

$u =$ forward velocity (m/s)

$I_z =$ yaw moment of inertia ($kg \cdot m^2$)

$F_{1Y}, F_{2Y} =$ respective front and rear lateral tire forces (N)

$a, b =$ respective lengths from mass center to front and rear (m)

$\sum F_X$ is not included since longitudinal forces play a minor role in the lateral response if vehicle forward speed, u , remains constant (i.e., steady turn conditions). This is a reasonable assumption if the braking due to brake steer does not cause an appreciable decrease in vehicle speed. This assumption will be

investigated using a nonlinear model, and may require consideration during controller design [Hessburg *et al.* 1991a]. The linearized vehicle model is useful for control design, and has been shown to provide accurate response characteristics compared to more complex models for conditions up to 0.3g lateral acceleration [Segel 1992].

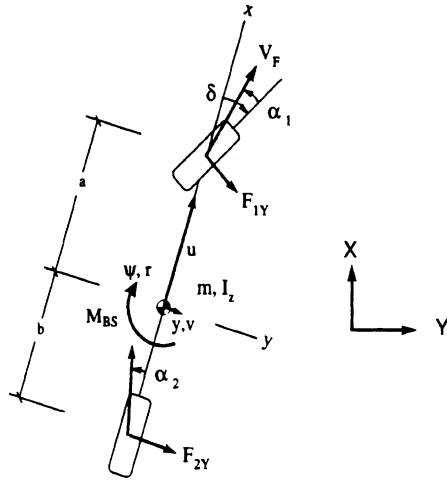


Figure 1 2 DoF Bicycle Model

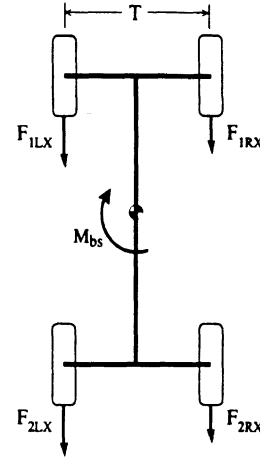


Figure 2 Brake Steer Application

The input brake steer moment, M_{BS} , is generated by different braking forces on the left as compared to the right of the vehicle (Figure 2). Longitudinal braking forces can be applied to the front, the rear or both to achieve differential braking. Possible detrimental issues involved in only using the rear brakes will be discussed later. The main point is that the input moment is formed by the braking differential through

$$M_{BS} = \frac{T}{2}(F_{RX} - F_{LX}) = \frac{T}{2}F_{BS} \quad (2)$$

where

$T =$ vehicle width or track (m)

$F_{RX}, F_{LX} =$ respective right and left longitudinal tire forces (N)

M_{BS} produces yaw rate, r , which in turn affects the front and rear sideslip angles, α_1 and α_2 respectively, through

$$\alpha_1 = \frac{v + ar}{u} - \delta \quad \text{and} \quad \alpha_2 = \frac{v - br}{u} \quad (3)$$

The resulting lateral forces are determined by the front and rear sideslip angles and the tire cornering coefficients, C_{α_i} , $i = 1, 2$,

$$F_{1Y} = C_{\alpha_1}\alpha_1 \quad \text{and} \quad F_{2Y} = C_{\alpha_2}\alpha_2 \quad (4)$$

where F_{1Y} and F_{2Y} reflect the lateral tire forces due to both the brake steer input, M_{BS} , as well as the normal steering input, δ . C_{α_i} in (4) are expressed in units of $\frac{N}{rad}$, and represent the cornering stiffness for combined left and right tires of the bicycle model. In the case where there is no steering input δ , the lateral forces generated by the sideslip angles provide acceleration needed to vary the lateral position of the vehicle. Although yaw moment does not directly impact (1a), the magnitude of differential braking forces affects the sideslip angles through the yaw rate.

Making the substitutions for lateral forces, F_{1Y} and F_{2Y} , into (1a)-(1b), and replacing M_{BS} with the longitudinal tire forces in (2), yields the state equations in the form:

$$\begin{Bmatrix} \dot{v} \\ \dot{r} \end{Bmatrix} = \begin{bmatrix} \frac{1}{mu}(C_{\alpha 2} + C_{\alpha 1}) & \frac{1}{mu}(-bC_{\alpha 2} + aC_{\alpha 1} - mu^2) \\ \frac{1}{I_z u}(aC_{\alpha 1} - bC_{\alpha 2}) & \frac{1}{I_z u}(a^2C_{\alpha 1} + b^2C_{\alpha 2}) \end{bmatrix} \begin{Bmatrix} v \\ r \end{Bmatrix} + \begin{bmatrix} \frac{-C_{\alpha 1}}{m} & 0 \\ \frac{-aC_{\alpha 1}}{I_z} & \frac{T/2}{I_z} \end{bmatrix} \begin{Bmatrix} \delta \\ F_{BS} \end{Bmatrix} \quad (5)$$

with the output equation:

$$\begin{Bmatrix} \ddot{y}_{CM} \\ r \end{Bmatrix} = \begin{bmatrix} \frac{C_{\alpha 1} + C_{\alpha 2}}{mu} & \frac{aC_{\alpha 1} - bC_{\alpha 2}}{mu} \\ 0 & 1 \end{bmatrix} \begin{Bmatrix} v \\ r \end{Bmatrix} + \begin{bmatrix} \frac{-C_{\alpha 1}}{m} & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} \delta \\ F_{BS} \end{Bmatrix} \quad (6)$$

where the inertial lateral acceleration, $\ddot{y}_{CM} = \dot{v} + ru$, and the yaw rate, r , are the outputs. Using the following numerical parameter values shown in Table 1 for a nominal Ford Taurus, the dimensionless understeer coefficient, $U = \frac{m g (a C_{\alpha 1} - b C_{\alpha 2})}{C_{\alpha 1} C_{\alpha 2} (a + b)} = 0.0267$, and the characteristic polynomial, $\Delta(s) = s^2 + 12.2s + 61.6$ with distinct roots $-6.1 \pm j4.9$. The steering input, δ , is expressed in degrees and is converted to radians inside the model. The transfer function matrix for the system is:

$$\begin{Bmatrix} \ddot{y}_{CM}(s) \\ \dot{r}(s) \end{Bmatrix} = \frac{1}{\Delta(s)} \begin{bmatrix} 1.29s^2 + 10.6s + 173 & K_{BS}(4.43(10^{-4})s + 0.05) \\ 1.01s + 6.23 & K_{BS}(3.63(10^{-4})s + 1.78(10^{-3})) \end{bmatrix} \begin{Bmatrix} \delta(s) \\ F_{BS}(s) \end{Bmatrix} \quad (7)$$

Vehicle mass, m	1670 kg (114 slug)
Moment of inertia, I_z	2100 kg - m ² (1547 slug-ft ²)
Forward speed, u	27.78 m/sec (62.2 mph)
CG to front wheel, a	0.99 m (3.2 ft)
CG to rear wheel, b	1.7 m (5.5 ft)
Front cornering coefficient (2 tires), $C_{\alpha 1}$	-123190 N/rad (-483 lb/deg)
Rear cornering coefficient (2 tires), $C_{\alpha 2}$	-104910 N/rad (-411 lb/deg)
Track (width), T	1.52 m (4.9 ft)

Table 1 Taurus Nominal Vehicle Parameters for 2 DoF Linear Model

Note that K_{BS} is a brake steer scaling factor where normally $K_{BS} = 1$. The introduction of K_{BS} permits sensitivity to be examined. For this system, the steady state gain of $\dot{y}^{(s)}/\delta(s)$ from the final value theorem with a unit step input is $2.8 \text{ m/s}^2 \text{ deg}$ (0.29 g/deg). To have comparable gain as for the steering input δ , $K_{BS} = 3490$ is required for the brake steer. Factors such as brake system capability and tire adhesion will affect realization of braking forces, therefore, this K_{BS} is meant only for comparative illustration. As an aside, it is interesting to note that $\dot{y}^{(s)}/F_{BS}(s)$ is nonminimum phase for $U < 0$.

Figure 3 provides a comparison of achieving a lane change using the steering wheel and using brake steer. The three meter lane change maneuver is accomplished using a sinusoidal input to the front wheels of amplitude 0.5 degree (9 degrees at the steering wheel using a static steering mechanism gain of 18). The same lane change is done by a sinusoidal brake steer input of amplitude 1745 N ($K_{BS}/2$). The negative portions of each input create the change in path curvature needed to straighten the vehicle in the new lane. The positive half of the brake steer command, F_{BS} , is generated by braking on the right side of the vehicle. From simulation results, if the desired response frequency is doubled to 0.5 Hz, the steer effort is quadrupled; 2 degrees for wheel steer and 6980 N for brake steer. Again, the tire/road interface will constrain the achievable response in terms of achievable longitudinal braking forces. The next section on brake modeling describes brake steer limitations in terms of maximum possible braking forces.

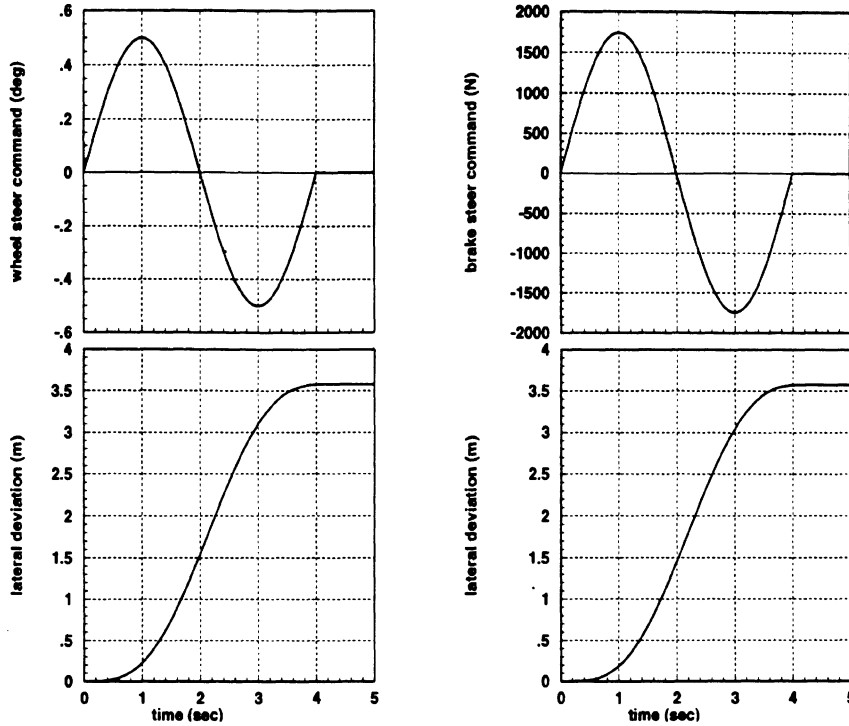


Figure 3 Comparison of Steering using Front Wheels vs. Brake Steering for 3 meter Lane Change Maneuver (hypothetical)

3 Brake System Model

Three aspects of a hydraulic ABS system have been modeled. They comprise (1) the saturation effect of the ABS controller in which brake pressure to the wheels is limited to prescribed wheel slip and acceleration, (2) a dynamic lag term introduced to represent the hydraulic system response to an input signal, and (3) the overall brake gain from hydraulic pressure to brake force (Figure 4). Rotational wheel effects are not considered.

The saturation effect of the ABS controller is the result of seeking a control performance where the maximum longitudinal braking force is imparted to the road without excessive longitudinal slip and wheel acceleration. The saturation threshold is assumed to be constant for a given vertical tire load and road condition using $F_{B,sat} \approx \mu F_Z$ where μ for the road is generally in the range [0.1, 1.0] (0.8 nominal) and vertical load is:

$$F_Z = \begin{cases} \frac{mgb}{2(a+b)} & \text{for one front corner (5160 N)} \\ \frac{mga}{2(a+b)} & \text{for one rear corner (3000 N)} \end{cases} \quad (8)$$

Additional assumptions are that the roll and pitch dynamic effects, as well as unsprung dynamics and passenger loading, are neglected. From the Taurus data using a nominal μ of 0.9, $F_{B,sat} = 4125N$ for one front corner and $2700N$ for one rear corner. The saturation effect is modeled as unity gain for $F_B < F_{B,Sat}$ such that

$$F_{B,limited} = \max(-F_{B,Sat}, \min(F_{B,desired}, +F_{B,Sat})) \quad (9)$$

As applied in this model, the saturation force is translated into a saturation *pressure* to reflect hydraulic system pressure and its effects at the wheel via the brake caliper.

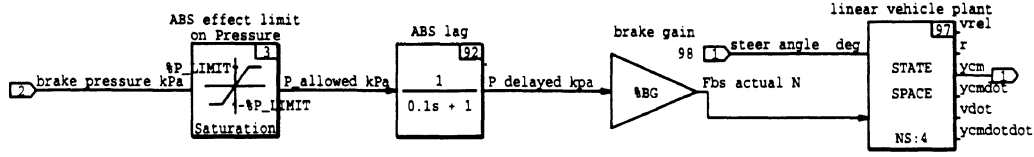


Figure 4 BSS System Model

The hydraulic system response is modeled as a first order lag with time constant $\tau_{ABS} = 0.1$ seconds [Bowman and Law 1993] such that

$$\dot{\bar{P}}_{hyd} = -\frac{1}{\tau_{ABS}} \bar{P}_{hyd} + \frac{1}{\tau_{ABS}} P_{hyd} \quad (10)$$

The model can be interpreted as the dynamic lag between a hydraulic pressure command P_{hyd} and the resulting brake pressure \bar{P}_{hyd} . Pure delay effects (computational) are not considered.

The overall brake gain has been represented as a scalar value based on the physical dimensions of the hydraulic system with the assumption that disk brakes are used. The brake gain describes the steady state gain from the desired hydraulic brake pressure in the disk brake caliper cylinder to an *ideal* longitudinal braking force applied at the tire/road interface. This ideal brake force, the result of hydraulic pressure desired by the BSS controller, may not be attainable due to road surface conditions and vertical tire loading. That does not present a problem since the ABS will saturate, and preclude the desired hydraulic pressure from creating an unachievable brake force. The static brake gain, K_B , used in this paper is $K_B = 0.58 \text{ N/kPa}$. It is derived from brake caliper and rotor dimensions.

The first order lag and brake gain model is only applicable when combined with the ABS saturation effect. The saturation element corrects brake pressures that would otherwise result in inaccurate or unachievable longitudinal brake forces at the tire.

4 Brake Steer Controller

Design of a suitable controller begins with combining the brake model and a corresponding brake pressure input to the vehicle model with lateral deviation output. Proportional plus derivative (PD) and state variable feedback controllers are examined. The addition of integral action i.e., proportional plus integral plus derivative (PID) to improve tracking ability is also considered.

4.1 Model Alterations

The system has been augmented with two free integrators to take inertial lateral acceleration, \ddot{y}_{CM} in (7), and generate lateral deviation in meters for the mass center. The states v and r are the relative lateral velocity and yaw rate from the original 2 DoF model, and y_{CM} and \dot{y}_{CM} are the augmented states representing the inertial lateral position and velocity. A fifth state corresponding to the pressure, \bar{P}_{hyd} , in the brake model is then prepended to the brake steer input to complete the linear plant model. The augmented scalar system representation with input F_{BS} and output y_{CM} takes the form:

$$\begin{Bmatrix} \dot{\bar{P}}_{hyd} \\ \dot{v} \\ \dot{r} \\ \dot{y}_{CM} \\ \ddot{y}_{CM} \end{Bmatrix} = \begin{bmatrix} -\frac{1}{\tau_{ABS}} & 0 & 0 & 0 & 0 \\ 0 & \frac{(C_{\alpha 2} + C_{\alpha 1})}{mu} & \frac{(-bC_{\alpha 2} + aC_{\alpha 1} - mu^2)}{mu} & 0 & 0 \\ T & \frac{(aC_{\alpha 1} - bC_{\alpha 2})}{I_z u} & \frac{(a^2 C_{\alpha 1} + b^2 C_{\alpha 2})}{I_z u} & 0 & 0 \\ 2I_z K_B & I_z u & I_z u & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{(C_{\alpha 2} + C_{\alpha 1})}{mu} & \frac{(-bC_{\alpha 2} + aC_{\alpha 1})}{mu} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \bar{P}_{hyd} \\ v \\ r \\ y_{CM} \\ \dot{y}_{CM} \end{Bmatrix} + \begin{Bmatrix} 1/\tau_{ABS} \\ 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix} F_{BS} \quad (11)$$

$$y_{CM} = [0 \ 0 \ 0 \ 1 \ 0] \begin{bmatrix} \bar{P}_{hyd} \\ v \\ r \\ y_{CM} \\ \dot{y}_{CM} \end{bmatrix}^T \quad (12)$$

The system described in (11)-(12) is used in the linear controller design, and does not include the ABS saturation term. It does include the brake state, and is therefore suitable for state feedback design as well as PD or PID design. Example block diagrams of a PD and a state feedback controller for BSS are shown in Figures 5 and 6.

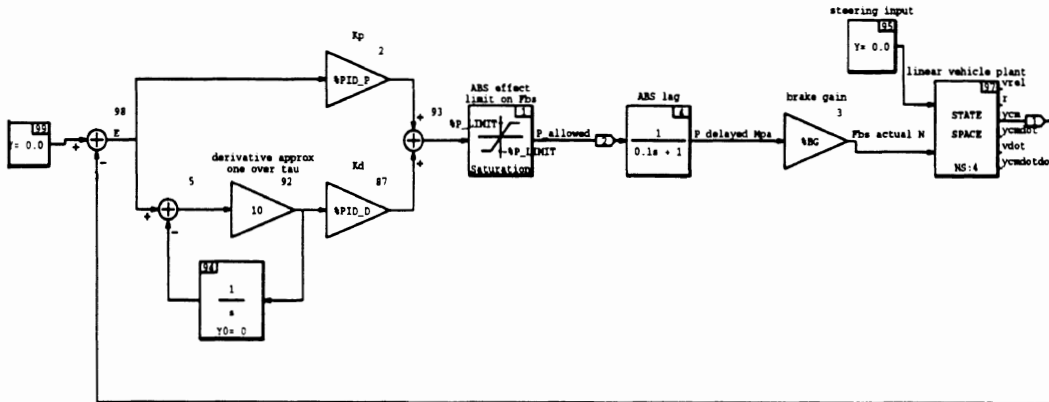


Figure 5 PD Controller

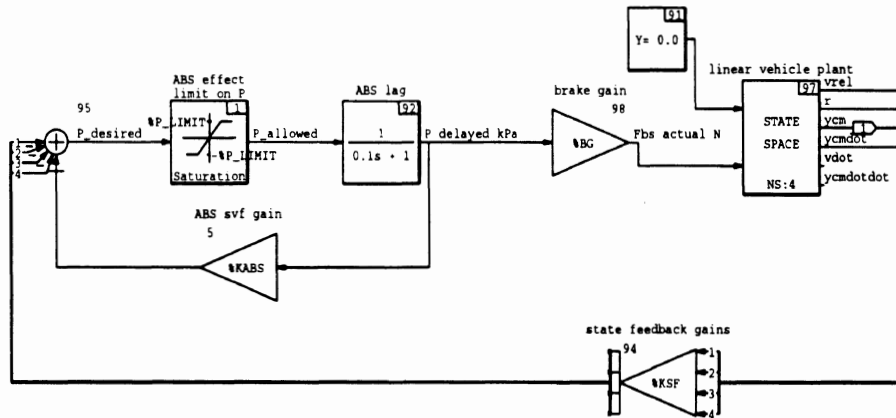


Figure 6 State Feedback Controller

The controller design problem is arranged to be single-input single-output (SISO) by considering the steering input δ as a disturbance input to the plant and lateral deviation as the only output as shown in Figure 4. This approach treats the model as a deviation model where the input steering angle is the difference between the actual steering input and required steering input as determined by knowledge of road curvature. The lateral deviation output reflects the error between the desired and actual paths. Road curvature is provided by a system outside of the brake steer system. This information is necessary to differentiate between the change in inertial lateral position as the result of following a curve and the lateral position as the result of steering error. An alternative to this approach is to adjust each state equation to the curvature or speed specified yaw rate, yaw angle and lateral acceleration [Peng and Tomizuka 1990a].

The deviation model is motivated by the nature of the regulator (i.e., the desire to drive the system states to zero). Without road curvature information, it would not be possible to feed back the system states that represent the deviation from the desired trajectory. For straight-line travel this is not an issue. Travel along a curve, on the other hand, would result in the states, whether measured or observed, reflecting not only lateral lane deviation, but the overall effect (e.g., centripetal acceleration) of negotiating the curved road section as well. Implementation of a compensator such as PD with a lateral deviation sensor as feedback does not require such consideration.

The effect of vehicle forward speed can be observed in simulations by modifying the linear vehicle model to incorporate a time varying vehicle speed effect. A non-constant speed model is formed by assuming vehicle forward momentum is dissipated by the applied braking force. The $\sum F_x$ term left out of (1a)-(1b) is reintroduced using

$$\sum F_x: \quad m\dot{u} = F_{BS} \quad (13)$$

Additional external forces are neglected, and it is assumed that the engine torque remains nearly constant in comparison to the braking torque at the wheels. The system of equations in (5) changes to

$$\begin{aligned} \dot{v} &= \frac{1}{mu} (C_{\alpha 2} + C_{\alpha 1})v + \frac{1}{mu} (-bC_{\alpha 2} + aC_{\alpha 1} - mu^2)r + \frac{-C_{\alpha 1}}{m} \delta \\ \dot{r} &= \frac{1}{I_z u} (aC_{\alpha 1} - bC_{\alpha 2})v + \frac{1}{I_z u} (a^2 C_{\alpha 2} + b^2 C_{\alpha 1})r + \frac{-aC_{\alpha 1}}{I_z} \delta + \frac{T}{2I_z} F_{BS} \\ \dot{u} &= \frac{F_{BS}}{m} \end{aligned} \quad (14)$$

4.2 Controller Design

The brake steer controller performance criteria are to achieve short settling time with minimal overshoot. Constraints on control effort are secondary criteria. Table 2 shows a summary of PD, pole placement and LQ design results for a 1 meter initial lateral deviation. The summary is based on results from the linear vehicle model combined with the ABS model with the nonlinear saturation element. Values within {}'s correspond to 0.5 meter initial lateral deviation results. The values are determined by analysis of the closed loop systems and from examination of model simulation output. Figures 5 and 6 are examples of block diagram representations of the PD and state feedback controllers.

	PD	Pole Placement	LQ
Design criteria	30°PM	ITAE pole pattern $\omega_n = 4 \frac{\text{rad}}{\text{sec}}$	$Q = \text{diag}[0 \ 0 \ 0 \ 10^4 \ 0]$ $R=0.00001$
Rise time (10-90%, sec)	1.1 {0.6}*	0.8 {0.5}	0.8 {0.5}
Settling time (2%, sec)	3.2 {3.1}	1.6 {1.3}	2.3 {1.8}
Overshoot (%)	5.8 {10}	1.7 {1.1}	6.0 {2.6}
Closed loop ev's	$[-12.9, -1.1 \pm j1.6,$ $-8.0 \pm j5.8, -1.1]$	$[-3.6, -2.3 \pm j2.1,$ $-1.5 \pm j5.2]$	$[-10, -2.5 \pm j2.9,$ $-6.1 \pm j4.8]$
Gains (10^3)	$K_p = 2.00,$ $K_d = 3.05$	$[-0.0011, -4.6, 14.5,$ $3.6, 3.0]$	$[0.00052, -10.2, 31.1,$ $31.6, 20.1]$

Table 2 Controller Design Summary for 1 m Disturbance *{...}'s for 0.5 m Disturbance

PD control design provides a satisfactory rise time and overshoot, however, settling time for the PD controller is greater than for the other two approaches. Increased performance with PD is constrained by the degree to which closed loop poles can be altered by the design. The pole placement and LQ methods performed better than PD in all categories. Figures 7a-c show simulation results using the linear vehicle model with the nonlinear brake model.

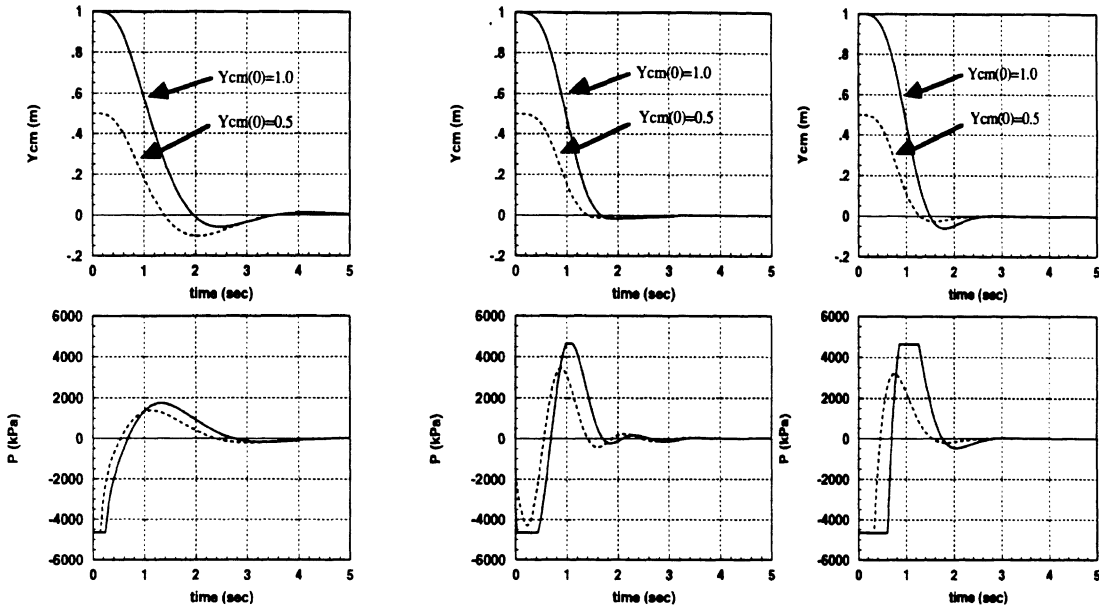


Figure 7a PD

Figure 7b Poleplace

Figure 7c LQ

PD, Poleplace and LQ Response to 1.0 and 0.5 m Initial Conditions

Figure 8a shows a comparison between controller responses for the linear vehicle model and a nonlinear vehicle (2 DoF bicycle) model with a Pacejka tire model [Asgari and Hrovat 1990] [Bakker *et al.* 1987]. The three pairs of traces closely overlay each other, indicating negligible difference between results (rise time, settling time, overshoot) obtained from the linear tire assumption and the nonlinear tire for each of the controllers. The linear and nonlinear traces are not explicitly labeled in Figure 8a since the overlap is close. A more graphically evident effect of the tire nonlinearity is seen in Figure 8b where step brake steer inputs are compared. A 2800 N step input results in a 11 per cent error in lateral acceleration when the linear tire model is used. The error diminishes as the response becomes more linear. Since the tested controllers operate in the nonlinear (ABS) range for brief periods, the effect of the nonlinearity is reduced.

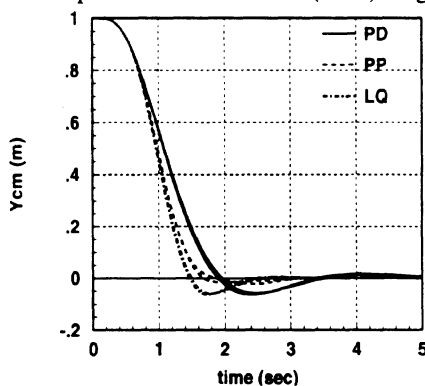


Figure 8a Controller Comparison with Linear and Nonlinear Tire Models

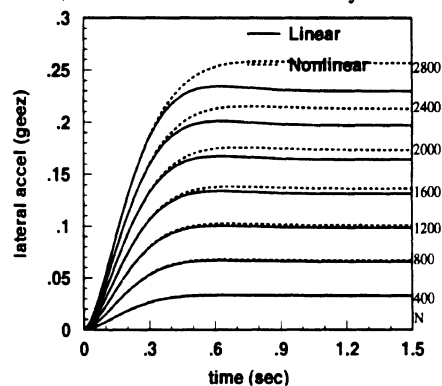


Figure 8b Step Responses of Linear and Nonlinear Tire Models (step size: 0:400:2800 N)

The pole placement design is based on the integral of the time absolute error - ITAE [Franklin and Powell 1986]. The LQ method is an output weighting form which places performance emphasis on the

lateral deviation, while placing a low penalty on control effort. Both methods assume a sufficient supply of control effort. Typically this is not feasible, however, due to the saturating effect of ABS, and it is possible to use ABS to achieve a bang-bang type response in brake hydraulic pressure as shown in Figures 7b and 7c. The saturation effect is shown by the clipped pressure waveform (threshold set in the brake model). The negative and positive values represent brake pressures for the respective left and right brakes on the vehicle. ABS saturation constrains control effort, and makes the response similar to the time-optimal bang-bang response possible for the state feedback methods. The PD methods do not have the necessary design freedom to achieve this type of control response.

The saturation effect also manifests itself in the rise and settling times of each controller design depending on the initial lateral deviation. Figure 7b illustrates the nonlinear effect of ABS saturation where the initial lateral deviation of 0.5 meters is not clipped. For deviations less than 0.5 meters the controller has a linear response. The nonlinear effect is also apparent from the performance criteria in Table 2. The values within curly brackets {}'s show that the controller performance is improved with the 0.5 meter initial deviation with the exception that overshoot is steadily increased in the PD controller as initial deviation is decreased. Performance would not change once in the linear range.

The controllers pertain to a nominal linear vehicle model where the tire cornering coefficient and vehicle speed are assumed constant. Previous work [Peng and Tomizuka 1990a] describes controller sensitivity to changes in vehicle speed and cornering coefficient, and has employed an adaptive approach using gain scheduling [Astrom 1987] to compensate for variations in these two parameters. The process involves prior determination of the PD gains (or state feedback gains) over a range of expected vehicle speed and cornering coefficients. Vehicle speed can be measured from existing onboard systems. The cornering coefficient, on the other hand, must be estimated. Peng and Tomizuka assume that $C_{\alpha 1} \approx C_{\alpha 2}$, which permits C_{α} to be estimated on-line from (5) (6) using measured vehicle data. The braking requirement of BSS will test the application of this assumption since it has been shown that longitudinal slip (as the result of braking) reduces the cornering coefficient of the braking wheel [Asgari and Hrovat 1992]. More elaborate parameter estimation methods have been applied that do not require $C_{\alpha 1} \approx C_{\alpha 2}$ [Bowman and Law 1993], however, on-line computational overhead would be increased, and the gain scheduling table would increase in complexity given the need to accommodate different front and rear cornering coefficients simultaneously.

Observer design requirements for the state feedback controllers combine the state estimation problem with the parameter estimation problem. It is possible to gain schedule controller gains, however, the observer requires a system model that reflects the time-varying vehicle speed and cornering coefficient. Thus, gain scheduling the observer is not practical given the changes in the system model during normal vehicle operation. A possible solution may be found using an extended Kalman filter which requires on-line computation of the filter gains and covariance matrix [Gelb 1989]. The problem of nonlinear observer design would need to be further addressed in a brake steer implementation.

4.3 Steering Wheel Disturbance Rejection

Thus far, the brake steer controller design evaluations have been made using an initial lateral deviation. If the steering wheel forcing function is considered the source of the lateral deviation as compared to the free response of a perturbed vehicle state, it becomes apparent that the controller has limitations as to the type of steering wheel error it can reject. An examination of system type using Figure 8 shows that the brake steer system is able to track a ramp reference input with zero steady state error ($\frac{\epsilon}{r} = \frac{1}{1+G_c G_2}$), but is unable to reject a step steering disturbance steady state error ($\frac{\epsilon}{\delta} = \frac{-G_1}{1+G_c G_2}$) without an integral term in the controller, G_c .

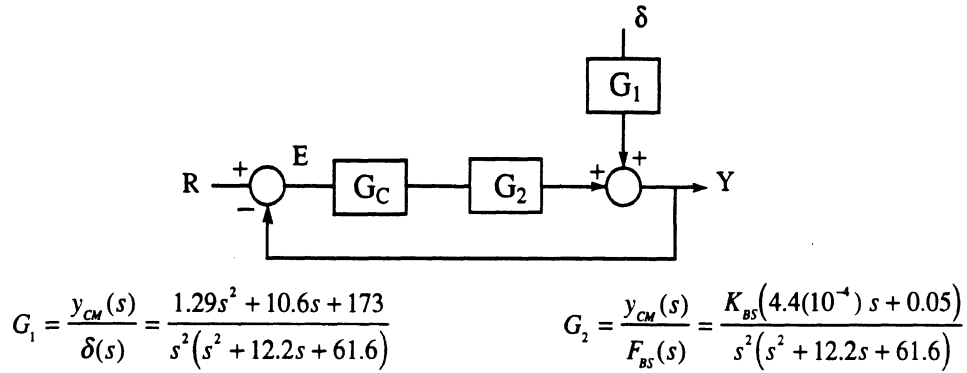


Figure 8 System Type Examination

But before integral action is considered, non-persistent steering wheel disturbances can be appreciably reduced using the controllers as they are currently designed. The systems can deal with a brief steering error as would be generated by a pulse movement of the steering wheel. Modeled as a pulse of 1 sec duration through a lag filter of 0.2 sec time constant, the steering error is applied to the BSS controller to see the limits of intervention authority. If maximum control effort is exerted to maintain heading, Figure 9 shows that a pulse steer error of 0.5 degree (9 degree at the steering wheel) can be controlled to approximately 0.16 m lateral deviation error using the ITAE controller. Left uncorrected, the pulse steer error results in 0.3 m lateral deviation at $t \approx 1.5$ second, and 0.8 m at $t \approx 2.0$ second. When this test is extended to include a constant steer angle error, there is a constant lateral deviation error proportional to steer angle error.

Addition of an integral term removes the lateral deviation error due to a constant steer input. The implementation is straightforward in the PID formulation using

$$G_C = \frac{u(s)}{e(s)} = K_P + \frac{K_I}{s} + K_D s \quad (15)$$

or via the state space formulation where an additional state is added to the system in (11)-(12) [Franklin and Powell 1986] to represent the integral of the output error using:

$$\begin{bmatrix} \dot{x}_I \\ \dot{\mathbf{x}} \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{C} \\ 0 & \mathbf{A} \end{bmatrix} \begin{bmatrix} x_I \\ \mathbf{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{B} \end{bmatrix} u \quad (16)$$

In both cases, the existence of the saturation element representing the effect of ABS presents the need for an anti-reset windup mechanism [Franklin and Powell 1986]. A typical implementation is to compare the relative magnitudes between desired brake pressure and actual pressure. If a saturation condition exists (desired > saturation limit), the integrator state is held to its last value before saturation, and this prevents the integrator state from building up. The anti-reset windup effect creates a nonlinear effect in addition to the ABS saturation element.

Figure 10 is a simulation of the 0.5 degree steering error extended over a period of 4.5 seconds using the augmented ITAE controller. Although the brakes do not saturate, the effect of the integral term can be seen during the period when the steer error is constant. The lateral deviation is reduced to a minimal amount using the integral error state, and does not experience the offset when no integral state is used.

Rejection of a constant steer input has implications regarding the ability/desirability of the controller to null out a persistent steer error. The addition of integral-type control adds the capability of controlling steering errors of a more persistent nature than those which can be addressed without the integral error state. This application remains constrained to the braking forces that can be developed, and must also consider excessive brake wear due to the duration of brake operation during persistent steer errors. In practice this would require brake temperature monitoring either through direct measurement or estimation.

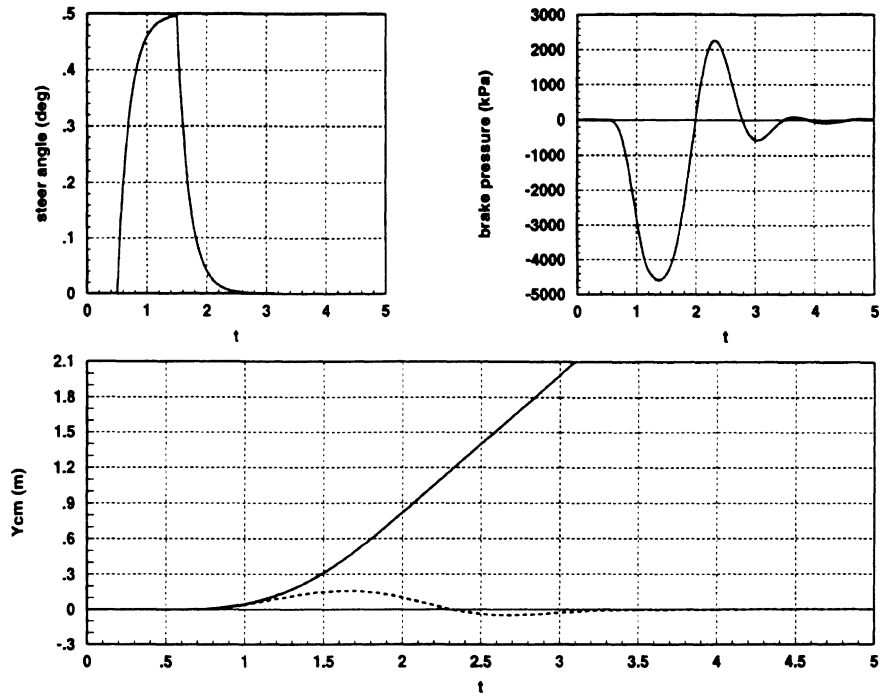


Figure 9 BSS Pulse-Type Steering Wheel Error Rejection (no integral state)

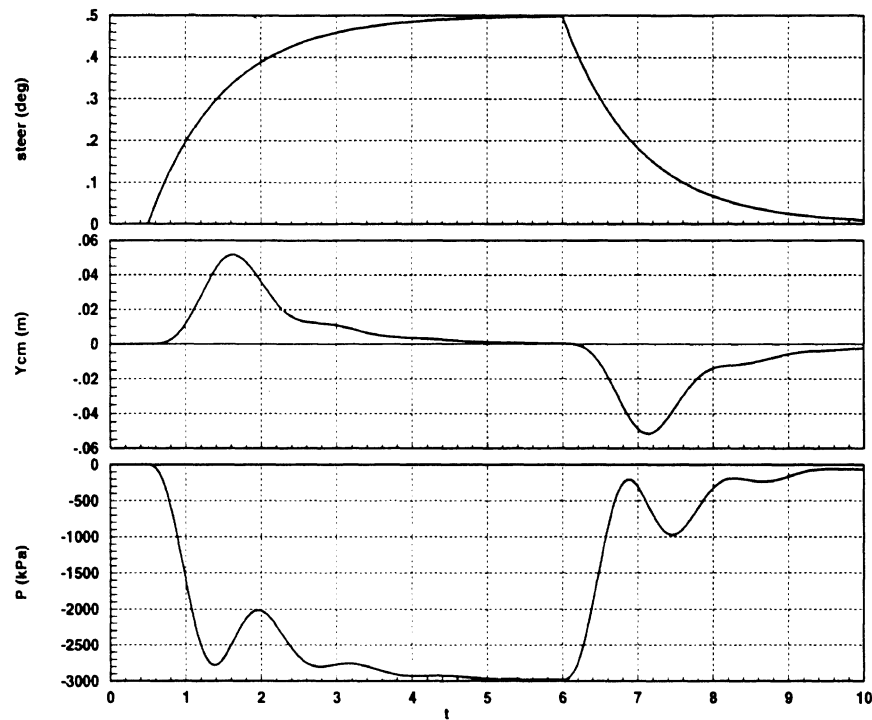


Figure 10 BSS Persistent Steering Wheel Error Rejection (with integral state)

4.4 Regions of Allowable Operation

So far we have considered a limit placed on differential force imposed by the braking forces which can be generated by the front and rear wheels on either the left or right sides. This is a longitudinal constraint imposed on differential braking.

It has been shown [Asgari 93] that there is also a bound on differential braking from the standpoint of steady state vehicle stability. The bound is illustrated in Figures 11(a) and (b) where vehicle trajectories

with and without brake steer are compared. In Figure 11(a), the vehicle is in a steady turn of approximately 0.5 deg on snow. The maximum achievable brake steer effort results in almost a doubling of “effective” steer angle. In Figure 11(b) the steer angle is increased to approximately 2.3 deg . In this case, the maximum achievable brake steer effort has a negligible incremental impact on the effective steer angle. The vehicle is able to perform a brake steer maneuver with the smaller steer angle (0.5 deg), but when the steer angle is increased (to 2.3 deg with all other conditions held constant) the vehicle can withstand only very minor brake steer efforts. If additional braking forces are applied, the vehicle can be forced to spin out.

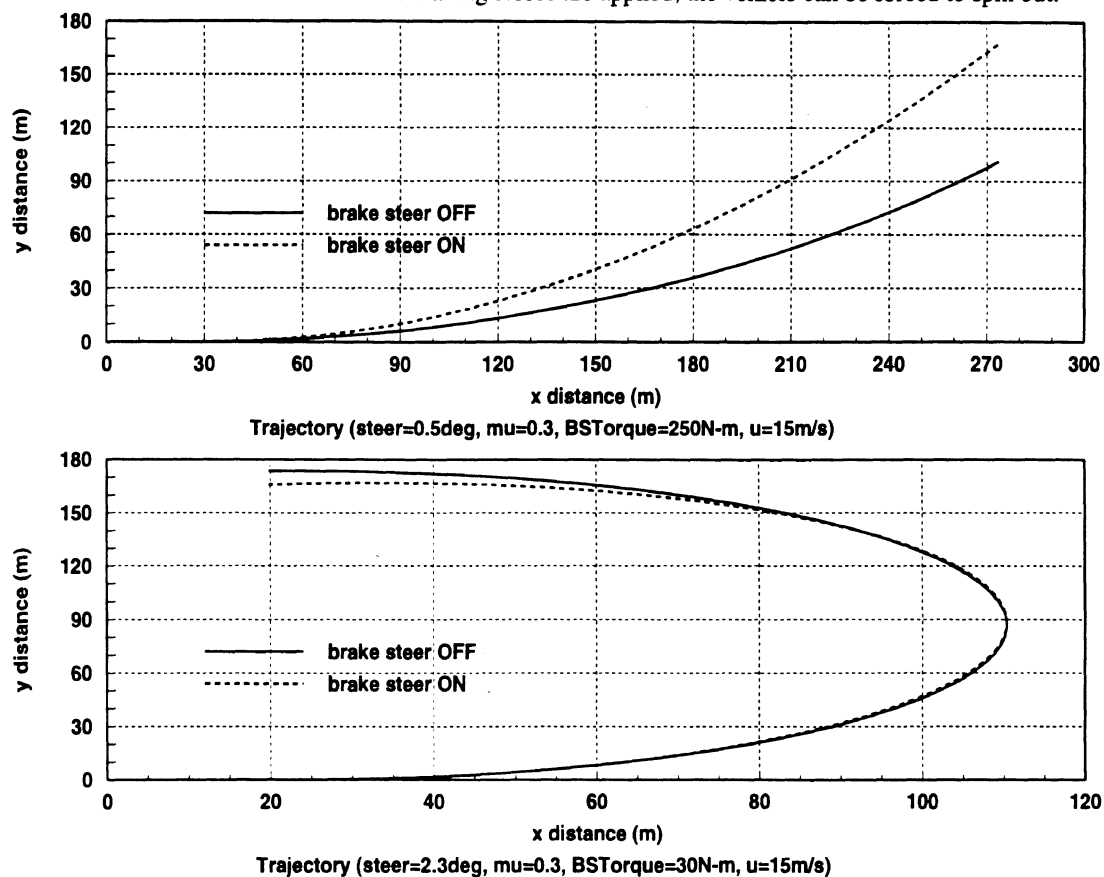


Figure 11 Vehicle Trajectories on Snow with and without Brake Steer

(a) $\delta = 0.5$, $\mu = 0.3$, $M_{BS} = 250 \text{ N} \cdot \text{m}$, and $u = 15 \text{ m/s}$

(b) $\delta = 2.3$, $\mu = 0.3$, $M_{BS} = 30 \text{ N} \cdot \text{m}$, and $u = 15 \text{ m/s}$

The spin out instability is caused by a rapid increase in rear side slip angle, α_2 resulting in reduced cornering force capability at the rear tires. A steady state turn in the direction of the desired brake steer turn reduces the amount of differential braking by “using up” containable yaw rate, resulting in reduced brake steer range of authority. This would occur when the desired effect is to add yaw rate to the steady turn when additional effective steer angle is required during a turn (driver not steering enough). If less effective steer angle is required, yaw rate from differential braking is subtracted (imposed yaw moment is opposite that of turn), and does not present the spin-out risk experienced when yaw moment is added.

Similarly, reduced road friction, μ , results in lower overall containable yaw rate and results in lower allowable brake steer force. The sensitivity of allowable differential brake steer forces to road friction and steady turns suggests that it would be helpful to determine the regions of allowable differential brake operation. Toward this goal, a brake steer region of operation can be defined as a function of μ and δ , as well as vehicle speed, u , based on known vehicle dynamics sensitivity to vehicle speed.

For this investigation, a 2 DoF bicycle model with a nonlinear Pacejka tire model is used. The parameters ranges are selected according to conditions that can be expected during highway driving. The following ranges are used: $0 \leq \delta \leq 2 \text{ degrees}$, $(50 \leq u \leq 70 \text{ mph})$ and $0.1 \leq \mu \leq 1.0$. The steer

ratio assumed is 18:1. The speed range was selected to reflect typical highway driving. The road friction coefficients range covers the gamut from ideal to polished ice. The stability criterion chosen is defined as the boundary where yaw acceleration, \dot{r} , is non-zero for steady state input conditions. Since this is a steady state examination, \dot{r} should be zero for constant δ , u and μ along with steady brake steer force.

The results (Figure 12) show that maximum differential braking is found at low speeds and zero steering angle, and the minimum is found at high speed and high steering angle. Steering angle has a greater effect on maximum differential braking than vehicle speed. Reduction in road friction serves to shift the region of operation down along the z axis ($F_{BS_{max}}$) until brake steer capability reduces to zero. This is especially the case with non-zero steer angle. Overall, the road coefficient of friction exerts the largest impact on allowable brake effort, with steer angle being next, and vehicle speed being the least influential of the selected set.

It is important to note that the differential brake force is a "generic" force, and has not been specified in terms of being generated from the front and/or rear brakes. Thus the yaw rate instability described here is independent of front or rear braking. Furthermore, the yaw instability is only seen in nonlinear tire model simulations. When the linear tire assumptions hold, the lateral stability is always maintained.

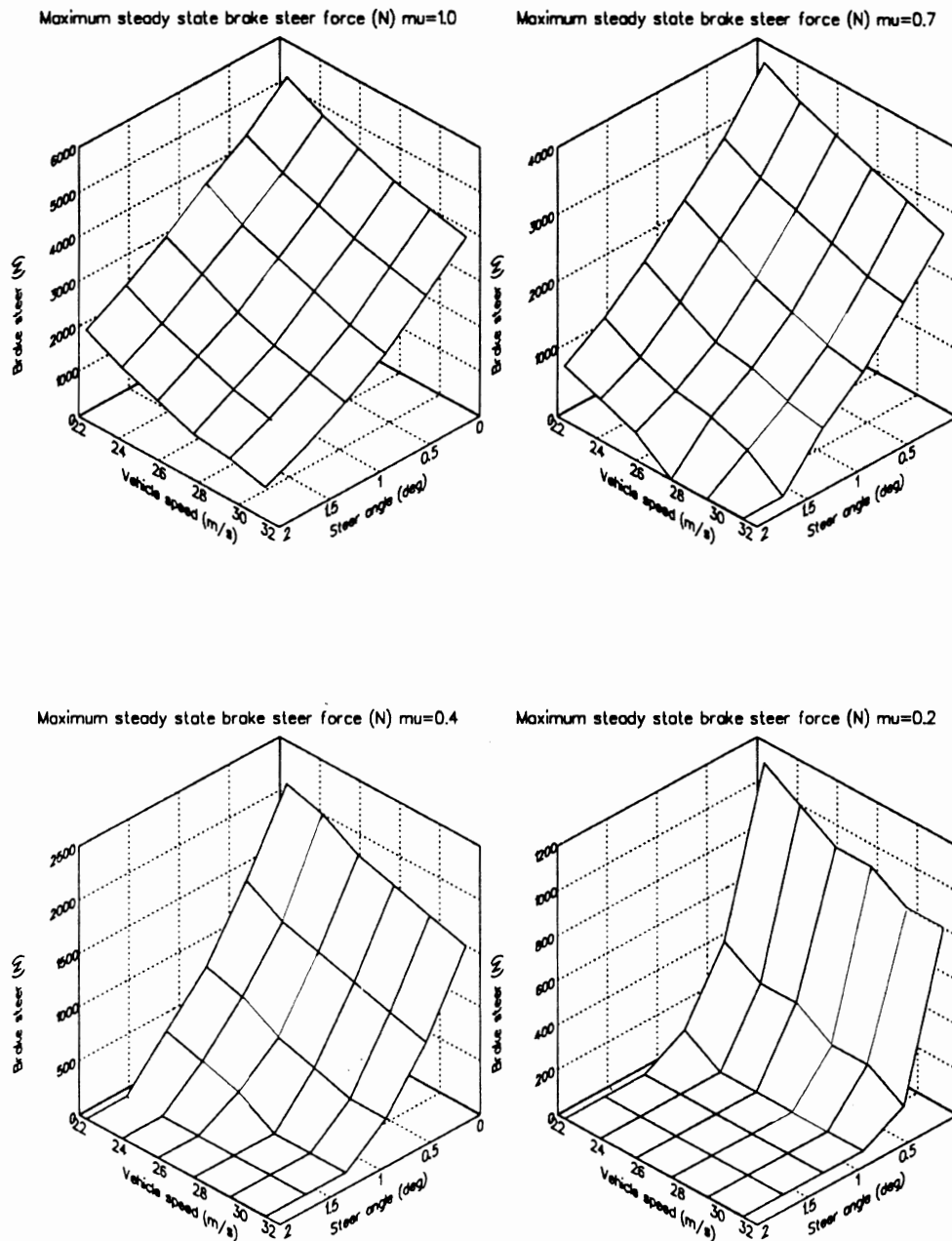


Figure 12 Maximum steady state brake forces
(a) $\mu=1.0$, (b) $\mu=0.7$, (c) $\mu=0.4$ and (d) $\mu=0.2$

One must keep in mind that the region of operation is determined using a steady state assumption where the steer angle and braking effort are constant inputs. Actual brake steer controller simulations, however, typically do not involve constant differential braking efforts. In this simulation, brake steer forces up to 5000N are exerted for a short period of time. This is almost twice the allowable steady state limit. The command signal quickly changes sign reflecting braking on the opposite side of the vehicle. In this case, negative brake steer (negative yaw moment, z axis into the page) acts to reduce the instability by taking yaw rate out of the vehicle. Therefore, it would seem to be conservative to constrain the controller to the allowable steady state limits.

A possible extension of the control region concept would be to schedule controller gains designed to ensure local stability for each δ_i, u_j and μ_k , where i, j and k represent increments in the ranges of steer angle, vehicle speed and road coefficient, respectively. Gain scheduling has been done for linear vehicle model parameter variation [Peng and Tomizuka 90a], but this new approach would involve linearizing the nonlinear vehicle at the same i, j and k inputs used to form the steady state control regions. But instead of computing a surface of maximum differential brake forces, a control design algorithm would be applied to each of the δ_i, u_j and μ_k linearized operating points to determine a surface of locally stable control gains. Then, δ, u and μ would be used as inputs to the gain scheduler with the continuing good input knowledge assumption. The end result is hypothesized to allow brake commands in excess of the steady state control region limits defined here, yet still ensure that the system is locally stable.

4.5 Additional Considerations

The BSS results in Sections 4.2 and 4.3 use only the left or right rear wheel (as limited by the brake system model) to generate the yaw moment for brake steer. The reason for that is two-fold. First, it makes for a more conservative design since rear wheels have less braking capability than the front, and second, it readily applies to rear wheel drive vehicles with traction control. Additional brake steer effect can be gained by applying brake force to both wheels on a given side. Front brakes are capable of considerably more (app. 70 per cent) braking force than the rear as the result of greater tire vertical load. Proportioning of fore/aft brake pressure during brake steer maneuvers could be addressed using a method similar to the proportioning used for ABS to ensure that both wheels on a given side approach saturation at an approximately even rate. However, the effect at the steering wheel of applying brake pressure to one front wheel and not the other front wheel have not been simulated.

So far, there has been no discussion of what happens to vehicle speed as the brakes are applied during a brake steer maneuver. One approach is to think of cruise control, and consider maintaining speed during the brake steer maneuver. A second approach is to avoid the situation where the engine increases power output to maintain speed, and either turn off cruise control if it is engaged or close the throttle directly (the latter assumes throttle-by-wire hardware capability). This second approach decelerates the vehicle using ABS-like action while providing steering intervention to prevent road departure.

From the control design standpoint, the change in vehicle speed affects the system around which the controller is designed. As speed is reduced the steady state gain $\dot{y}^{(s)}/r_{BS}^{(s)}$ is reduced. This may require adjustments in controller implementation in terms of gain scheduling, with regard to the ability to respond during an intervention maneuver. Efforts to make the linear design more robust would improve insensitivity to parameter changes due to deceleration. Figure 13 shows the amount of deceleration for the pole placement design as compared to the LQ design for the 1 meter initial lateral deviation. The change in vehicle speed is $-2.5 \frac{\text{m}}{\text{sec}}$ ($-5.6 \frac{\text{mile}}{\text{hour}}$). For persistent steer errors the decrease in vehicle speed is greater, and will increase controller sensitivity.

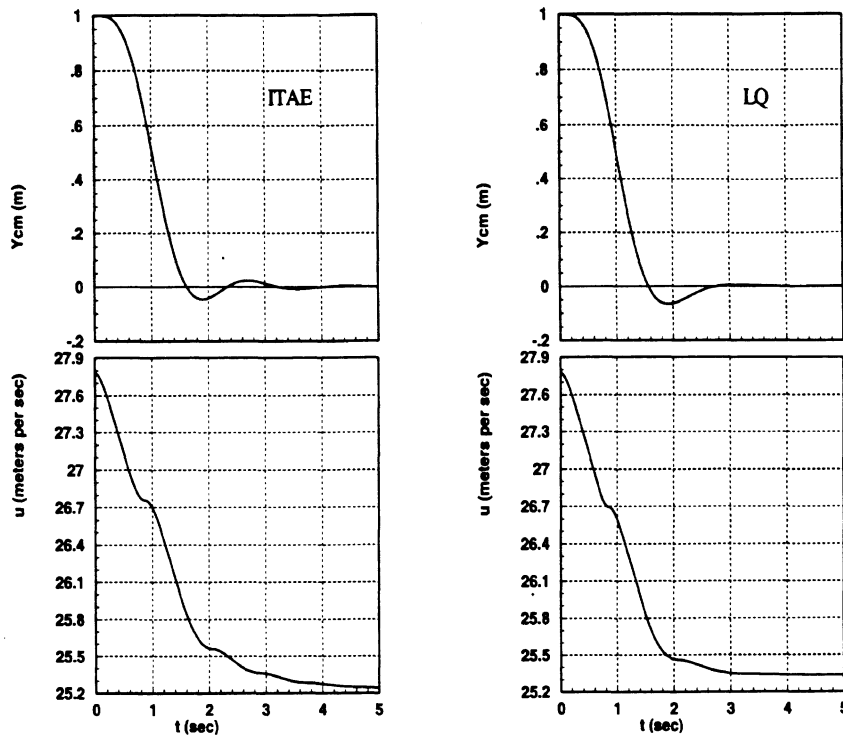


Figure 13 Effect of BSS on Vehicle Speed

Using the controllers described, the longitudinal acceleration \dot{u} experienced by the vehicle during brake steer maneuvers does not exceed 0.15 g's during maximum brake effort for the rear wheel only. This is a moderate, yet acceptable amount of deceleration. Addition of front and rear wheel braking, however, has the ability to exceed moderate braking levels. If front and rear braking is used, additional steps in controller design may be necessary to limit longitudinal deceleration. One possible solution may be to augment the performance criteria in an LQ approach similar to the index on ride quality (lateral acceleration) [Peng and Tomizuka 1990a]. This approach would permit longitudinal acceleration to be augmented into the system and incur a cost penalty which would serve to inhibit overly aggressive action that exceed performance requirements.

The deceleration that will be felt during brake steer will also serve as an unmistakable form of feedback to the driver, and may aid in increasing or calling attention to the driver alertness level.

5 Summary and Conclusions

A preliminary feasibility investigation of a steering intervention system based on differential braking has been demonstrated in simulation. A simple vehicle model for control design and simulation has been used. Controllers have been developed using linear design methods, and represent examples of both traditional PD/PID as well as state feedback designs. The simple 2 DoF vehicle model has been modified to include an ABS model containing a saturation effect and a lag. The simulation model shows the effect of brake steer commands where the brake actuator (ABS) is frequently saturated.

The response times of BSS are within useful bounds in terms of providing steering intervention for maneuvers to avoid road departure. BSS is limited by achievable tire braking forces, and will have reduced performance in poor road conditions much the same way normal steering is affected by road conditions.

The use of brakes to perform steering maneuvers also provides a level of capability that up until now could only be achieved by altering the steering system. This can be a costly approach, as compared to the functionality gained by less obtrusive modifications to the brake system for brake steer.

6 Future Work

As stated earlier, this represents a preliminary investigation of BSS. Open issues remain, and several have been either directly or indirectly referred to in the course of the paper. This section seeks to summarize work that will need to be addressed in subsequent levels of controller development and system implementation.

Existing issues in controller development involve parameter estimation and observer design. At the heart of these issues is the tire/road interface: Variations in the *effective* tire cornering coefficient for a given road condition are a nonlinear function of side slip angle and longitudinal tire slip. When road conditions alter the friction coefficient (an uncertain measurement), an additional dimension is added to the controller. Changes in vehicle speed add yet another dimension. The regulator/observer system, therefore, must be able to adjust gains (gain scheduling) to accommodate a wide range of operating conditions as well as be insensitive to model variations within each selected gain (robust design techniques). Additionally, the existence of allowable regions of steady state operation will require consideration to perform brake steer maneuvers in a stable manner. The results of previous work [Peng *et al.* 1992] will need to be incorporated into the BSS design and tailored to accommodate specific BSS operational requirements.

Implementation issues involve what it takes to get the BSS idea into a functional hardware prototype. Examples of differential braking hardware tests [Matsumoto *et al.* 1992] show that the ability to impose a moment on the vehicle is a workable concept, however, the context of steering by brakes alone is untested. BSS adds the additional requirement of knowing the vehicle lateral position with respect to the road edge. Upcoming road curvature is also used by BSS. In this paper, perfect road edge and road curvature sensors are assumed to be available. In terms of actuators, the ABS saturation effect has been only simulated. Actual tests on the ability of ABS to maintain lateral stability in the presence of BSS command will have to be examined. BSS will also require the brake actuator to control pressure from the saturation limit (normal ABS operation) down to near zero pressure (not normal ABS operation).

Operational issues must also be addressed. For example, a maximum acceptable amount of longitudinal deceleration may be incorporated into the design to prevent the vehicle from extreme brake maneuvers. Another issue is the interaction between the driver and BSS. Since the driver is not removed from the loop, balanced cooperation is important. This cooperation comes in the form of the system having the ability to correct an errant steer input as well as the ability of the system to transition from an intervention state back to an idled or disabled state. These are formidable issues, and there are certainly more to address.

References

1. Asgari, J. "Steady-State Analysis of 2D Vehicle for Integrated Brake Steer and Traction Control." *Ford Motor Company Internal Report, Dearborn MI*, 1993.
2. Asgari, J.; Hrovat, D. "2D Bicycle Type Vehicle Models for Analysis of Vehicle Handling and Ride." *Ford Motor Company Internal Report, Dearborn MI*, 1990.
3. Asgari, J.; Hrovat, D. "Steady State Analysis of 2D Vehicle Handling Model on Slippery Roads." *Ford Motor Company Internal Report, Dearborn MI*, 1992.
4. Astrom, K. "Adaptive Control". N.Y.: Addison-Wesley; 1987.
5. Bakker, E.; Nyborg, L.; Pacejka, H. "Tyre Modelling for Use in Vehicle Dynamics Studies." *SAE Technical Paper Series*. 1987; No. 870421.
6. Bowman, J.; Law, E. "A Feasibility Study of an Automotive Slip Control Braking System." *SAE Technical Paper*. 1993; No. 930762.
7. Franklin, G.; Powell, J. "Feedback Control Systems". : Addison-Wesley; 1986.
8. Gelb, A. "Applied Optimal Estimation". 11 ed. Cambridge, MA: MIT Press; 1989.
9. Hessburg, Thomas; Peng, Huei; Tomizuka, Masayoshi; Zhang, Wei-Bin; Kamei, Eiichi. "An experimental study on lateral control of a vehicle". *Proceedings of the American Control Conference*. American Automatic Control Council; 1991 1; 3; pp. 3084-3089.
10. Matsumoto, S.; Yamaguchi, H.; Inoue, H.; Yasuno, Y. "Improvement of Vehicle Dynamics Through Braking Force

- Distribution Control." *SAE Technical Paper*. 1992; No. 920645.
11. Nakazato, H.; Iwata, K.; Yoshioka, Y. "A New System for Independently Controlling Braking Force Between Inner and Outer Rear Wheels." *SAE Technical Paper*. 1989; No. 890835.
 12. Peng, H.; Hessburg, T.; Tomizuka, M.; Zhang, W. "A Theoretical and Experimental Study on Vehicle Lateral Control". *Proceedings of the American Control Conference*. American Automatic Control Council; 1992; 2; pp. 1738-1742.
 13. Peng, Huei; Tomizuka, Masayoshi. "Vehicle lateral control for highway automation". *Proceedings of the American Control Conference*. American Automatic Control Council; 1990 1; 1; pp. 788-794.
 14. Segel, L. "Analysis and Prediction of the Dynamic Behavior of Motor Vehicles". Ann Arbor, MI: Univ. of Michigan; 1992. 400 p. (Lecture Notes).
 15. Shladover, Steven E.; Desoer, Charles A.; Hedrick, J. Karl; Tomizuka, Masayoshi; Walrand, Jean; Zhang, Wei-Bin; McMahan, Donn H.; Peng, Huei; Sheikholeslam, Shahab; McKeown, Nick. "Automatic vehicle control developments in the PATH program." *IEEE Transactions on Vehicular Technology*. 1991 Feb; 40(1-1); p. 114-130.
 16. Tran, V. T. "Crosswind Feedforward Control - A Measure to Improve Vehicle Crosswind Characteristics". *12th IAVSD Symposium, Dynamics of Vehicles, Lyon France*. 1991.

Appendix I

"Lane Sensing and Path Prediction
for
Preventing Vehicle Road-
Departure Accidents"

C.-F. Lin, Ph.D. dissertation, University of
Michigan,
1995

**LANE SENSING AND PATH PREDICTION
FOR
PREVENTING VEHICLE ROAD-DEPARTURE ACCIDENTS**

by

Chiu-Feng Lin

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
1995

Doctoral Committee :

Professor A. Galip Ulsoy, Chairman
Assistant Professor Yili Liu
Assistant Professor Huei Peng
Associate Professor Prof. Jeffrey L. Stein

ABSTRACT

LANE SENSING AND PATH PREDICTION FOR PREVENTING VEHICLE ROAD-DEPARTURE ACCIDENTS

by

Chiu-Feng Lin

Chair : A. Galip Ulsoy

This dissertation focuses on the estimation of the time to lane crossing, a metric to assess the lane tracking margin of a vehicle. Characterization of the uncertainty in calculating the time to lane crossing is also studied. The result is used by a driver assistance system to prevent road-departure accidents. For time to lane crossing estimation, algorithms for down range road geometry perception and vehicle path prediction are also developed. Furthermore, uncertainty characterization for the vehicle path prediction and road geometry perception are also studied such that the uncertainty of the time to lane crossing can be characterized.

The time to lane crossing is obtained by first acquiring the intersection of the predicted vehicle path and the perceived road geometry and then estimating the time required for the vehicle to reach the intersection. The predicted vehicle path and the road geometry are expressed with polynomial equations. The uncertainty characterization for time to lane crossing estimation utilizes the uncertainty of the polynomial coefficients to assess the uncertainty of the acquired time to lane crossing.

To acquire down range road geometry, a least square curve fit and two Kalman filter algorithms are developed. The uncertainty characterization for the acquired geometry is developed based on Kalman filtering theory. For the vehicle path prediction, a two degree of freedom vehicle model is used. Front wheel steering angle and vehicle yaw rate are assumed to be measured. The lateral vehicle velocity and external disturbances acting

on the vehicle are estimated through an observer. Uncertainty characterization is associated with the equation for path projection; the measurement/estimation covariance for the vehicle dynamics and the front wheel steering angle are assumed.

The results show that the developed algorithms are successful for assessing the time to lane crossing for typical highway driving. Results also show that an accurate road geometry perception is more significant than an accurate path prediction. However, an accurate path prediction is also necessary to obtain a satisfactory time to lane crossing. Simulations also show that time to lane crossing seems to be a good metric for an active safety system, which is yet to be verified in the future study.

ACKNOWLEDGMENTS

I want to thank my God for leading me through my life here at the University of Michigan. He is always with me regardless of whether I am joyous or depressed. I want to praise Him and give all my honor to Him.

I want to give my special appreciation to my advisor, Dr. A. Galip Ulsoy. His requirement for me to have good writing and presentation skills inspires me to continuously improve. His guidance on my research helps me to do a decent job and to realize what an engineer should be. I also thank him for financially supporting me during my graduate study here.

I want to thank my wife, Hui-Hui, for her dedication to our family and for her support of my Ph.D. studies. She provided me with a comfortable home in which I regain my strength for everyday life. I also like to thank my parents, Chingyeng and Meichan, for their encouragement and their blessing throughout my academic career.

Finally, I would like to thank Dave LeBlanc for his help on my research. His suggestions helped me in solving many of my difficulties. I wish him have a great future career. I also would like to thank Paul Venhovens for his excellent work on the development of the CAPC simulator such that I had a wonderful tool to implement my algorithms.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	ii
LIST OF FIGURES	v
LIST OF TABLES	vii
CHAPTER	
I. INTRODUCTION	1
Motivation	1
Literature Review	3
Outline Of The Dissertation	11
Original Contributions	12
References	14
II. TIME TO LANE CROSSING CALCULATION AND CHARACTERIZATION OF ITS ASSOCIATED UNCERTAINTY	25
Abstract	25
Introduction	25
Time To Lane Crossing Calculation Algorithm	27
Time To Lane Crossing Frequency Distribution	29
Significant Factors For Time To Lane Crossing Accuracy	32
Characterization Of Time To Lane Crossing Uncertainty	34
Summary And Conclusion	36
References	38
III. VEHICLE PATH PREDICTION AND THE CHARACTERIZATION OF ITS ASSOCIATED UNCERTAINTY	44
Abstract	44
Introduction	44
Linearized Vehicle Path Projection Model	47
Kalman Filter For Vehicle Dynamics And External Disturbance Estimation	49
External Disturbance Characterization For Prediction	52
Predicted Path Uncertainty Characterization	55
Simulations And Discussions	57
Summary And Conclusions	61
References	63

IV. LANE GEOMETRY PERCEPTION AND THE CHARACTERIZATION OF Its Associated Uncertainty	70
Abstract	70
Introduction	71
Down Range Road Geometry Reconstruction	74
Perception Range Extension Algorithm	76
Least Square Curve Fitting Scheme For Lane Geometry Modeling	77
Kalman Filter For Lane Geometry Modeling	78
Characterization Of The Road Geometry Perception Uncertainty	82
Results And Discussion	83
Summary And Conclusions	90
References	93
V. SUMMARY AND CONCLUSIONS	105
Summary	105
Conclusions	108

LIST OF FIGURES

Figure		
I.1	Structure Of The Active Safety Control System	23
I.2	Elements For Time To Lane Crossing Calculation Error	24
I.3	Advantages Of Time To Lane Crossing For “Lane Tracking Margin” Assessment For A Vehicle	24
II.1	Elements for Time to Lane Crossing Calculation Error	40
II.2	Time To Lane Crossing Calculation Algorithm	40
II.3	Time to Lane Crossing (“Intoxicated” Driver Experiencing A Strong Side Wind)	41
II.4	Pixel Resolution, Vehicle Vibration, and Superelevation Effects On Time To Lane Crossing Accuracy	41
II.5	Model Simplification, Unknown External Disturbance, and Initial Condition Measurement Error Effects On Time To Lane Crossing Accuracy	42
II.6	Comparison Of The Monte Carlo Simulation And Model Prediction For Time To Lane Crossing Uncertainty	42
II.7	A Typical (a) Perceived Time To Lane Crossing And (b) Its Associated Uncertainty (“Intoxicated” Driver Experiencing A Strong Side Wind)	43
III.1	Geometrical Relation For A 2nd Order Polynomial Equation	64
III.2	Comparison Of The Nonlinear And Linear Equations For Vehicle Path Projection	64
III.3	Lateral Velocity And External Disturbances Estimation In A Large Superelevation And Strong Wind Condition	65
III.4	Lateral Velocity And External Disturbances Estimation In A Large Superelevation And Small Wind Condition	65
III.5	Lateral Velocity And External Disturbances Estimation In A No Superelevation And Small Wind Condition	66
III.6	Time To Lane Crossing Estimation Errors Under Different External Disturbances	66

III.7	Lateral Velocity And External Disturbances Estimation Under Significant Vehicle Vibration Effect	67
III.8	Lateral Velocity And External Disturbances Estimation In A Tight Curve Driving	67
III.9	Lateral Velocity And External Disturbances Estimation In A Straight Section Driving	68
III.10	Time To Lane Crossing Errors Under Different Effects	68
III.11	Lateral Velocity And External Disturbances Estimation Under Vehicle Forward Velocity Variation Effect	69
III.12	Comparison Of Different Disturbance Characterization Model For Future Value Prediction; (a) No Disturbance Estimation, (b) Piecewise Constant Model, (c) Linear Model, (d) Linear Model Plus Stochastic Model	69
IV.1	Lane Geometry Reconstruction Procedures	95
IV.2	Field Of View Constraint Of A Sensor	95
IV.3	Geometrical Relation For A 2nd Order Polynomial Equation	96
IV.4	Road Geometry Reconstruction Under A Good Road Condition; (a) 2nd Order Polynomial Equation, (b) 3rd Order Polynomial Equation	97
IV.5	Deviation Of The Perceived Road Geometry From the True Road Geometry Associated With The Road Geometry Reconstruction In Fig. IV.4	98
IV.6	Road Geometry Reconstruction Under A Rough Road Surface Condition; (a) 2nd Order Polynomial Equation, (b) 3rd Order Polynomial Equation	99
IV.7	Deviation Of The Perceived Road Geometry From the True Road Geometry Associated With The Road Geometry Reconstruction In Fig. IV.6	100
IV.8	Road Geometry Reconstruction Under A Superelevated Road Condition; (a) 2nd Order Polynomial Equation, (b) 3rd Order Polynomial Equation	101
IV.9	Deviation Of The Perceived Road Geometry From the True Road Geometry Associated With The Road Geometry	102

Reconstruction In Fig. IV.8

IV.10	Road Geometry Perception In A Superelevated Roadway	102
IV.11	Road Geometry Reconstruction With A 3rd Order Kalman Filter	103
IV.12	Road Geometry Reconstruction With A 4th Order Kalman Filter	103
IV.13	Road Geometry Perception Uncertainty Associated With A 3rd Order Kalman Filter	104
IV.14	Road Geometry Perception Uncertainty Associated With A 4th Order Kalman Filter	104

LIST OF TABLES

<u>Table</u>		
I.1	Examples of Vision Lane Sensing Systems	12
II.1	Improvement In Time To Lane Crossing With A Linear Interpolation Scheme	29
II.2	Time To Lane Crossing Bandwidth In Different Maneuvering Situations	31
III.1	Time To Lane Crossing Uncertainty Of Some Vehicle Path Prediction Uncertainties With A Nominal Time To Lane Crossing Of 3.0 sec	61
IV.1	Validation Of Zero Mean Assumption For Geometry Perception Of A 3rd Order Kalman Filter	88
IV.2	Validation Of Zero Mean Assumption For Geometry Perception Of A 4th Order Kalman Filter	88
IV.3	Comparison Of The True And Predicted Values Of The Coefficient Perception Errors Standard Deviation Of A 3rd Order Kalman Filter	89
IV.4	Comparison Of The True And Predicted Values Of The Coefficient Perception Errors Standard Deviation Of A 4th Order Kalman Filter	89
IV.5	Comparisons Of The TLC Uncertainty Prediction Between The Results With The True Geometry Perception Uncertainty And The Predicted Geometry Perception Uncertainty Associated With A 3rd Order Kalman Filter	90
IV.6	Comparisons Of The TLC Uncertainty Prediction Between The Results With The True Geometry Perception Uncertainty And The Predicted Geometry Perception Uncertainty Associated With A 4th Order Kalman Filter	90

CHAPTER I

INTRODUCTION

This dissertation focuses on the estimation of time to lane crossing, a metric that can be used to assess the lane tracking ability of a vehicle. Characterization of the uncertainty in the time to lane crossing assessment is also studied. To estimate time to lane crossing, and characterize its associated uncertainty, perception of the down range road geometry, prediction of the future vehicle path, and the associated uncertainties for these two curves are necessary.

Motivation

Among the traffic accidents in 1991 in the USA, road departure accidents account for 25% of the total. If fatalities are considered, road departure accidents account for almost one-third [The Hansen Report, 1992]. Such accidents usually involve a single vehicle which does not follow the roadway and runs into a roadside infrastructure, and they occur mainly due to driver impairment (e.g., drowsiness or drunkenness) rather than vehicle malfunction. Thus, it is postulated that an effective on-board system can help to prevent such accidents.

To solve this problem, two different system concepts have been proposed. The first one is a fully automatic vehicle control system on a restricted roadway. In such systems, the driver is expected to ride in a "hands-off" fashion and his/her role is to provide the high level commands to the controller such as the destination of the journey. Such a system concept has received attention from several researchers([Fenton, 1991], [Shladover, 1991], [Okuno, 1992], [Graefe, 1992]). Another system concept is an active

safety system. This system acts like a "co-pilot" for the vehicle with the human driver. It monitors driver performance, issues warning signals, intervenes, or takes over control of the vehicle when a dangerous situation is encountered. By intervention, or even only issuing a warning signal, it is expected that accidents can be significantly reduced [Parker, 1993].

Both the fully automatic control system and the active safety system seem technically promising. However, the active safety system is attractive because it can be developed as an evolutionary improvement to current vehicles where drivers are still the primary controller. Having the driver as the primary controller has two advantages : 1) most people feel safer when their lives are in their own hands, 2) it can keep the auto manufacturers from assuming too much liability [Dingle, 1993], which in turn enhances manufacturers' interest in developing such a system. Therefore, developing an active safety system has been adopted as the goal of our research group at the University of Michigan to solve the road departure accident problem [Huh *et.al.*, 1992].

A proposed active safety system structure is shown in Fig. I.1. This system uses the time to lane crossing (TLC) (i.e., the time required for vehicle to run off the roadway) as a metric for assessing the "lane tracking margin" of a vehicle. A "lane tracking margin" refers to any suitable metric appropriate for characterizing how well the vehicle motion tracks the lane that the vehicle occupies on the roadway. Based on the TLC and the assessment of the driver physical status, a decision is made regarding whether to issue a warning signal, to intervene, or to take over control of the vehicle.

Prior to the TLC calculation, two elements are necessary : (1) forward vehicle path prediction, and (2) down range road geometry perception. The TLC is obtained by first finding the intersection of the predicted forward vehicle path and the down range road boundary and then estimating the time span for the vehicle to reach the intersection. However, as shown in Figure I.2, due to the limitation of the on-board sensing system, the perceived roadway geometry deviates from the true geometry. This situation also occurs

with the predicted forward vehicle path, where the uncertainty is mainly caused by the unknown future external disturbances on the vehicle and the unknown future driver input. These errors subsequently lead to incorrect estimation of the time to lane crossing. Since these errors are not deterministic, they are modeled statistically and referred to as uncertainty.

This study focuses on the development of three elements of the active safety system: (1) an algorithm for geometrically modeling the previewed roadway, (2) an algorithm for future vehicle path prediction, and (3) an algorithm for time to lane crossing calculation. Since the decision module is expected to be sensitive to the TLC level, the uncertainty of TLC becomes an important element for the active safety system. Therefore, the associated uncertainty for each element is also characterized. For the lane geometry modeling module, lane marker locations along the lane boundary are assumed to be available from image processing (which is also true for the active safety system design). The other elements; the driver physical status assessment module, the decision module, and the vehicle lateral motion controller are being studied by another member in the road-departure active safety research team ([Pilutti, 1995a], [Pilutti, 1995b]).

Literature Review

Lane Tracking Margin Assessment

In the past, several different schemes have been utilized for “lane tracking margin” assessment. One is the lateral deviation of the vehicle from the desired trajectory (usually lane center) ([Shladover, 1991], [Fenton, 1991], [Dickmanns, 1988]). Beside lateral deviation, Kamada [1992] and Turk [1988] include the relative angle between vehicle heading and the tangent of the instantaneous desired trajectory. Furthermore, instead of using the instantaneous relative vehicle displacements, Okuno [1992] uses the predicted relative location of the vehicle from the lane boundary at a future time. A common feature in these schemes is that, from these methods, the decision module can only assess the lane

tracking margin at a specific time (i.e., either the current time or a future time) and will have no information about what happens at other times. This feature is satisfactory for an automated vehicle control system which is to follow a desired trajectory; the controller responds to the deviation from the desired trajectory at a specific point. However, such metrics may mislead an active safety system because a large deviation from the desired trajectory does not always mean a dangerous maneuver

A metric called Time to Lane Crossing (TLC) is more appropriate for an active safety system. This is a metric which describes the required time for the vehicle to reach either edge of the roadway. It is obtained by finding the intersection of the vehicle path and the road boundary and then estimating the time span for the vehicle to reach the intersection. Thus, compared to vehicle lateral deviation, TLC is less ambiguous for the decision module to use in judging the safety of the vehicle motion. It is postulated that a simpler rules for the decision module can be attained with TLC because it contains rich information. The TLC calculation explicitly includes all of the following factors which are important for assessing whether a vehicle might run off the road : 1) the down range roadway geometry, 2) the lateral position (and velocity) of the vehicle in the lane, 3) the vehicle heading angle (and yaw rate), 4) the vehicle forward velocity, and 5) driver's response. The advantages of TLC for "lane tracking margin" assessment over other metric's can also be realized through Fig. I.3. If the instantaneous lateral deviation is used as a metric for assessing "lane tracking margin", the situation in Fig. I.3(a) will be easy to trigger many false alarm, which is a common situation in highway driving because some people tend to drive closely to the lane edge. On the other hand, if the lateral deviation and heading angle are used only but neglect the driver's maneuver (steering angle) at the moment, the situation in Fig. I.3(b) may introduce false alarm. Finally, if the down range road geometry is neglected, a missed alarm will happen and cause an accident. Therefore, by using all the information as necessary, a more accurate decision can be made and by

combining these information into a metric, as TLC, a simpler rule base for decision module should be able to attain.

One way to locate the intersection between the perceived roadway and the projected vehicle path is to evaluate the future vehicle path in a “brute force” manner until a projected point lies outside of the road boundary [Godthelp, 1984]. Then, the time required for the vehicle to travel along the projected path to the intersection is the time to lane crossing. This method features simplicity and has been widely adopted. Other ways involve solving polynomial equations (i.e., finding the intersection of two polynomial equations). Such methods require additional logic to identify the correct solution and become complex when the order of the polynomial equation is large.

When the “brute force” method is used to calculate TLC, the time step associated with the discrete vehicle path projection introduces errors in the TLC. These errors can be improved by interpolation. An interpolation can be implemented by fitting a polynomial equation to the discrete vehicle locations in the vicinity of the intersection point. Then this polynomial equation and the road boundary are used to reduce the TLC error. There are basically three different polynomial fitting schemes: 1) the interpolating polynomial fit which requires that every data points lie on the polynomial equation, 2) the least square polynomial fit which has minimum variance between the data points and the curve, and 3) the minimax principle which minimizes the maximum distance between the data points and the curve [10].

There are several issues related to TLC which are important for active safety systems, but have not been investigated previously. The first issue is the bandwidth of TLC. An understanding of TLC bandwidth is important such that an appropriate sampling rate (i.e., TLC calculation speed) can be determined. Another issue is the accuracy of the TLC, which was noted but not characterized in previous TLC studies [Godthelp, 1984]. As shown in Figure I.2, there are errors in the predicted future vehicle path and the perceived road geometry, which subsequently cause errors in the TLC. The vehicle

prediction error mainly comes from unknown future external disturbances (e.g. wind gust and superelevation) and the driver steering input during the projection time. The lane geometry perception error is substantially affected by the vehicle vibration. For an automated vehicle control system, these errors are generally not significant within the range of interest and controllers are often able to reduce their effects. However, for the decision module to make a good decision, such errors must be explicitly characterized because the possible errors in the range of interest for the proposed active safety system (about 100 m ahead of the vehicle) is usually too large to be ignored.

Road Geometry Perception

The issue of down range roadway geometry perception has been widely studied using various sensing principles. For instance, there are systems using magnetic field, laser, radar, and computer vision principles. The main purpose in using different sensing principles is to develop a robust system for different environments (e.g., road surface, weather, shadows). Among the candidate sensing systems, computer vision has received the most attention due to its flexibility; it does not need substantial roadway infrastructure support (most of the time it only requires a painted white line). For this reason, computer vision is the sensing system chosen for the proposed active safety system. The following sections briefly introduce these techniques and the existing systems.

Magnetic field principle

The magnetic field principle has been used by the Partners for Advanced Transit and Highway (PATH) program at the University of California at Berkeley [Shladover, 1991], and the Automated Highway System (AHS) program at Ohio State University [Fenton, 1991]. The PATH program uses roadway embedded magnetic markers for vehicle position referencing and for coding the roadway geometry information. On the

other hand, the AHS program used roadway embedded magnetic wires for vehicle position referencing.

The magnetic reference and sensor system of the PATH program is an Intelligent Roadway Reference System [Shladover, 1991]. Permanent magnetic markers are discretely installed along the lane center line. The vehicle-borne sensing system acquires the pertinent information while passing over these markers, thus determining the vehicle deviation and road geometry that lies ahead. Lateral deviation of the vehicle from the center line can be expressed as a function of the magnetic field strength. Additionally, roadway information can be coded in a binary system which incorporates a series of magnetic markers so that the sensing system can describe the future roadway geometry.

The magnetic wire-reference systems of the AHS program [Olson, 1977] can only detect the vehicle's lateral deviation. Three different system configurations have been utilized. The first one is a two-wire amplitude sensing configuration and the second one is a one-wire amplitude sensing configuration. Both of these two configurations have a sensing system to measure the magnetic field strength of the vehicle location to calculate the lateral deviation and relate magnetic field strength to lateral deviation. The third configuration is different from the previous method. This method uses the phenomenon that when a sensor crosses the magnetic wire, there is a phase (sign) change of the magnetic field. The sensing system consists of a lateral array of equally spaced sensors. By counting the total number of phase (sign) changes, the number of sensors crossing the magnetic wire can be obtained, which in turn measures the lateral displacement of the vehicle. This configuration is less affected by conductive sheets under the surface of the guided path such as the steel-reinforcing mats and bridge structure materials.

Laser

Currently, only the Environmental Research Institute of Michigan (ERIM) Autonomous Land Vehicle (ALV) laser sensing system [Beyer, 1987] can be classified as a

laser-based lane sensing system. This system is installed on ALVIN, the Autonomous Land Vehicle at Marietta Denver Aerospace [Turk, 1988]. It uses a phase modulation method to calculate the range from the sensor to the target. However, alternative laser ranging methods are also suitable for this purpose. These ranging methods include time of flight, frequency modulation, and active triangulation. Detailed discussions of these methods can be found in [Fu, 1987] and [Everett, 1987].

The ERIM ALV consists of two parts : (i) an optical device for image acquisition, and (ii) a software system for image processing. The optical device includes a laser diode source operating in the near infrared region (820 nm) and a photodetector to receive reflected signals. The laser is scanned across the field of view using a nodding mirror and a rotating polygon mirror (80 deg horizontal and 30 deg vertical). In every direction, signals are sent out , hit the target, reflect back to the sensing system, and are received by the photodetector. Right after the photodetector receives a reflected signal, the traveling distance is calculated in real-time using a phase modulation method. This range is assigned to an image pixel which corresponds to the direction. The image size is a 256 by 64 array and the range is digitized using an 8 bit binary system. This is called a range image in which every pixel is assigned a range value instead of an intensity value.

The next step is to extract roadway configuration information from the range image. There are four substeps required to complete this step. The first substep is to remove unreliable range data from the image and to smooth out noise by using edge preserving morphological smoothers. The second substep is to find and label the ambiguity range, which is the range larger than half the signal wave length. The third substep is to transform the forward looking range image into a top-down view plan map. Then, a simple rule is used: "the road is a flatter and smoother 3-D surface than the NOT road", to extract the roadway configuration to the surrounding environment. This system does not actually generate a model for the road geometry.

Radar

A radar is similar to a laser in the sense that both of them utilize electromagnetic waves. However, radar is not as collimated as a laser and its operating frequency range is a lot smaller than a laser. It can also be utilized to measure the range from an object to the sensor. The ranging methods suitable for radar are time of flight, phase modulation, and frequency modulation. However, only frequency modulation has been applied to lane sensing [Mayhan, 1982]. Furthermore, this system can detect the lateral deviation of the vehicle, but not the previewed roadway.

In the frequency modulation method, the transmitted optical frequency is repetitively swept linearly between $[v+dv/2, v-dv/2]$ to create a total frequency deviation of dv during the period $1/f$ (f is the linear sweep modulation frequency), the reflected signal can be mixed coherently with a reference signal at the detector to create a beat frequency signal that depends on the range to the object R . This process is known as FM coherent heterodyne detection.

Computer vision

The process of down range road geometry perception with a vision system can be typically divided into three stages. The first stage extracts the desired object from the image (usually the painted white line or the road edge). The second stage calculates several imaginary lane marker locations along the painted white line (or the road edge). These two stages are usually referred as image processing. In the final stage the acquired lane marker locations are used to generate a model for the down range road geometry. Studies have been focused on these three stages.

For the first stage, the extraction of the desired object is difficult when the contrast between the object and its environment is not sharp ([Thorpe, 1992], [Kenue, 1989], [Waxman, 1985], [Liou, 1986]). A monochrome camera is usually used for its low cost and its ability to handle the well-conditioned (straight, evenly illuminated, well marked)

roadway ([Kenue, 1989], [Okuno, 1992]). On the other hand, a monochrome camera system usually has difficulty in detecting an ill-illuminated or ill-conditioned roadway. The RGB camera system can give better results in this kind of situation ([Thorpe, 1992], [Turk, 1988]). Processing speed is also crucial, particularly for a high speed automated vehicle [Graefe, 1992]. Thus, different computer system have been utilized for the processing.

The problem associated with the second stage is generally called the inverse optics problem, which arises because the image is in a 2-D plane and the acquired scene is in a 3-D space; the mapping between them is essentially one to multiple ([Poggio, 1985], [Fu, 1987]). Many factors can cause inaccurate perception of the 3-D location of a spatial object. Two substantial factors are the super-elevation of a roadway and vehicle vibration. A typical method for solving the inverse optics problem is the flat-earth assumption which eliminates the extra degree of freedom [Turk, 1988]. Binocular camera system has also been used to solve this problem; however, the image processing is time consuming [Jain, 1992]. People also impose constraints on the perceived lane marker locations to improve the perception accuracy ([Waxman, 1985], [Turk, 1988], [DeMenthon, 1987]), which is usually called the forward-geometry solution. Another method to solve this problem is called the inverse-geometry solution which uses the location within the image plane of a point whose 3-D location is known to solve for the coordinates of the other objects ([Turk, 1988],[Chatila, 1985]).

For road geometry modeling, two schemes are typically considered; (1) least square curve fitting (which includes Hough Transform [Kenue, 1989]) and (2) Kalman filtering. The down range road geometry is usually expressed by a polynomial equation or a set of polynomial equations. This is because a polynomial equation contains enough information for the automated control system (generally the curvature, lateral deviation, and heading angle are needed); also, it is natural to describe a curve with a polynomial equation. The difference is the order of the polynomial equation; for example, Kenue [1989] and Kamada [1992] use a first order equation, Thorpe [1992] uses a second order equation and

Dickmanns [1988] and Franke [1991] use a third order equation. The equation order is chosen based on the adequacy of describing the previewed roadway in the field of view of interest; a large area generally requires a higher order. Both the least square curve fitting and Kalman Filtering approaches find an optimal polynomial equation (with a pre-chosen order) which minimizes the variance between the equation and the acquired lane marker locations. There are two ways to do least square curve fitting. One way is to simply fit an polynomial equation to the lane marker array contained in the current image. The other utilizes the knowledge of the vehicle kinematics to transform all the lane marker locations from several images to a global reference frame to create a lane marker array which contains historical information for curve fitting ([Kenue, 1989], [Thorpe, 1992]). In a sense, a Kalman filter is similar to the latter least square curve fitting scheme; it also utilizes the information from the previous images and the vehicle kinematics [Dickmanns, 1988]. However, the Kalman Filter approach uses a recursive algorithm which is more time and memory efficient than the second least square curve fitting scheme([Dickmanns, 1988], [Broida, 1986], [Rives, 1986], [Kriegman, 1989]). Previously, no comparison between the least square curve fit using only current image information and the Kalman filter schemes has been conducted. Furthermore, Kalman filter were tested on typical highways; its performance for some irregular road geometry and surfaces than typical highway conditions have never been discussed.

Examples of existing lane sensing systems using computer vision are summarized in Table I.1. Some of the listed systems are not directly for detecting outdoor roadway geometry, but perform similar tasks.

Table I.1 Examples of Vision Lane Sensing Systems

<i>Systems</i>	<i>Optical Device</i>	<i>Computer</i>	<i>Lane Marker 3-D Locating</i>	<i>Curve Equation Generator</i>	<i>Reference</i>
VITS	RGB monocular	no description	inverse geometry solution and forward geometry solution	do not generate curve equation	[Turk, 1988]
Vision System for MARS	monochrome binocular	VAX 11/750	no description	do not generate curve equation	[Kriegman, 1989]
Vision System for VaMoRs	monochrome monocular	BVV2 (a multi-microcomputer system)	flat earth assumption	Kalman filter	[Graefe, 1992]
SCARF	RGB monocular	Sun 3/160	forward geometry solution	Hough transform	[Thorpe, 1988]
YARF	RGB monocular	Sun 3/160	forward geometry solution	least-square fit using	[Thorpe, 1992]
LaneLok I	monochrome monocular	Digital VAX 8600	flat earth assumption	Hough transform	[Kenue, 1989a]
LaneLok II	monochrome monocular	Digital VAX 8600	flat earth assumption	least-square line fitting	[Kenue, 1989b]
A Vision System by Kamada et.al.	RGB monocular	Intel 80286 CPU	flat earth assumption	Hough transform	[Kamada, 1992]
Vision System for MOVER-2	monochrome monocular	NEC PC 9801 LS (80386 CPU)	flat earth assumption	no description	[Okuno, 1992]

One important issue for road geometry sensing is the field of view of a sensing system. The camera field of view directly influences the robustness of the vehicle control system. This is because there are certain situations in which the object is outside the field of view of a camera, which leads to loss of information to the controller and subsequent robustness problems. Usually, a multi-far-field-sensor system is implemented to extend the perception area ([Graefe, 1992], [Tsugawa, 1984]). For a single-far-field-sensor system, an algorithm which utilizes the information located in the field of view of the camera is necessary to extend the range of the perceived road geometry.

Besides down range road geometry acquisition, an explicit form of the perceived road geometry uncertainty range is also necessary for the uncertainty characterization of TLC. This issue has never been studied previously. The perception error in the range of interest of an autonomous vehicle system is usually small, and its effect on the system performance can be successfully reduced by the control system. Most of the studies on perception uncertainty are at the level of characterizing the uncertainty for a spatial point ([Kriegman, 1989],[Matthies, 1987]), as needed by a Kalman Filter. Such studies are for the guidance of a mobile robot in an indoor environment; therefore, the uncertainty characterization does not consider vehicle pitch and roll effects since they are relatively small in this situation. Furthermore, these studies on the uncertainty characterization are associated with the computer vision process. No study for other sensing principles have been conducted. In such studies, the uncertainty range is usually modeled in statistical form and typically as Gaussian [Durrant-Whyte, 1988]. Other methods for uncertainty characterization are scalar weights [Moravec, 1980] and uncertainty manifolds [Brooks, 1985]. The Gaussian Distribution is widely used because of its convenience for analysis.

Vehicle Path Prediction

Accurate path prediction requires accurate initial conditions (i.e., vehicle lateral velocity, yaw rate, and front wheel steering angle), vehicle model, knowledge of future maneuvering, and future external disturbances. Usually, yaw rate and front wheel steering angle can be measured; however, a considerable measurement error may appear in the steering angle signal. On the other hand, lateral velocity, future maneuvers and external disturbances are usually unknown. To attain satisfactory path prediction, effective algorithms must be developed to estimate these variables. For most of the previous studies, the external disturbances are simply omitted. As to the future maneuver, some people simply ignore the possible future steering variation by assuming a fixed steering angle for path projection (Godthelp [1984] and Okuno [1992]). Many people have

developed different models to predict the possible maneuver of a human driver (e.g., the preview-predictor models ([Johnson, 1965], [Sheridan, 1964], [MacAdam, 1988]), the describing function models [McRuer, 1975]). Finally, for the projection model, Hayward [1972] uses a kinematics projection (i.e., use the current velocities and accelerations and assume these variables remain constant in the projection period). Godthelp [1984] and Okuno [1992] consider the dynamics of the vehicle by utilizing the steady state path curvature gain which relates the curvature of the vehicle path in steady state motion with the front wheel steering input [Gillespie, 1992]. Transient dynamics are omitted in these studies for the purpose of simplicity and such an approach seems to work reasonably well in mild maneuvers (i.e., straight road driving with mild external disturbances). In a cornering maneuver, or a maneuver involving a severe external disturbance, the transient dynamics must be considered. To consider transient dynamics, a vehicle dynamics model is used. A commonly used vehicle dynamics model is a 2 Degree of Freedom (DoF) vehicle model [Gillespie, 1992]), which only considers lateral and yaw motion. Typically, this model is utilized for lateral motion controller design ([Fenton, 1991], [Shladover, 1991]) and driver modeling ([MacAdam, 1988], [Hess, 1990]). However, it is not adequate for many detailed vehicle dynamics studies. Usually extra DoF or nonlinearity are added to this simple vehicle model. A common addition is the roll motion, resulting in a 3 DoF linear vehicle dynamic model. Furthermore, nonlinear vehicle models have been derived to fully represent the vehicle dynamics in particular studies ([Allen, 1993], [Shladover, 1991], [Vedamuthu, 1993]). On the other hand, a nonlinear vehicle dynamic model (including, for example, tire dynamics [Gillespie, 1992] or lateral load distribution [Clover, 1993]) is used for simulation and validation purposes.

The estimation of the lateral velocity and external disturbances can be attained through an observer; a state and unknown disturbance estimation problem. This problem was first discussed by Johnson [1971] in which the disturbances are characterized with differential equations (which means that the disturbances are combinations of some known

frequency but unknown amplitude sinusoidal/exponential functions). Then these noise dynamics models are augmented into the system dynamics model to form a new model. Finally, observer technique like pole placement method [Franklin, 1990] is designed to obtain the magnitude of the external disturbance and the system dynamics states. Similar scheme was discussed by Gourishankar [1977], in which the disturbance is expressed as a polynomial function and necessary and sufficient conditions were derived for such a model. The same technique was extended by Johnson [1984] to accommodate disturbances which contains both waveform-type and noise-type components. The noise-type disturbances are modeled as the output of a filter with random white noise as the input. For observer design, Kalman filter technique [Brown, 1983] is used to accommodate for noisy measurements. The same modeling technique were also used by Park [1987] and the subsequent researchers ([Hou, 1992], [Maquin, 1994]). However, for observer design, they apply singular value decomposition methods, in which the state equation is decomposed into two subsystems: one is unknown input dependent and the other is not. Through algebraic manipulation, the estimated states of the unknown input free subsystem and the measurement vector are used to reconstruct the other states and the unknown inputs. Because the results depend on the first derivative of the measurements, for a system with noisy measurement, these methods may not be applicable.

Beside the estimation of external disturbances, models to characterize these disturbances are necessary for prediction. The models for disturbances are various, and include sinusoidal, step, and impulse functions as well as stochastic models [Aström, 1992]. Since two substantial components of the disturbances acting on vehicle come from superelevation of the roadway and wind, stochastic models seem most appropriate to account for the characteristics of the wind force.

Outline Of The Dissertation

The main theme of this dissertation is to develop algorithms for time to lane crossing calculation and its associated uncertainty characterization. To accomplish this goal, two algorithms for (i) the down range road geometry, and (ii) vehicle path prediction are also developed. Thus, this dissertation is divided into three portions; 1) time to lane crossing calculation algorithm and the characterization of its associated uncertainty assuming the road geometry and vehicle path and their associated uncertainties are available, 2) vehicle path prediction and the characterization of the associated uncertainty, and 3) down range road geometry perception and the associated uncertainty. Three subjects are discussed in three subsequent chapters in the order as described; each is in the format of a paper for a conference ([Lin and Ulsoy, 1995a], [Lin and Ulsoy, 1995b], and [Lin et.al, 1995]) and will also be submitted to appropriate journals.

Chapter II discusses the time to lane crossing calculation algorithm and the characterization of its associated uncertainty. It also contains studies of some characteristics of time to lane crossing. Chapter III introduces the vehicle path prediction algorithm and the characterization algorithm of its associated uncertainty. Simulations are included and discussed regarding the performances of these algorithm in different driving environments. Chapter IV discusses the down range road geometry perception and the characterization of its associated uncertainty. It contains the introduction of the algorithms and simulation results in several different situations. Finally, chapter V summaries the studies discussed in this dissertation and presents conclusions.

Original Contributions

This section summarizes the original contributions of this research.

The following are the original contributions regarding the time to lane crossing calculation (Chapter II) :

- 1) An algorithm to characterize the uncertainty in the time to lane crossing estimate, which is able to predict the standard deviation of the TLC uncertainty with $\pm 10\%$ accuracy for typical highway driving is developed.
- 2) Time to lane crossing bandwidth for typical highway driving is identified as 2.5 Hz.
- 3) A linear interpolation scheme is developed to reduce the time to lane crossing error caused by the discrete time span for vehicle path projection.

The following are the original contributions regarding the vehicle path prediction (Chapter III) :

- 1) A Kalman filter is developed to simultaneously estimate the external disturbances acting on the vehicle and the vehicle dynamics. It significantly improves the results of the time to lane crossing calculation.
- 2) A piecewise constant model, to characterize external disturbances for path prediction, is shown to be adequate for the TLC calculation.
- 3) An uncertainty characterization algorithm is developed for the predicted vehicle path.
- 4) A fixed Kalman filter gain is shown to be adequate for disturbances estimation for typical highway driving expect when the vehicle encounters a tight curve,
- 5) A linear path projection model, which includes a two degree of freedom vehicle dynamics model, is shown to be adequate for path projection for the purpose of time to lane crossing calculation.

The following are the original contributions regarding the down range road geometry perception (Chapter IV):

- 1) It is found that, in general, a Kalman filter has a better performance for road geometry reconstruction than a least square curve fit because it successfully filters out the perception noise; however, its performance is worse than a least square curve fit in the transition period because of filtering lags.
- 2) A Kalman filter can be developed to recover the road curvature on a significant superelevation, the reason being that it uses the knowledge of the vehicle kinematics and recursively averages the perceived road geometry.
- 3) It is shown that a fourth order Kalman filter (which models the road geometry with 3rd order polynomial equation) is generally better than a 3rd order Kalman filter (which models the road geometry with 2nd order polynomial equation) because : i) a 4th order Kalman filter allows the curvature to vary along the curve, and ii) a 3rd order polynomial equation have a better fit to the acquired lane marker locations than a 2nd order polynomial equation.
- 4) An uncertainty characterization scheme is developed for the perceived down range road geometry and is able to predict the standard deviation of the perception error in a good accuracy.
- 5) An algorithm is developed to resolve the field of view problem of a single-far-field sensor lane sensing system.

References

- Allen R.W, Rosenthal T.J., "A Computer Simulation Analysis Of Safety Critical Maneuvers For Assessing Ground Vehicle Dynamic Stability", *Society of Automotive Engineering Paper No. 930760*, 1993.
- Beyer J., Jacob S. C., Pont F., "Autonomous Vehicle Guidance Using Laser Range Imagery", *Proc. SPIE Mobile Robots II*, Vol. 852, pp. 34-43, 1987.
- Brooks R.A., "Visual Map Making For A Mobile Robot", in *IEEE Int. Conf. on Robotics and Automation*, 1985.
- Broida T.J., Chellappa R., "Estimation Of Object Motion Parameters From Noisy Image", *IEEE Transaction On Pattern Analysis And Machine Intelligence*, Vol. PAMI-8, No.1, pp 90-99, 1986.
- Brown R.G., *Introduction To Random Signal Analysis And Kalman Filtering*, John Wiley & Sons, Inc., 1983.
- Chatila Raja, Laumond Jean-Paul, "Position referencing and consistent world modeling for mobile robots", in *IEEE Int. Conf. on Robotics and Automation*, 1985.
- Clover C.L., Bernard J.E., "The influence of lateral load transfer distribution on directional response", *Society of Automotive Engineering Paper No. 930763*, 1993.
- DeMenthon D., "A Zero-bank Algorithm for Inverse Perspective of A Road from A Single Image", in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 1444-1449, April 1987.
- Dickmanns E.D., Graefe V., "Dynamic Monocular Machine Vision", *Machine Vision and Applications* 1, pp. 223-240, 1988.
- Dingle Julie, "IVHS Legal Issues-Liability And Privacy", A Presentation to the SAE TOPTEC Conference, Orlando, Fl., Jan. 1993.
- Durrant-Whyte H.F., "Uncertain Geometry in Robotics", *IEEE Journal of Robotics and Automation*, Vol. 4, No. 1, Feb. 1988.
- Everett H.R., "Survey of Collision Avoidance and Ranging Sensors for Mobile Robots", *Robotics and Autonomous Systems* 5, pp 5-67, 1989.
- Fenton R.E. and Mayhan R.J., "Automated Highway Studies At The Ohio State University-An Overview", *IEEE Trans. on Vehicular Technology*, Vol. 40, No.1, pp. 100-113, Feb. 1991.
- Franke U., "Real Time 3D-road Modeling for Autonomous Vehicle Guidance", *7th Scandinavian Conference on Image Analysis*, Ralborg, Denmark, 1991.
- Franklin G.F., Powell J. D., Workman M.L., *Digital Control of Dynamic Systems*, New York: Addison-Wesley Publishing Company; 1990.
- Fu K.S., Gonzalez R.C., Lee C.S.G., *Robotics :Control, Sensing, Vision, and Intelligence*, McGraw-Hill International Editions, 1987.

- Gillespie T.D., *Fundamental of Vehicle Dynamics*, Society of Automotive Engineers, Inc., 1992.
- Godthelp, H.; Milgram, P.; Blaauw, G. "The Development of a Time-Related Measure to Describe Driving Strategy", *Human Factors*. 1984; 26(3); p. 257-268, 1984.
- Gourishankar V., Kudva P., and Ramar K., "Reduced-order Observers For Multivariable Systems With Inaccessible Disturbance Inputs", *Int. Journal of Control*, Vol. 25, No. 2, pp. 311-319, 1977.
- Graefe V., Kuhnert K., "Vision-based Autonomous Road Vehicles", in *Vision-Based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Hayward J., "Near-Miss Determination Through Use Of A Scale Of Danger", *Highway Research Record*, No. 384, pp. 24-34, 1972.
- Hess R.A, Madjtahedzadeh A., "A Control Theoretic Model Of Driver Steering Behavior", *IEEE Control System Magazine*, pp. 3-8, 1990.
- Hou M. and Muller P.C., "Design Of Observers For Linear Systems With Unknown Inputs", *IEEE Trans. on Automatic Control*, vol. 37, no 6, June 1992.
- Huh K., Pilutti T., Ulsoy A.G., MacAdam C., Lin C.F., Ervin B., *An Advanced Vehicle-Steering Control System To Avoid Run-Off Road Accidents*, IVHS Progress Report, The University of Michigan, 1992.
- Jain R., Course Pack of EECS 442, *Introduction to Computer Vision*, The University of Michigan, 1992.
- Johnson C.D., "Accommodation of Disturbances in Optimal Control Problems", *Int. Journal of Control*, Vol. 15, No. 2, pp. 209-231, 1972.
- Johnson C.D., "Disturbance-utilizing Controllers for Noisy Measurements And Disturbances", *Int. Journal of Control*, Vol. 39, No. 5, pp. 859-868, 1984.
- Johnson, W., *Two -Time-Scale Predictor Model Using Scan of Anticipates Inputs*, department of Mechanical engineering, M.I.T., Cambridge, master's thesis, 1965.
- Kamada H., Yoshida M., "A Visual Control System Using Image Processing and Fuzzy Theory", *Vision-Based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Kenue S.K., "LANELOK: Detection of Lane Boundaries and Vehicle Tracking Using Image-Processing Techniques-Part I: Hough Transform, Region-Tracking and Correlation Algorithm", *Proc. SPIE Conference on Mobile Robots IV*, Philadelphia, PA., Nov. 1989.
- Kriegman D.J., Triendl E., and Binford T.O., "Stereo Vision and Navigation in Buildings for Mobile Robots", *IEEE Trans. on Robotics and Automation*, Vol. 5, No. 6, Dec. 1989.

- Lin Chiu-F. and Ulsoy A. Galip, "Calculation Of The Time To Lane Crossing And Analysis Of Its Frequency Distribution", *Proc. of American Control Conference*, Seattle, WA. 1995a.
- Lin Chiu-F. and Ulsoy A. Galip, "Vehicle Dynamics And External Disturbance Estimation for Vehicle Path Projection", *Proc. of the American Control Conference*, Seattle, WA., 1995b.
- Lin Chiu-F., Ulsoy A. Galip, LeBlanc D.J., "Lane Geometry Reconstruction : Least Square Curve Fit Versus Kalman Filter", submitted to *American Society of Mechanical Engineering International Mechanical Engineering Conference and Exhibition*, San Francisco, CA, 1995.
- Liou S., Jain R., "Road Following Using Vanishing Points", *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition*, Miami Beach, FL., pp.41-46, June 1986.
- MacAdam Charles C. , *Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis*, Final Technical Report for The U.S. Army Tank Automotive Command, Report No. UMTRI-88-53, Dec. 1988.
- Maquin D., Gaddouna B., and Ragot J., "Estimation Of Unknown Inputs In Linear Systems", *Proceeding of the American Control Conference*, Baltimore, June 1994.
- Matthies L., and Shafer S.A., "Error modeling in stereo navigation", *IEEE J. of Robotics and Automation.*, Vol. RA-3, No. 3, pp. 239-248, 1987.
- Mayhan R.J. and Bishel R.A., "A Two-Frequency Radar for Vehicle Automatic Lateral Control", *IEEE Transaction on Vehicular Technology* Vol. Vt-31, No.1, Feb. 1982.
- McRuer D.T., Weir D.H., Jex H.R., Magdaleno R.E., and Allen R.W., "Measurement Of Driver-Vehicle Multiloop Response Properties With A Single Disturbance Input", *IEEE Transactions, SMC-5(5)*, pp. 490-497, 1975.
- Moravec H.P., *Obstacle Avoidance And Navigation In The Real World By A Seeing Robot Rover*, Ph.D. dissertation, Stanford University, Stanford CA, Sep. 1980.
- Okuno,A. Fujita, K., and Kutami A., "Visual navigation of an autonomous on-road vehicle: autonomous cruising on highways", *Vision-Based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Olson K. W., "Wire-Reference Configurations in Vehicle Lateral Control", *IEEE Trans. on Vehicular Technology*, Vol. VT-26, No. 2, May 1977.
- Park Youngjin, *State And Input Observers For Model-Based Machine Diagnostics*, Ph. D. Dissertation, The University Of Michigan, 1987.
- Parker G., *IVHS Research Topics*, The University of Michigan, Feb. 4, 1993.
- Pilutti T. and Ulsoy A. G. , "Online Identification Of Driver State For Lane-Keeping Tasks", *Proc. of American Control Conference*, Seattle, WA. 1995a.

- Pilutti T. and Ulsoy A. G., "Vehicle Steering Intervention Through Differential Braking", *Proc. of American Control Conference*, Seattle, WA. 1995b.
- Poggio T., "Early Vision : From computational structure to algorithms and parallel hardware", *Comput. Vision Graphics, and Image Processing*, Vol. 31, pp. 139-155, 1985.
- Rives P., Breuil E., Espian B., "Recursive Estimation Of 3D Features Using Optical Flow And Camera Motion", Hertzberger LO (ed) *Proc. of the Conf. on Intelligent Autonomous Systems*, Amsterdam, pp 522-532, 1986.
- Sheridan T.B., Johnson W.M., Bell A.C., and Kreifeldt J.C., "Control Models Of Creature Which Look Ahead", *Proc. Fifth Nat. Symp. on Human Factors in Electronics*, San Diego, May 1964.
- Shladover S., Desoer C.A., Hedrick J.K., Tomizuka M., Walrand J., Zhang W.B., McMahon D.H., Peng H., Sheikholeslam S., McKeown N., "Automatic Vehicle Control Developments in the Path Program", *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 1, pp. 114-130, Feb. 1991.
- Thorpe C., Hebert M., Kanade T., Shafer S., "The New Generation System for the CMU Navlab", in *Vision-Based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Tsugawa S., Hirose T., and Yatabe T., "An Intelligent Vehicle With Obstacle Detection And Navigation Functions", *Proc. IECON*, Tokyo, pp.303-308, 1984.
- The Hansen Report, "NHTSA To Promote Collision-Avoidance Technology", The Hansen Report on Automotive Electronics, July/August, Vol.5, No.6, 1992.
- Turk M.A., Morgenthaler D.G., Gremban K.D., Marra M., "VITS-a vision system for autonomous land vehicle navigation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.10, No.3, May 1988.
- Vedamuthu S., Law E.H., "An Investigation Of The Pulse Steer Method For Determining Automobile Handling Qualities", *Society of Automobile Engineering Paper* No. 930829, 1993.
- Waxman A.M., Lemoigne J., and Scrinivasan B., "Visual Navigation Of Roadways", in *Proc. IEEE international conference on Robotics and Automation*, St. Louis, MO., Mar. 1985

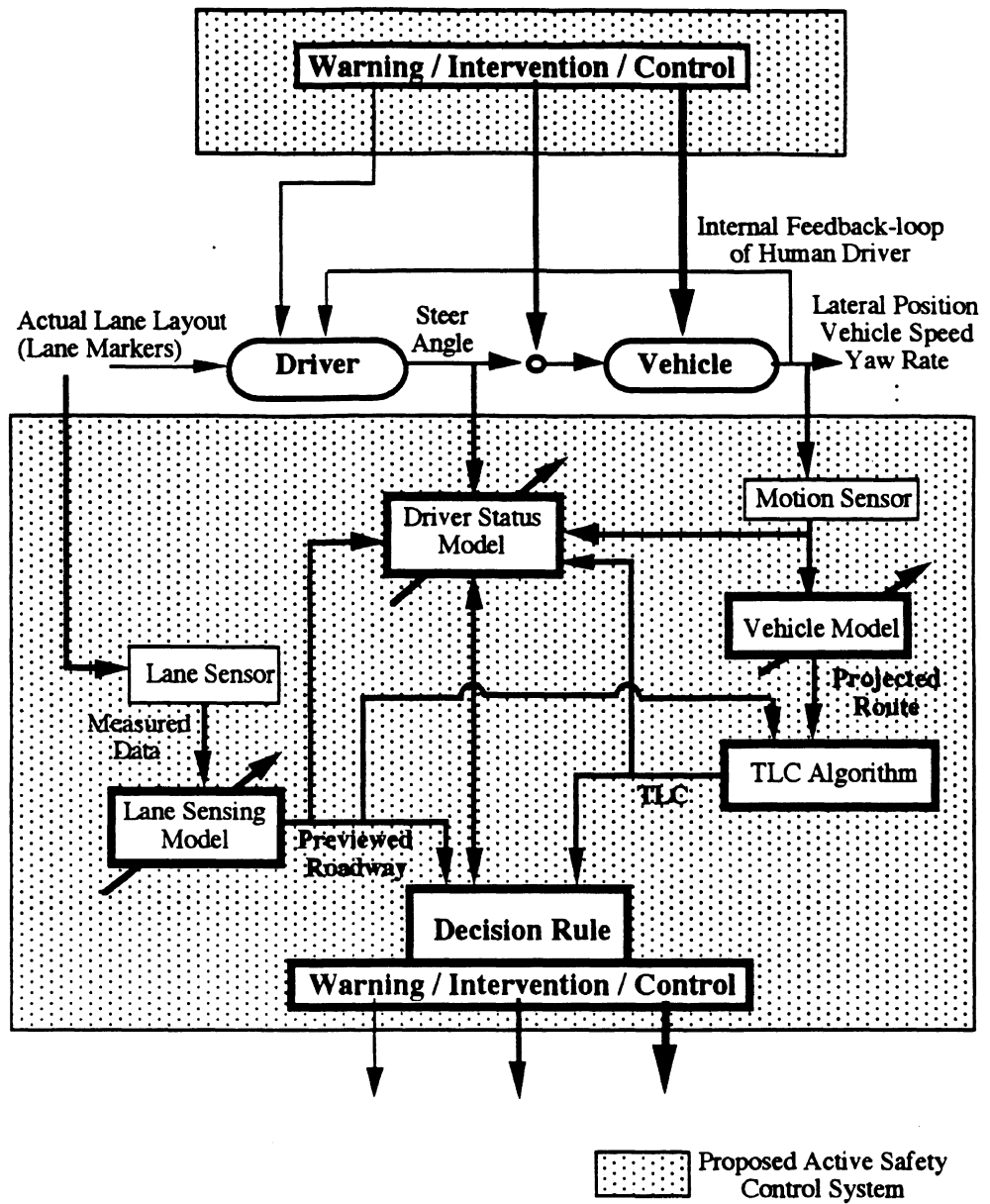


Figure I.1 : Structure Of The Active Safety Control System [Huh *et.al*, 1992]

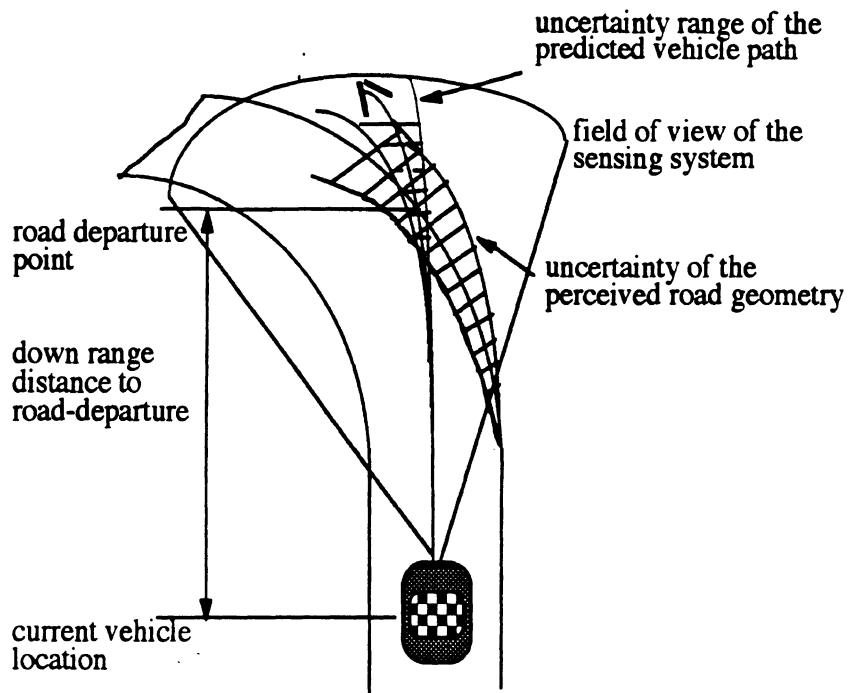


Figure I.2: Elements for Time to Lane Crossing Calculation Error

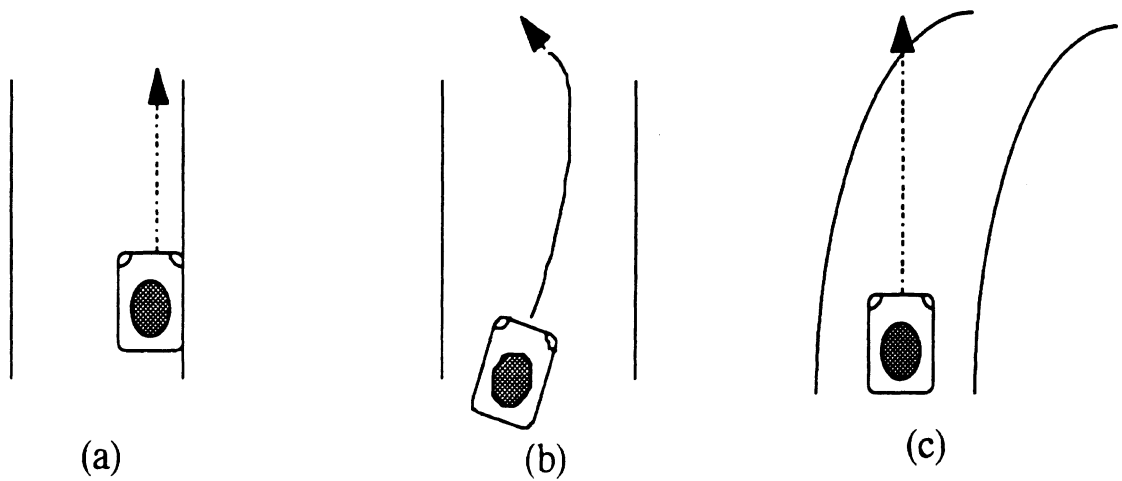


Figure I.3 : Advantages Of Time To Lane Crossing For "Lane Tracking Margin" Assessment For A Vehicle

CHAPTER II

TIME TO LANE CROSSING CALCULATION AND CHARACTERIZATION OF ITS ASSOCIATED UNCERTAINTY

Abstract

This chapter presents an algorithm to calculate Time to Lane Crossing (TLC). Several factors which may affect the accuracy of TLC are investigated, among which, the vehicle vibration, external disturbances acting on a vehicle, and vehicle state measurement errors are found to be the most significant. The bandwidth of time to lane crossing for typical highway driving is also studied; a bandwidth of 2.5 Hz is identified. Finally, an algorithm is proposed to characterize the uncertainty in the TLC estimate.

Introduction

This chapter discusses an algorithm to calculate Time to Lane Crossing (TLC) and presents the analysis of its frequency content. Furthermore, it also discusses an algorithm to characterize the uncertainty in the perceived TLC. Time to lane crossing in this context describes the time required for a vehicle to run off the road boundary assuming no further steering intervention will be taken by the driver (i.e., the current steering wheel angle is held constant). Such a metric is important for an adaptive safety system being developed at the University of Michigan to prevent road-departure accidents. It provides an appropriate way for the active safety system to assess the "lane tracking margin" of a vehicle. A characterization of the TLC uncertainty is also necessary to provide a reliability index for the active safety system decision making [LeBlanc *et.al.*, 1995].

To calculate TLC, down range road geometry and the future vehicle path are necessary such that the intersection of the vehicle path and road boundary can be located

and the time to reach this point can be calculated(see Fig. II.1). The current study assumes that two polynomial equations which represent the down range road boundaries will be available from a road geometry reconstruction module [Lin *et.al*, 1995]. It also assumes that an array which contains discrete points along a predicted future vehicle path will be available from a vehicle path prediction module, which is the result of a discrete vehicle path projection [Lin and Ulsoy, 1995]. The calculation algorithm uses this information to calculate TLC. It further applies an interpolation scheme to reduce the error caused by the time resolution for discrete vehicle path projection. On the other hand, the uncertainty characterization algorithm uses uncertainties of the perceived down range road geometry and the predicted future vehicle path to characterize the perceived TLC uncertainty. Here, the uncertainty range refers to statistical characteristics (one standard deviation) of the possible deviation of the perceived values from the real values. The uncertainties of the perceived road geometry and the predicted vehicle path are also assumed to be available from the road geometry reconstruction and vehicle path prediction modules.

In the past, several different schemes have been utilized for “lane tracking margin” assessment. These schemes include instantaneous lateral deviation ([Shladover, 1991], [Fenton, 1991], [Dickmanns, 1988]), instantaneous vehicle heading angle([Kamada, 1992], [Turk, 1988]), and predicted relative location of the vehicle from the lane boundary at a future time [Okuno, 1992]. These metrics only provide lane tracking margin at a specific time (i.e., either the current time or a future time) and have no information about what happens at other times. However, such metrics may mislead an active safety system because a large deviation from the desired trajectory does not always mean a dangerous maneuver. On the other hand, TLC is less ambiguous for the active safety system decision making because it evaluates the entire predicted vehicle path. The current study defines TLC as the time for the vehicle to reach a road edge assuming no future driver intervention (i.e., the steering wheel is help constant). Such a definition reduces the effect of human factors on TLC, a decoupling which benefits the active safety system design. Furthermore,

it also eliminates the uncertainty in the estimation caused by inaccurate prediction of the future driver maneuvers.

There are several issues related to TLC which are important for active safety systems, but have not been investigated previously. The first issue is the bandwidth of TLC. An understanding of TLC bandwidth is important such that an appropriate sampling rate (or the cycle time for a TLC calculation) can be determined. Another issue is the accuracy of the TLC, which was noted but not characterized in previous TLC studies [Godthelp, 1984]. The study of this accuracy issue should involve two aspects. The first aspect is to improve the TLC accuracy (i.e., to develop algorithms to compensate for TLC errors). The second aspect is that an uncertainty characterization algorithm must be developed to establish the expected reliability of TLC estimates. In this study, several different factors which may affect the accuracy of TLC are studied. Then, based on the results, an algorithm to characterize the TLC uncertainty is proposed.

The following section presents the TLC calculation scheme including the interpolation scheme used to refine TLC. The next section discusses the study of the TLC bandwidth. Then, the studies of the significance of different factors on TLC accuracy is presented. These are followed by an algorithm to characterize the TLC uncertainty. A summary and conclusions is included at the end of the chapter.

Time To Lane Crossing Calculation Algorithm

This section discusses a "brute force" approach to TLC calculation. With such a scheme, the vehicle path projection time step affects the TLC accuracy. An interpolation algorithm is introduced to minimize this time-step effect.

As shown in Fig. II.2, time to lane crossing is obtained by evaluating the discrete points along the vehicle path until a point lies outside of the road boundary. Then, an interpolation scheme is applied to refine the result. In this figure, (x_i, y_i) is a location of the vehicle in the down range predicted path, $f_l(x_i)$, $f_r(x_i)$ represent the lateral locations of

the left and right lane boundaries evaluated at x_i , Δt is the iteration time step for the discrete model, and TLC denotes time to lane crossing. Suppose that the vehicle location lies outside the road boundary at time step i , a rough estimate of the TLC is then $i\Delta t$. Then, based on this TLC estimate, a linear interpolation scheme [Carnahan, 1977] is applied to refine the TLC.

The previous step shows that the intersection lies between (x_i, y_i) and (x_{i-1}, y_{i-1}) . Thus, a linear polynomial equation can be developed to represent the vehicle path connecting these two points;

$$y(x) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x - x_{i-1}) \quad (\text{II.1}).$$

Then, Equation (II.1) can be used, along with the road boundary which the vehicle path intersects, to refine the TLC. A time efficient way is to use a binary search for a more accurate intersection. A binary search evaluates the middle point of a possible intersection area to reduce the area. For example, after Equation (II.1) is developed, the middle point between (x_i, y_i) and (x_{i-1}, y_{i-1}) is calculated;

$$\begin{aligned} x' &= x_{i-1} + \frac{1}{2}(x_i - x_{i-1}) \\ y' &= y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}}(x' - x_{i-1}) \end{aligned} \quad (\text{II.2}).$$

Then (x', y') is evaluated to see if it lies outside of the road boundary or not. If it is, then a new area of possible intersection is between (x_{i-1}, y_{i-1}) and (x', y') . If it is not, then the possible intersection is between (x', y') and (x_i, y_i) . The same procedure continues until the area of possible intersection is smaller than a specified threshold. Suppose (x'', y'') is the final point of this procedure, then a refined TLC can be obtained by

$$TLC = (i-1)\Delta t + \frac{\Delta t}{x_i - x_{i-1}}(x'' - x_{i-1}) \quad (\text{II.3}).$$

Table II.1 shows the improvement in TLC error by applying this interpolation scheme to a TLC simulation in which the TLC's are generally small (about 3.0 sec). The time step for vehicle path projection is 0.1 sec. The results show that a reduction of about 40% of the error can be achieved. However, for a large TLC, the performance is somewhat degraded. Under such situations, the real TLC does not fall within the time period of the rough estimation (see Figure II.2) and the interpolation scheme causes a larger error.

Table II.1 : Improvement In Time To Lane Crossing With A Linear Interpolation Scheme

	<i>With Interpolation</i>	<i>Without Interpolation</i>
RMS Value Of The TLC Error	0.06 sec.	0.10 sec.

A typical TLC is shown in Figure II.3, which simulates the TLC with an "intoxicated" driver driving at a speed of 90 Km/h on a typical highway geometry (i.e., a straight section connected by a curve). During driving, the vehicle experiences a strong wind disturbance. An "intoxicated" driver is simulated with an optimal driver model developed by MacAdam [1988], in which the delay time is assigned longer than the typical delay (i.e., 3.0 sec. for an "intoxicated" driver versus 2.0 sec. for a normal driver). During the initial period, the driver tries to regulate the vehicle under the effect of the wind disturbance; however, because the driver is "intoxicated", which causes a longer response time, the regulation takes a longer time to settle down the vehicle motion (about 4 sec). For a normal driver, it may take only about 2 sec. Then, the vehicle encounters a curved road geometry and the curvature serves as a disturbance to the vehicle dynamics. The same regulation phenomenon happens, which is shown by the ripples at the end.

Time To Lane Crossing Frequency Distribution

This section describes a study of the frequency distribution of TLC in typical highway driving. Identification of the maximum significant frequency component of TLC

is the goal of this study. Subsequently, the requirement on sampling rates for TLC is discussed.

Since TLC is determined by the predicted vehicle path and the down range road geometry, the variation of TLC is affected by the variation of roadway geometry and the projected vehicle path. For roadway geometry, a dramatic variation occurs at the intersection of two roadway segments with different curvatures (e.g., a straight line connected to a pure curve). On the other hand, the vehicle path has more variety. The variation of the vehicle path depends on the variation of the dynamic states, the steering input, and the external disturbance. Furthermore, vehicle dynamic states are functions of steering input and external disturbances (e.g. wind disturbance and superelevation). Therefore, the variation of the external disturbances and steering input in driving are the two fundamental sources of the geometric variation of the predicted path. In summary, the variation of steering input, external disturbance and roadway geometry must be considered while studying the frequency distribution of the TLC.

For this study, the external disturbance and roadway geometry are selected to have relatively fast variations compared to what a driver is generally expected to see in highway driving. The superelevation, roadway geometry, and the constant component of a wind force are ignored in the simulations because these factors change gradually, and, thus only affect the low frequency component of TLC.

On the other hand, the driver maneuver (i.e., variation of steering input) is divided into four typical conditions such that the TLC bandwidth in these situations can be compared. The driver maneuvering is simulated with an optimal driver model developed by [MacAdam, 1988]. The first maneuver features a normal driver following the center of the lane, a common maneuver for most people in highway driving. Although some people may drive at an offset from the center of the lane, it is expected that the frequency response of this situation does not differ from the situation of lane center tracking. The second maneuver features an "intoxicated" driver following the center of the lane. For simulating

an intoxicated driver, a longer time delay in the optimal driver model is applied and the preview time is kept the same as for the normal driver model. The third maneuver simulates driving with additional tasks such as locating a scenic point on the map, which also simulates driving with a drowsy driver. For such simulation, a fixed steering angle is assigned. The validity of using a fixed steering angle to simulate such maneuvers is discussed in [Godthelp, 1984] and [McDonald, 1980]. Then, frequency analyses are conducted to locate the TLC bandwidth for each of the different maneuvers. The results are shown in Table II.2. Finally, an emergency maneuver is simulated as the fourth case. A driving simulator with a steering wheel is used for the experiment. In the study, the subjects were asked to make a sharp turn of the steering wheel and the same procedures were repeated several times. No external disturbance is assigned in order that the effect from the maximum variational speed of the driver input can be manifested. The emergency maneuver case has the most rapid variation in TLC with a bandwidth of about 3.0 Hz, and is also included in Table II.2.

Table II.2 : Time To Lane Crossing Bandwidth In Different Maneuvering Situations

<i>Maneuvering Conditions</i>	<i>Normal Driver</i>	<i>"Intoxicated" Driver</i>	<i>Drowsy Driver</i>	<i>Emergency Maneuver</i>
TLC Bandwidth	2.5 Hz	2.5 Hz	1.0 Hz	3.0 Hz

The results show that the possible maximum frequency component is about 3.0 Hz, which occurs in an emergency steering maneuver. According to Shannon's sampling theorem, the corresponding sampling Frequency must be at least 6 Hz. However, in practice, 4 to 10 times the signal bandwidth is usually required for sampling the signal. Therefore, in order to successfully track the TLC variation for every different situation, the above maximum value sets a minimum of about 12 Hz, and preferably 30 Hz, sampling rate for the TLC. However, since the emergency maneuver situation is not a scenario in which the active safety system is expected to operate, this requirement can be somewhat relaxed. The

results from the first three simulations show that the maximum possible frequency component in the scenario of the active safety system is located at about 2.5 Hz. Therefore, it is desired that the sampling frequency for the TLC is at least 10 Hz but preferably at about 25 Hz. Since the TLC sampling rate is expected to be dominated by the image processing for the previewed scene. The above result suggests that a desired image processing rate is at about 10 Hz or higher.

Significant Factors For Time To Lane Crossing Accuracy

This section studies the significance of some of the factors which may affect the accuracy of time to lane crossing. The results will be considered when developing algorithms to compensate for the errors or to characterize the uncertainty of TLC. An assumption in this study is that the vehicle characteristics (e.g., vehicle mass, tire pressure, etc.) will not change while driving on the road. Also, the vehicle is assumed to be traveling at a speed of 90 Km/h under cruise control.

The factors which may affect the accuracy of TLC are categorized into two: (1) those which affect road geometry perception, and (2) those which affect future vehicle path prediction. For the road geometry perception, there are mainly three factors : 1) pixel resolution, 2) vehicle vibration (which causes a sensor to lose its orientation), and 3) superelevation (due to insufficient information to recover 3-D geometry from a 2-D image). On the other hand, there are also three factors which may cause inaccurate vehicle path prediction : 1) unknown future external disturbances, 2) linear 2 Degree of Freedom (DoF) vehicle model for path projection (the real system is nonlinear and higher order), 3) measurement error (or no measurement) for the initial conditions (i.e., lateral vehicle velocity, yaw rate, and front wheel steering angle). This section discusses the significance of these factors on TLC accuracy.

Figure II.4 shows the effects of pixel resolution, vehicle vibration, and superelevation on TLC accuracy. For the values considered, Fig. II.4 shows that pixel

resolution has a negligible effect on the TLC accuracy. Here, the pixel resolution features a 512(horizontal) x 256 (vertical) resolution in a 10 mm x 10 mm CCD chip. On the other hand, the effect of vehicle vibration introduces significant error in the TLC perception. This is because vehicle vibration causes unknown changes in the vision sensor orientation and leads to incorrect perception of the road geometry. Here, the vehicle vibration is excited by a typical highway road surface (smooth asphalt). A worse result is obtained when the vehicle runs on a rougher surface. The effects from typical highway superelevation (for the simulation, it is 0.5 deg) is less substantial than the vehicle vibration. However, simulation results show that superelevation introduces a significant error when it is large enough (e.g. a 2.5 deg superelevation).

The effects of model simplification, unknown external disturbances, and measurement errors for the initial conditions for vehicle path projection are shown in Fig. II.5. To study the model simplification effect, a 3 DoF vehicle model which considers a roll degree of freedom and the nonlinearity of the tires is compared with the 2 DoF model. To simulate the nonlinearity of a tire, a nonlinear tire model generally known as the magic formula [Bakker, 1989] is used. The results shows that for highway driving under a substantial external disturbance, a 2 DoF model works satisfactorily. The initial condition measurement error effects are studied by assuming a random error with 0.1 deg/s standard deviation for the yaw rate, 0.1 deg at the front wheel steering angle, and no lateral velocity measurement. This result shows that such measurement errors are significant in terms of TLC accuracy. Finally, the unknown external disturbance effect is studied by adding a wind force featuring a 4 m/s side wind velocity in the vehicle path prediction. Such a wind force is considerably strong for typical highway driving [DeHarpporte, 1983]. The results show that an estimation of the external disturbance under this situation for path prediction is substantial.

The above simulations show that for TLC calculation in typical highway driving, schemes must be developed to compensate for the TLC error due to the following effects :

1) vehicle vibration, 2) superelevation, 3) external disturbances, and 4) path projection initial condition measurement errors. Otherwise, these effects must be considered when developing an algorithm to characterize TLC uncertainty. Among these four factors, vehicle vibration is the most significant factor, which causes about twice the TLC error as the path projection initial condition measurement errors do. Therefore, an accurate road geometry perception will be more important than an accurate vehicle path projection. However, it is still necessary to develop an accurate vehicle path algorithm to reduce the TLC uncertainty for the TLC to be successfully utilized by the active safety system.

Characterization Of Time To Lane Crossing Uncertainty

This section discusses an algorithm to characterize the possible deviation of the perceived TLC from the real TLC. The result is a statistical characteristics (the standard deviation) of the possible errors and is called the uncertainty range in this context. This algorithm assumes that the uncertainties of the perceived roadway geometry and predicted vehicle path will be available. The uncertainty characterization schemes for the perceived roadway geometry and the predicted vehicle path consider those significant factors discussed in the previous section ([Lin et.al., 1995], [Lin and Ulsoy, 1995]).

To characterize TLC uncertainty, both the road boundary and predicted vehicle path are represented with 2nd order polynomial equations :

$$y_r = c_{r0} + c_{r1}x_r + c_{r2}x_r^2 \quad (II.4)$$

where (x_r, y_r) is a point on the lane boundary, and

$$y_v = c_{v0} + c_{v1}x_v + c_{v2}x_v^2 \quad (II.5)$$

where (x_v, y_v) is a point in the predicted vehicle path. At the intersection point,

$(x_r, y_r) = (x_v, y_v)$; therefore,

$$c_{r0} + c_{r1}x_r + c_{r2}x_r^2 = c_{v0} + c_{v1}x_v + c_{v2}x_v^2 \quad (II.6).$$

Denote $x_v = x_r = l$ and rearrange Equation (II.6) to obtain

$$(c_{r0} - c_{v0}) + (c_{r1} - c_{v1})l + (c_{r2} - c_{v2})l^2 = 0 \quad (II.7).$$

The solution of Equation (II.7), l , can be obtained in one of two ways; the first one is to solve Equation (II.7) and the other way is to calculate l from

$$TLC = l/u \quad (II.8)$$

by assuming constant forward velocity, u , of the vehicle. The time to lane crossing in equation (II.8) is obtained from the time to lane crossing calculation algorithm. Then, the error analysis for l can be conducted by

$$\begin{aligned} & (c_{r_o} + \Delta c_{r_o} - c_{v_o} - \Delta c_{v_o}) + (c_{r_1} + \Delta c_{r_1} - c_{v_1} - \Delta c_{v_1})(l + \Delta l) + \\ & (c_{r_2} + \Delta c_{r_2} - c_{v_2} - \Delta c_{v_2})(l + \Delta l)^2 = 0 \end{aligned} \quad (II.9).$$

By assuming Δ 's $\ll 1$ and denoting

$$\vartheta = (c_{v_1} - c_{r_1}) + 2l(c_{v_2} - c_{r_2}) \quad (II.10),$$

the following equation can be obtained

$$\Delta l = (1/\vartheta)((\Delta c_{r_o} - \Delta c_{v_o}) + (\Delta c_{r_1} - \Delta c_{v_1})l + (\Delta c_{r_2} - \Delta c_{v_2})l^2) \quad (II.11).$$

Furthermore, assume the mean values of the coefficient errors are zero,

$$m_{\Delta c_{r_o}} = m_{\Delta c_{v_o}} = m_{\Delta c_{r_1}} = m_{\Delta c_{v_1}} = m_{\Delta c_{r_2}} = m_{\Delta c_{v_2}} = 0 \quad (II.12),$$

then the mean value of the l deviation, $m_{\Delta l}$, is also zero. Equation (II.11) can be rewritten as the product of two vectors;

$$\Delta l = \mathbf{A}_{\Delta l} \mathbf{C}_{\Delta l} \quad (II.13).$$

where

$$\mathbf{A}_{\Delta l} = \left[\begin{array}{cccccc} (1/\vartheta) & -(1/\vartheta) & (1/\vartheta)l & -(1/\vartheta)l & (\frac{1}{\vartheta})l^2 & -(\frac{1}{\vartheta})l^2 \end{array} \right] \quad (II.14)$$

and

$$\mathbf{C}_{\Delta l} = \left[\begin{array}{cccccc} c_{r_o} & c_{v_o} & c_{r_1} & c_{v_1} & c_{r_2} & c_{v_2} \end{array} \right]^T \quad (II.15)$$

Variance of Δl , which is also the variance of l , can be obtained by

$$\sigma^2(\Delta l) = \mathbf{A}_{\Delta l} E[\mathbf{C}_{\Delta l} \mathbf{C}_{\Delta l}^T] \mathbf{A}_{\Delta l}^T \quad (II.16)$$

where $E[\mathbf{C}_{\Delta l} \mathbf{C}_{\Delta l}^T]$ is a covariance matrix with the variance of

$\Delta c_{r_o}, \Delta c_{v_o}, \Delta c_{r_1}, \Delta c_{v_1}, \Delta c_{r_2}, \Delta c_{v_2}$ located along the diagonal. Finally, the variance of the time to lane crossing can be calculated from

$$\sigma^2(\Delta TLC) = \sigma^2(\Delta l) / u^2 \quad (II.17).$$

To validate the above algorithm, Monte Carlo simulations are conducted at different nominal down range road geometry and future vehicle paths (which yield different nominal TLCs). For the Monte Carlo simulations with respect to a nominal TLC, random errors are assigned to the coefficients based on typical road geometry perception and vehicle path prediction errors. Then, the standard deviation of the TLC errors for a nominal TLC case from the Monte Carlo simulation is compared with the prediction using the algorithm discussed above and the results are shown in Fig. II.6. The TLC uncertainty values in Fig. II.6 feature road geometry perception in typical highway driving and a wind disturbance of 5 m/s for constant component and 0.6 m/s standard deviation for the stochastic component. The results show that the proposed algorithm predicts the TLC uncertainty to within an acceptable accuracy; the maximum difference between the predicted value and the result from Monte Carlo simulation is about 10% of the real value (i.e., the value from Monte Carlo simulation). From Fig. II.6, one can see that the assumption of small Δl causes differences between the Monte Carlo simulation and model prediction results when the nominal TLC is large. For a more accurate TLC uncertainty prediction, no assumption on Δl should be used, which leads to a more complex model, however, the small Δl assumption is satisfactory. This algorithm is then applied to simulate the TLC uncertainty in typical highway driving. The simulation has an “intoxicated” (e.g., drunk) driver experiencing a strong side wind acting on the vehicle between 1 and 2 seconds. It is assumed that disturbance characterization and vehicle dynamics estimation schemes are applied, which lead to small uncertainty in the predicted vehicle path. The result is shown in Fig. II.7, which shows that, for the conditions described, a minimum TLC of about 2.0 sec. and a maximum TLC uncertainty of about 0.25 sec. are obtained.

Summary And Conclusion

This chapter presents an algorithm to calculate time to lane crossing which includes an interpolation scheme to refine time to lane crossing. Then, several factors which may

affect time to lane crossing are studied to understand their significance for TLC accuracy. Subsequently, a study of the frequency distribution of typical time to lane crossing values in highway driving is presented. Finally, an algorithm is proposed to characterize the perceived TLC uncertainty under the assumption that effective algorithms are developed to compensate for major errors in the road geometry perception and vehicle path prediction errors (i.e., the polynomial equations for the perceived road geometry and the predicted vehicle have small deviation from the polynomial equations representing the real geometry).

The results show that the interpolation scheme improves the TLC error by about 40%. Results also show that among the factors considered, vehicle vibration, superelevation, external disturbances, and path projection initial condition measurement errors significantly influence TLC error. Among these four factors, vehicle vibration, which causes the sensing system to lose its orientation, is the most significant factor. Vehicle vibration causes about twice the TLC error as the path projection initial condition measurement errors do. Therefore, an accurate road geometry perception will be more important than an accurate vehicle path projection. However, it is still necessary to develop an accurate vehicle path algorithm to reduce the TLC uncertainty for the TLC to be successfully utilized by the active safety system. Furthermore, a 2.5 Hz bandwidth is identified for typical TLC for highway driving. This bandwidth leads to a requirement of at least 10 Hz for the TLC sampling rate (i.e., TLC must be calculated every 0.1 sec). On the other hand, the proposed time to lane crossing uncertainty characterization algorithm is validated through a comparison between the model predicted values and the Monte Carlo simulated values; a maximum difference of about 10% of the real value is obtained. Finally, a typical time to lane crossing and its associated uncertainty are simulated, which shows that when an “intoxicated” driver maneuvers a vehicle on a typical highway geometry and experiences a strong side wind, the minimum TLC is about 2.5 sec. and the maximum TLC uncertainty is about 0.4 sec. The value of the TLC uncertainty is obtained

under the assumption that the vehicle dynamics (i.e., lateral velocity and yaw rate) and external disturbances and the front wheel steering angle can be accurately obtained.

References

- Carnahan, B., Luther H.A., Wilkes J.O., *Applied Numerical Methods*, John Wiley & Sons, Inc., 1977.
- Bakker E., Pacejka H.B., Lidner L., "A New Tire Model With An Application In Vehicle Dynamics Studies", *4th Auto Technologies Conference*, Monte Carlo, SAE 890087, 1989.
- DeHarpporte D., *Northeast and Great Lakes Wind Atlas*, Van Nostrand Reinhold Company, New York, 1983.
- Dickmanns E.D., Graefe V., "Dynamic Monocular Machine Vision", *Machine Vision and Applications* 1, pp. 241-261, 1988.
- Fenton R.E. and Mayhan R.J., "Automated Highway Studies At The Ohio State University-An Overview", *IEEE Trans. on Vehicular Technology*, Vol. 40, No.1, pp. 100-113, Feb. 1991.
- Godthelp, H.; Milgram, P.; Blaauw, G. "The Development of a Time-Related Measure to Describe Driving Strategy", *Human Factors*, 26(3), 1984.
- Kamada H., Yoshida M., "A Visual Control System Using Image Processing and Fuzzy Theory", *Vision-Based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- LeBlanc D.J., Venhovens P.J.Th., Pilutti T.E., Lin C. F., Ervin R.D., Ulsoy A.G., MacAdam C.C., "A Warning And Intervention System For Preventing Inadvertent Road Departure", *14th IAVSD Symposium on the Dynamics of Vehicles on Roads and Tracks*, Ann Arbor, MI, Aug. 1995.
- Lin Chiu-F., Ulsoy A. Galip, and LeBlanc D.J., "Lane Geometry Reconstruction : Least Square Versus Kalman Filter ", *ASME Annual Meeting*, San Francisco, 1995.
- Lin Chiu-F. and Ulsoy A. Galip, "Vehicle Dynamics And External Disturbance Estimation for Vehicle Path Projection", *Proc. of the American Control Conference*, Seattle, WA., 1995.
- MacAdam Charles C. , *Development of Driver/Vehicle Steering Interaction Models for Dynamic Analysis*, Final Technical Report for The U.S. Army Tank Automotive Command, Report No. UMTRI-88-53, Dec. 1988.
- McDonald W.A., and Hoffmann E.R., "Review of Relationship Between Steering Wheel Reversal Rate And Driving Task Demand", *Human Factors*, 22, pp. 733-739, 1980.
- Okuno,A. Fujita, K., and Kutami A., "Visual Navigation Of An Autonomous On-Road Vehicle: Autonomous Cruising On Highways", *Vision-based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Shladover S., Desoer C.A., Hedrick J.K., Tomizuka M., Walrand J., Zhang W.B., McMahan D.H., Peng H., Sheikholeslam S., McKeown N., "Automatic Vehicle Control Developments in the Path Program", *IEEE Transactions on Vehicular Technology*, Vol. 40, No. 1, pp. 114-130, Feb. 1991.

Turk M.A., Morgenthaler D.G., Gremban K.D., Marra M., "VITS-A Vision System For Autonomous Land Vehicle Navigation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.10, No.3, May 1988.

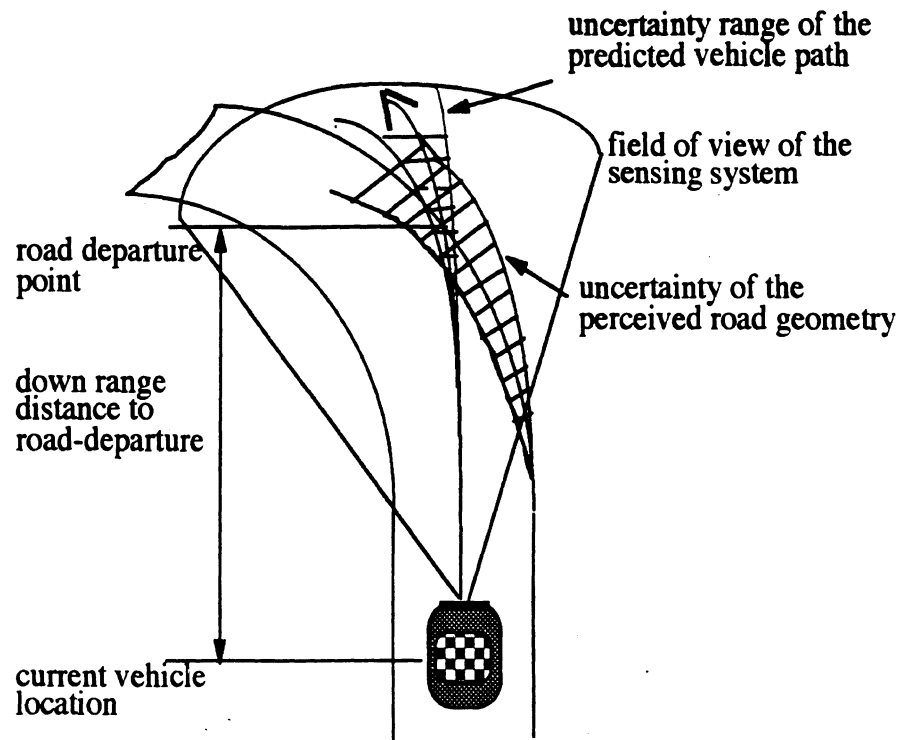


Figure II.1: Elements for Time to Lane Crossing Calculation Error

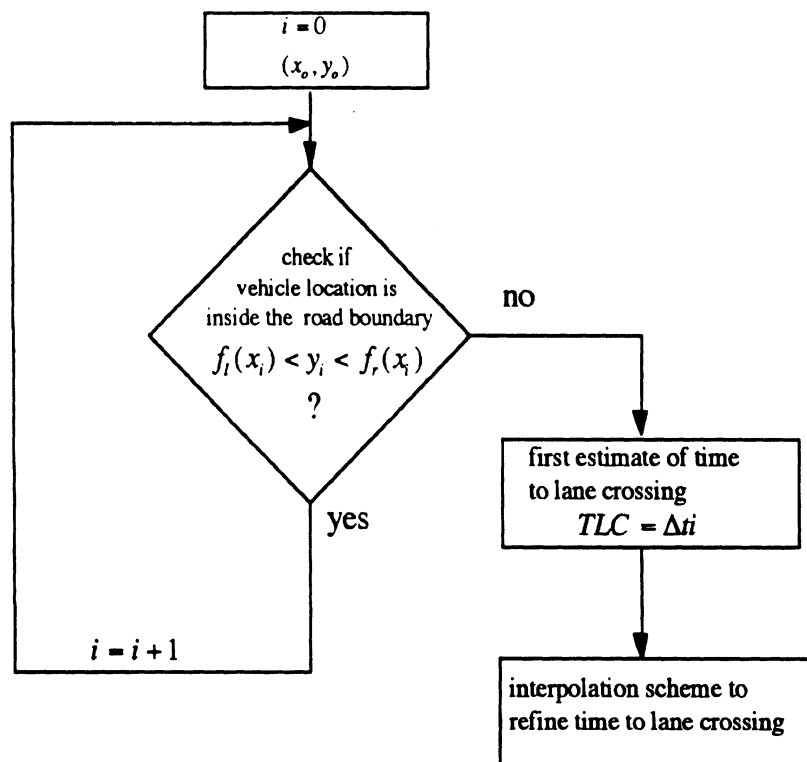


Figure II.2 : Time To Lane Crossing Calculation Algorithm

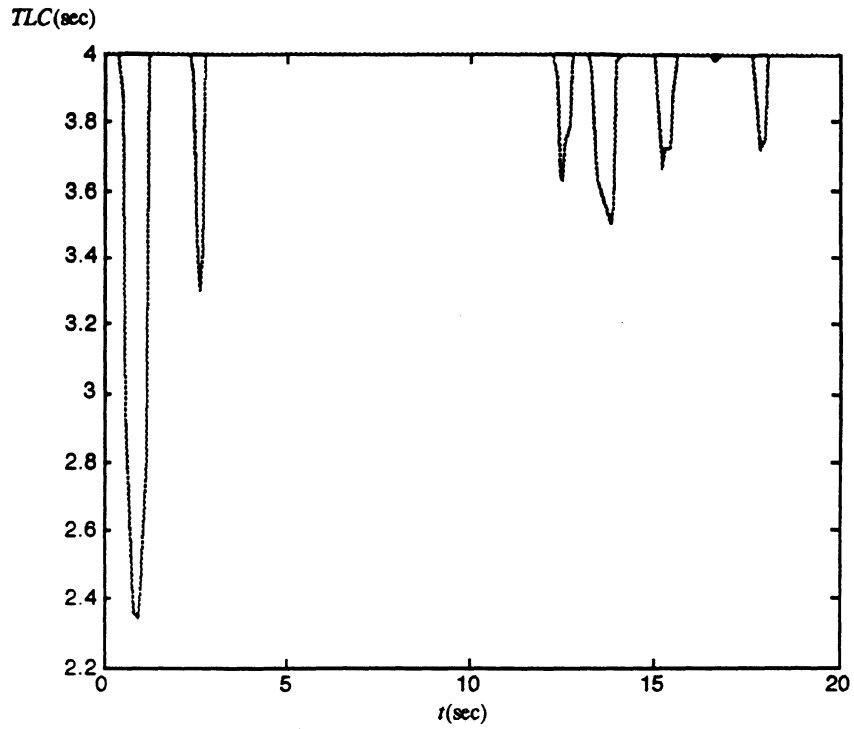


Figure II.3 : Time to Lane Crossing (“Intoxicated” Driver Experiencing A Strong Side Wind)

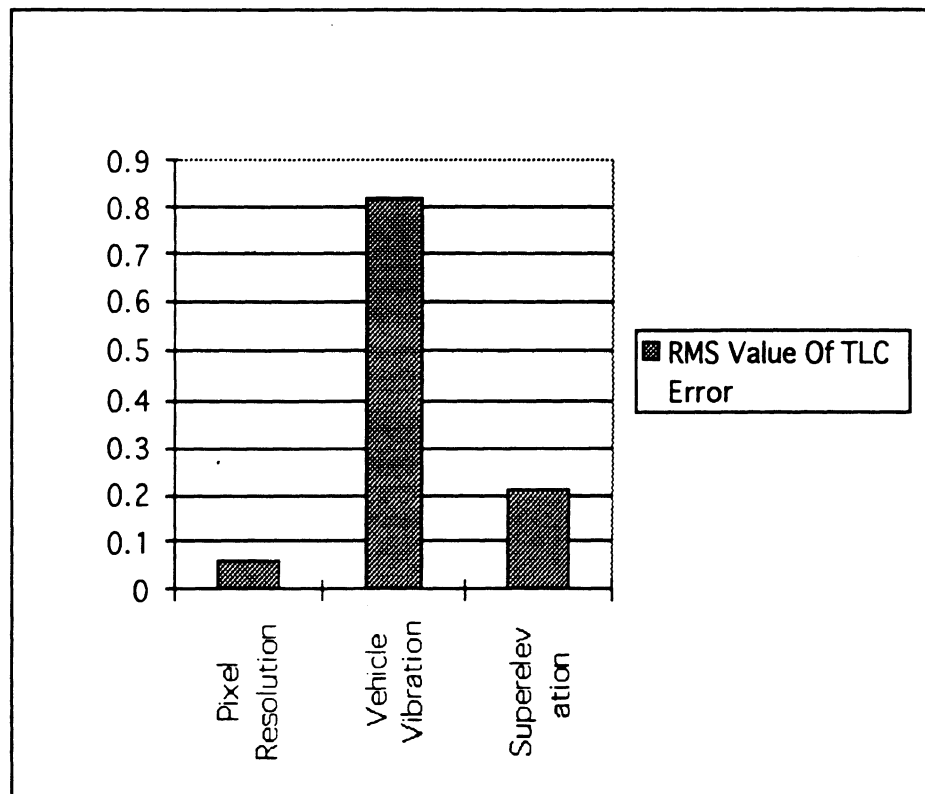


Figure II.4 : Pixel Resolution, Vehicle Vibration, and Superelevation Effects On Time To Lane Crossing Accuracy

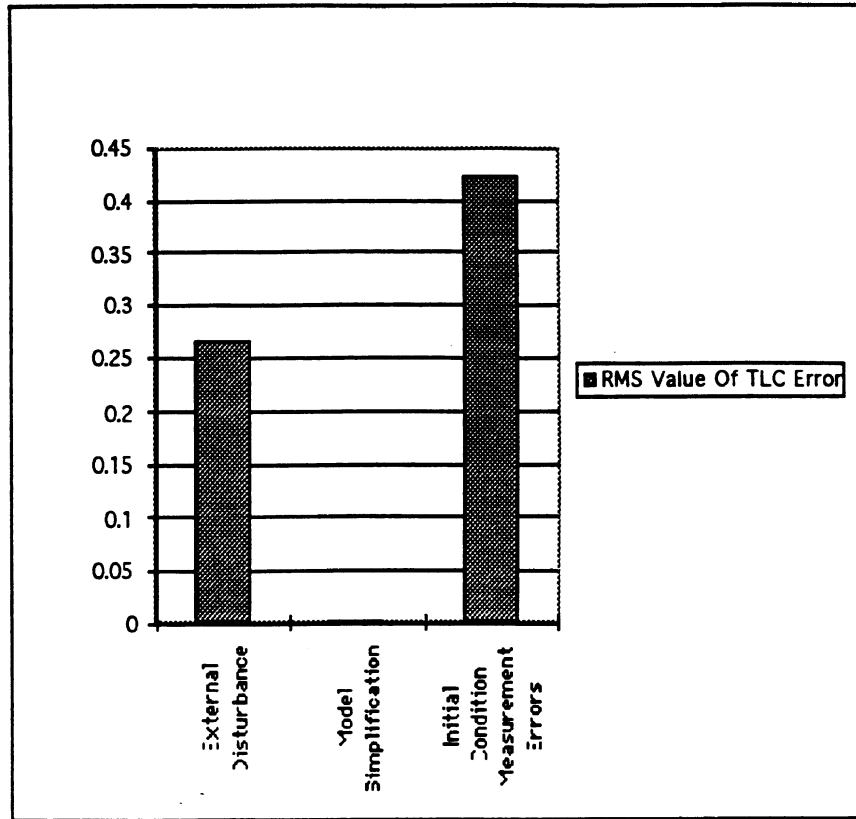


Figure II.5 : Model Simplification, Unknown External Disturbance, and Initial Condition Measurement Error Effects On Time To Lane Crossing Accuracy

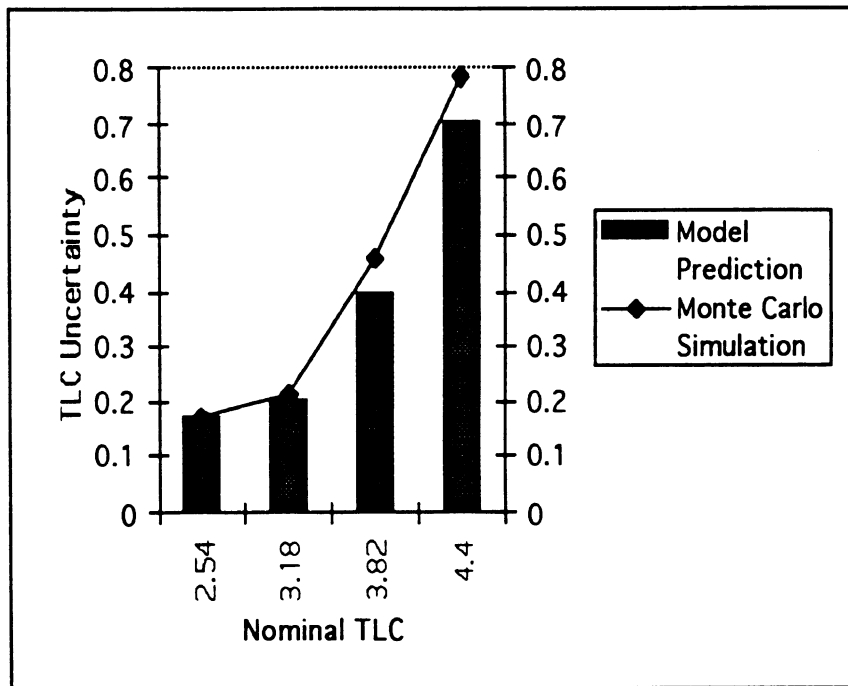


Figure II.6 : Comparison Of The Monte Carlo Simulation And Model Prediction For Time To Lane Crossing Uncertainty

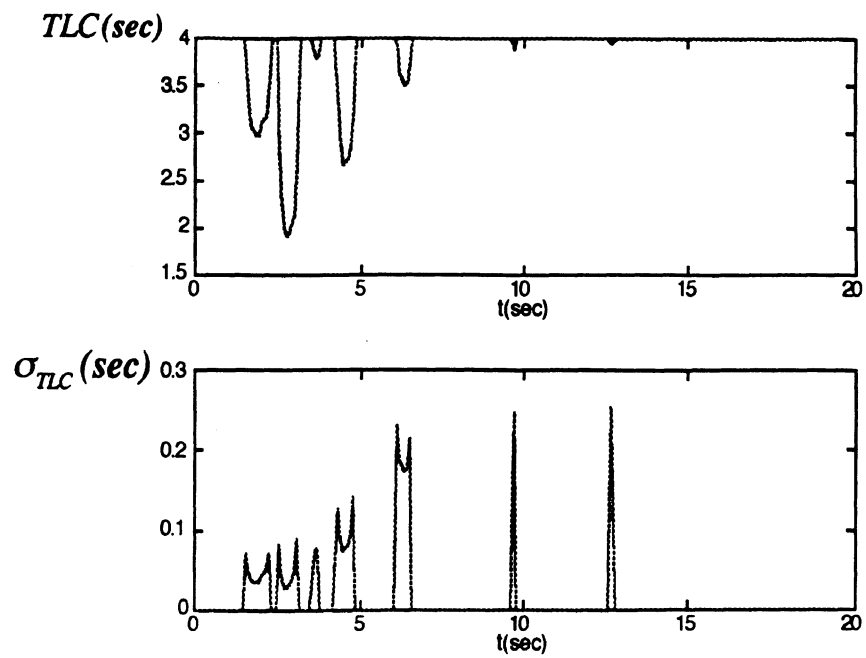


Figure II.7 : A Typical (a) Perceived Time To Lane Crossing And (b) Its Associated Uncertainty (“Intoxicated” Driver Experiencing A Strong Wind Gust Between 1 ~ 2 seconds)

CHAPTER III

VEHICLE PATH PREDICTION AND THE CHARACTERIZATION OF ITS ASSOCIATED UNCERTAINTY

Abstract

This chapter discusses a vehicle path prediction scheme and an algorithm to characterize the prediction uncertainty. For the path prediction, a linearized model is utilized. Furthermore, for accurate path prediction, a steady state Kalman filter which uses perceived lane marker locations, obtained from an on-board near field sensor, is developed to simultaneously estimate vehicle lateral velocity and external disturbances and is shown to significantly improve the result. It is concluded that a fixed constant Kalman filter gain is quite adequate for typical highway driving except when the vehicle encounters a relatively tight curve. Subsequently, different schemes for disturbance characterization to predict future disturbance inputs to the vehicle are discussed. Studies show that a piecewise constant model is adequate for obtaining accurate path prediction. Finally, an algorithm to characterize path prediction uncertainty is proposed. The algorithm assumes that the statistical characteristics of the measurement/estimation errors of the vehicle lateral velocity, yaw rate, and front wheel steering angle are available from off-line characterization.

Introduction

This chapter discusses a future vehicle path prediction algorithm and a scheme to characterize the associated uncertainty. A vehicle path prediction is important for an active safety system being developed at the University of Michigan to prevent road-departure accidents [LeBlanc *et.al.*, 1995]. The predicted path is used to estimate the time to lane crossing of a vehicle. Furthermore, the uncertainty range of the predicted path provides a

reliability index for the predicted path and is used to characterize the uncertainty of the assessed time to lane crossing.

The vehicle path prediction algorithm includes a linearized path projection model and an external disturbance characterization scheme. The linearized path projection model encompasses a 2 Degree of Freedom (DoF) vehicle model and a linear equation to calculate vehicle path. The external disturbance characterization algorithm has a steady state Kalman filter to estimate the external disturbances. Furthermore, based on the estimated data, a system identification scheme is implemented to characterize the external disturbance and to predict future values. The inputs to the Kalman filter are lane marker locations from a near-field sensor and measurements of the vehicle yaw rate and the front wheel steer. Since no lateral velocity measurement is available for the active safety system (which is needed for path projection), the Kalman filter simultaneously estimates this variable. Finally, the uncertainty characterization uses the statistical characteristics of the lateral velocity, yaw rate, and front wheel steering angle measurement/prediction errors to characterize the uncertainty of the predicted path (the predicted path is expressed by a 2nd order polynomial equation and the uncertainty of the coefficients are characterized). Here, the uncertainty refers to the standard deviation of the possible deviation.

Accurate path prediction requires accurate initial conditions (i.e., vehicle lateral velocity, yaw rate, and front wheel steering angle), an accurate path projection model, and knowledge of (future) maneuvers and external disturbances that will occur during the projection time. Previously, the transient dynamics of a vehicle and the external disturbances acting on a vehicle were simply ignored in path projection ([Godthelp, 1984], [Okuno, 1992]). Such an assumption is valid only for a mild environment (i.e., small wind force and superelevation) and introduces significant path projection errors in more severe conditions. This study considers the transient vehicle dynamics by utilizing a 2 DoF vehicle model. The 2 DoF model provides vehicle lateral velocity and yaw rate at each time step. To obtain vehicle path, a nonlinear equation is involved. This study linearized the

nonlinear equation to improve the computational speed. The validity of the linearization is discussed. Furthermore, the external disturbances are also considered here to improve the path prediction by characterizing the variation of the disturbances and then predicting the future values. On the other hand, the future driver steering inputs are usually assumed to be constant because of difficulties in predicting these values, which require a good driver model, accurate perception of the down range road geometry, and current vehicle dynamics. The same assumption (i.e., constant driver input) is also adopted in this study. It offers an advantage in that it reduce the effects of driver related human factors on time to lane crossing calculation. Finally, simulations also show that initial condition measurement errors introduce significant errors in the predicted vehicle path. Such errors are noted but not characterized in previous studies [Godthelp, 1984]. Since the path prediction errors introduced by measurement errors are usually not deterministic, they are characterized statistically in this study and the result is an uncertainty range of the possible deviation of the predicted path from the actual path.

The estimation of the lateral velocity and external disturbances can be achieved through a dynamic state observer. This problem is typically known as the problem of state observation with unknown inputs. The problem of state estimation with unknown inputs has been intensely studied in the past; a literature review can be found in [Park, 1988]. However, simultaneous estimation of the states and unknown inputs did not appear until the works of [Park, 1988] and subsequent researchers ([Hou, 1992], [Maquin. 1994]). These studies apply singular value decomposition methods to accomplish their goals. One important feature in these schemes is that the first derivatives of the measurements are needed; for a system with noisy measurements, these methods may not be applicable. In the current study, a Kalman filter scheme is utilized, in which the measurement noise effect on the estimation is minimized.

The following section discusses the linearized model for path projection. The development of the Kalman filters for lateral velocity and external disturbance estimation is

discussed in the next section, which is followed by a discussion of the external disturbance characterization schemes. Then, an uncertainty characterization scheme is described, which is followed by simulation results to validate the linearized path projection model, to show the performance of the external disturbance characterization scheme (which includes the studies of using a fixed Kalman filter gain for different road surface and geometry), and to validate the uncertainty characterization scheme. Finally, a summary and conclusions are included at the end of the chapter.

Linearized Vehicle Path Projection Model

The vehicle path projection includes a vehicle dynamics model to simulate the variation of the lateral velocity and yaw rate. Furthermore, an equation is used to calculate the vehicle path based on the vehicle kinematics. Such an equation is nonlinear and is linearized in this study to improve computational speed. The result and the 2 DoF model are augmented into a state space model for path projection.

The vehicle lateral dynamics can be approximately described by a 2 DoF lateral dynamics model :

$$\begin{Bmatrix} \dot{v} \\ \dot{r} \end{Bmatrix} = \begin{bmatrix} \frac{1}{mu} (C_r + C_f) & \frac{1}{mu} (-bC_r + aC_f - mu^2) \\ \frac{1}{I_z u} (aC_f - bC_r) & \frac{1}{I_z u} (a^2 C_r + b^2 C_f) \end{bmatrix} \begin{Bmatrix} v \\ r \end{Bmatrix} + \begin{Bmatrix} \frac{-C_f}{m} \\ -aC_f \\ \frac{I_z}{I_z} \end{Bmatrix} \delta + \begin{Bmatrix} e_{F_r} \\ e_{M_r} \end{Bmatrix} \quad (III.1)$$

where v and r and δ are the vehicle lateral velocity, yaw rate, and the front wheel steering angle respectively; C_f and C_r are the cornering stiffness of the front and rear tires respectively, which relate wheel slip angle and lateral forces on the tires; m is the lumped vehicle mass, u is the constant vehicle forward speed, and l is the length of the vehicle wheel base; I_z is the moment of inertia about the z axis of the vehicle fixed coordinate system; a and b are the distances from the front tire axis and rear tire axis to the vehicle's center of gravity respectively; e_{F_r} and e_{M_r} represent lumped disturbances leading to vehicle

lateral and yaw accelerations respectively. This model can be used to simulate vehicle lateral velocity and yaw rate with respect to the vehicle fixed coordinate system defined as the standard SAE coordinate system [Gillespie, 1992]. The vehicle velocity in the reference frame of the predicted vehicle path is then calculated as

$$\begin{aligned} V_x(t) &= u(t) \cos(\varphi_v(t)) - v(t) \sin(\varphi_v(t)) \\ V_y(t) &= u(t) \sin(\varphi_v(t)) + v(t) \cos(\varphi_v(t)) \end{aligned} \quad (III.2)$$

where φ_v is the heading angle of the vehicle with respect to the reference frame. Then, the vehicle location at a time, t , can be obtained by integrating the velocities;

$$\begin{aligned} x_v(t) &= x_{v_o} + \int_0^t V_x(\zeta) d\zeta \\ y_v(t) &= y_{v_o} + \int_0^t V_y(\zeta) d\zeta \end{aligned} \quad (III.3)$$

where (x_{v_o}, y_{v_o}) is the initial condition for the path projection. Suppose the heading angle and the lateral velocity of the vehicle are small along the whole curve and that the forward velocity is constant, equation (III.2) can then be simplified to

$$\begin{aligned} V_x(t) &= u \\ V_y(t) &= u\varphi(t) + v(t) \end{aligned} \quad (III.4).$$

By combining equation (III.1) and equation (III.4) and augmenting heading angle into the dynamics model, the following model can be obtained;

$$\begin{Bmatrix} \dot{y}_v \\ \dot{v} \\ \dot{r} \\ \dot{\varphi}_v \end{Bmatrix} = \begin{bmatrix} 0 & 1 & 0 & u \\ 0 & \frac{1}{mu}(C_r + C_f) & \frac{1}{mu}(-bC_r + aC_f - mu^2) & 0 \\ 0 & \frac{1}{I_z u}(aC_f - bC_r) & \frac{1}{I_z u}(a^2C_r + b^2C_f) & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{Bmatrix} y_v \\ v \\ r \\ \varphi_v \end{Bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ -C_f & 1 & 0 \\ \frac{m}{-aC_f} & 0 & 1 \\ \frac{I_z}{0} & 0 & 0 \end{bmatrix} \begin{Bmatrix} \delta \\ e_F \\ e_M \end{Bmatrix} \quad (III.5).$$

$$\dot{x} = u$$

For path prediction, equation (III.5) is further discretized using an algorithm described in [Franklin, 1990], which features a modified version of the partial Taylor expansion method.

Kalman Filter For Vehicle Dynamics And External Disturbance Estimation

This section discusses the derivation of the model for Kalman filter implementation to estimate vehicle lateral velocity and external disturbances for the usage of predicting vehicle path. Assumptions are that lane marker locations from an on-board near-field sensor and measurements of yaw rate and front wheel steering angle will be available. The idea is to estimate the vehicle lateral velocity with the perceived variation of the previewed lane geometry and, subsequently, to estimate the disturbances acting on the vehicle with the knowledge of lateral velocity and yaw rate. The estimation of the lateral velocity is necessary for the disturbances to be observable.

For the purpose described above, the following model is developed;

$$\begin{Bmatrix} \dot{c}_{r_0} \\ \dot{c}_{r_1} \\ \dot{c}_{r_2} \\ \dot{v} \\ \dot{r} \\ \dot{e}_{F_r} \\ \dot{e}_{M_r} \end{Bmatrix} = \begin{bmatrix} 0 & u & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 2u & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{mu}(C_f + C_r) & \frac{1}{mu}(-bC_r + aC_f - mu^2) & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{I_z u}(aC_f - bC_r) & \frac{1}{I_z u}(a^2C_r + b^2C_f) & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} c_{r_0} \\ c_{r_1} \\ c_{r_2} \\ v \\ r \\ e_{F_r} \\ e_{M_r} \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ 0 \\ \frac{-C_f}{m} \\ \frac{-aC_f}{I_z} \\ 0 \\ 0 \end{Bmatrix} \delta \quad (III.6)$$

where $c_{r_0}, c_{r_1}, c_{r_2}$ are the coefficients of a 2nd order polynomial equation to model the previewed road geometry. In other words, the road geometry is modeled as

$$y_r(x_r) = c_{r_0} + c_{r_1}x_r + c_{r_2}x_r^2 \quad (III.7),$$

where (x_r, y_r) is a point on the curve, which is referred to a vehicle fixed frame (see Fig. III.1). The vehicle-fixed frame is defined according to the SAE standard coordinate system [Gillespie, 1992]. The other parameters are associated with the 2 DoF model introduced previously (see equation (III.1)). Essentially, this model contains two subsystems; one subsystem describes the relation between the vehicle kinematics and the perceived geometry variation and the other subsystem is a two DoF vehicle model which describes lateral vehicle dynamics. The reason for this construction is to use the perceived geometry variation to estimate the vehicle lateral velocity and subsequently use the estimated lateral

velocity and the measure yaw rate to assess external disturbances. With a Kalman filtering process, a better (smoother) perception of the geometry variation can be achieved to acquire better lateral velocity and external disturbances estimation.

Basically, a 2nd order polynomial equation assumes a constant curvature (see Fig. III. 1) ;

$$\kappa(l) = 2c_{r2} \quad (III.8)$$

where l is a spatial parameter along the curve, $\kappa(l)$ means the curvature at l , and c_{r2} is a constant value. As shown in Fig. III.1, the radius of the curve, ρ , is constant across the curve. With the assumption of small curvature, the tangential angle at l is obtained by integrating equation (III.8);

$$\vartheta(l) = \int_0^l 2c_{r2} d\lambda = c_{r1} + 2c_{r2}l \quad (III.9).$$

Furthermore, the y coordinate of the curve at a location l is obtained by;

$$y_r(l) = \int_0^l \vartheta(\lambda) d\lambda = \int_0^l c_{r1} + 2c_{r2}\lambda d\lambda = c_{r0} + c_{r1}l + c_{r2}l^2 \quad (III.10)$$

In equations (III.8) ~ (III.10), l can be replaced by x_r , the x coordinate of the curve at l , under the assumption of small curvature to obtain equation (III.7). For a more detailed discussion regarding to the derivation of equations (III.8) ~ (III.10), the reader is referred to [Dickmanns, 1986]. From equation (III.7), one can observe that

$$c_{r0} = y_r|_{x_r=0} \quad (III.11)$$

Thus,

$$\dot{c}_{r0} = \dot{y}_r|_{x_r=0} = (c_{r1}\dot{x}_r + 2c_{r2}x_r\dot{x}_r)_{x_r=0} - v = c_{r1}u - v \quad (III.12)$$

Furthermore, equation (III.9) shows that

$$c_{r1} = \vartheta|_{x_r=0} \quad (III.13)$$

Therefore,

$$\dot{c}_{r1} = \dot{\theta} \Big|_{x_r=0} - r = 2c_{r2}\dot{x}_r - r = 2c_{r2}\mu - r \quad (III.14)$$

Finally, because of the constant curvature assumption,

$$\dot{c}_{r2} = 0 \quad (III.15)$$

Besides the relation between the perceived lane geometry and vehicle kinematics, a 2 degree of freedom vehicle lateral dynamics model can be developed under the assumption that the tire forces are linearly proportional to the tire slip angle as in equation (III.1). Assume that the external disturbances are piecewise constant, then

$$\begin{aligned} \dot{e}_{F_y} &= 0 \\ \dot{e}_{M_z} &= 0 \end{aligned} \quad (III.16)$$

By combining equations (III.1), (III.12), (III.14), (III.15), and (III.16), equation (III.6) can be obtained.

Equation (III.6) is then discretized and is expressed as

$$\mathbf{x}(k+1) = \Phi\mathbf{x}(k) + \Gamma\mathbf{u}(k) \quad (III.17)$$

The implementation of the Kalman filter to estimate lateral velocity and external disturbances can be achieved with

$$\hat{\mathbf{x}}(k+1) = \Phi\hat{\mathbf{x}}(k) + \Gamma\mathbf{u}(k) + \mathbf{L}(\mathbf{y}(k) - \mathbf{H}\hat{\mathbf{x}}(k)) \quad (III.18)$$

where

$$\mathbf{y}(k) = \{c_{ro}(k) \quad c_{r1}(k) \quad c_{r2}(k) \quad r(k)\}^T \quad (III.19)$$

The coefficients (i.e., c_{ro}, c_{r1}, c_{r2}) are obtained by least square curve fitting of the 2nd order polynomial equation to the lane marker locations from the near-field sensor. Suppose a lane marker array (i.e., $(x_1, y_1) \dots (x_n, y_n)$) is available from the near-field sensor, the least square curve fitting is conducted as

$$\mathbf{C}_r = (\mathbf{X}_r^T \mathbf{X}_r)^{-1} \mathbf{X}_r^T \mathbf{Y}_r \quad (III.20)$$

where

$$\begin{aligned}
\mathbf{C}_r &= \{c_{r0} \quad c_{r1} \quad c_{r2}\}^T \\
\mathbf{X}_r &= \begin{bmatrix} 1 & x_{r1} & x_{r1}^2 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 1 & x_{rn} & x_{rn}^2 \end{bmatrix} \\
\mathbf{Y}_r &= [y_{r1} \quad \cdot \quad \cdot \quad y_{rn}]
\end{aligned} \tag{III.21}$$

In equation (III.18), \mathbf{H} is the output matrix, and \mathbf{L} is the Kalman Filter gain. The Kalman Filter gain is obtained from a process noise covariance matrix and a measurement error covariance matrix. To obtain these two matrices, a more sophisticated vehicle dynamics model is utilized to simulate the perception error of the sensors and the accuracy of the model in equation (III.6) [Venhovens, 1995]. The process noise also depends on the front wheel steering angle measurement error and the measurement noise also depends on the yaw rate measurement error. The usage of the perceived lane marker locations by the near-field sensor is important for accurate lateral velocity estimation and consequently accurate external disturbance estimation.

External Disturbance Characterization For Prediction

Once the external disturbances are estimated, they need to be characterized for prediction. This section discusses three different schemes to characterize the estimated values from the previous section.

One simple model for the external disturbances is to assume that they are piecewise constant:

$$\dot{e} = 0 \tag{III.22}$$

Thus, the disturbances in the path projection period are assumed the same as estimated at each time t . The second model characterizes the external disturbances by assuming they vary linearly with time. In other words,

$$e(t) = k_1 t + k_0 \tag{III.23}$$

To obtain the coefficients in equation (III.23) (i.e., k_1 and k_o), this equation is fitted to the estimated data in a previous time span in a least square sense. The goal here is to catch the trend of the signal. For this reason, the effects from the stochastic component need to be minimized. Since a main contribution to the stochastic component of the disturbances is wind force which typically has a bandwidth of 0.25 Hz [MacAdam, 1989], the data in a prior time span of 4 second are used, and equation (III.23) is fitted to the data to obtain the coefficients. In this case, the time at $t-4$ (suppose the current time is t) is reinitialized to 0, the current time is 4 and the prediction is to estimate the values from 4 to 8 second which correspond to t and $t+4$ second in real time axis. Suppose that the estimates of a disturbance from $t-4$ second to t second are

$$\mathbf{E} = \{e_o \quad e_1 \quad \dots \quad e_n\}^T \quad (\text{III.24})$$

and

$$\mathbf{t} = \{t_o \quad t_1 \quad \dots \quad t_n\}^T \quad (\text{III.25})$$

is the corresponding times for each estimation in the reinitialized time axis, then

$$\mathbf{k} = \{k_1 \quad k_o\}^T \quad (\text{III.26})$$

can be obtained by

$$\mathbf{k} = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{T}^T \mathbf{E} \quad (\text{III.27})$$

where

$$\mathbf{T} = \begin{bmatrix} t_o & t_1 & \dots & t_n \\ 1 & 1 & \dots & 1 \end{bmatrix}^T \quad (\text{III.28})$$

Beside characterizing the trend of the signal, the third scheme described below further characterizes the stochastic component of the disturbances. The stochastic component of the signal is obtained by subtracting the trend from the estimated data from the Kalman filter. In other words, suppose $e(t)$ is the estimated data and $e_{trend}(t)$ is the estimated trend from the previous scheme, the stochastic component is obtained by

$$d(t) = e(t) - e_{rend}(t) \quad (III.29)$$

Then, a stochastic model is used to characterize $d(t)$. Essentially, this assumes that $d(t)$ is the output from a filter with random white noise as the inputs;

$$\varphi(q^{-1})d(t) = w(t) \quad (III.30)$$

where q^{-1} is a backward shift operator and $\varphi(q^{-1})$ is a polynomial with q^{-1} as its parameter [Box, 1994]. Through various model selection tests (i.e., frequency plot comparison, Akaike's Final Prediction Error test, and residual test) [Ljung, 1987], it is found that the disturbances can be characterized using a 3rd order filter. Therefore, equation (III.30) has the following form;

$$(1 + a_1q^{-1} + a_2q^{-2} + a_3q^{-3})d(t) = w(t) \quad (III.31)$$

which is rewritten as

$$d_t = w_t - a_1d_{t-1} - a_2d_{t-2} - a_3d_{t-3} \quad (III.32)$$

where d_{t-n} represents $d(t - n\Delta t)$ and Δt is the discrete time span [Box, 1994]. To obtain the coefficients of the model (i.e., a_1, a_2, a_3), initial n data (i.e., $(d_0, d_1, d_2, \dots, d_n)$) is used to obtain the initial estimation of the model parameters and a recursive scheme [Franklin, 1990] is implemented to update the estimate every time a new data is obtained. Thus, the initial parameter estimate is

$$\hat{\theta} = (\Theta^T \Theta)^{-1} \Theta^T \mathbf{Z} \quad (III.33)$$

where

$$\begin{aligned} \hat{\theta} &= \{\hat{a}_1 \quad \hat{a}_2 \quad \hat{a}_3\}^T \\ \Theta &= \begin{bmatrix} -d_3 & -d_2 & -d_1 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ -d_{n-1} & -d_{n-2} & -d_{n-3} \end{bmatrix} \\ \mathbf{Z} &= [d_4 \quad \cdot \quad \cdot \quad d_n]^T \end{aligned} \quad (III.34)$$

To implement recursive least square algorithm, denote

$$\mathbf{P} = (\Theta^T \Theta)^{-1} \quad (\text{III.35})$$

Suppose, a new estimated disturbance value is provided from the Kalman filter, d_{n+1} , then a new parameter estimate can be calculated as

$$\hat{\theta}_{n+1} = \hat{\theta}_n + \mathbf{K}(z - \phi' \hat{\theta}_n) \quad (\text{III.36})$$

where

$$\begin{aligned} \phi &= \{-d_n \quad -d_{n-1} \quad -d_{n-2}\}^T \\ z &= d_{n+1} \\ \mathbf{K} &= \mathbf{P}_n \phi (1 + \phi' \mathbf{P}_n \phi)^{-1} \end{aligned} \quad (\text{III.37})$$

and P is updated as

$$\mathbf{P}_{n+1} = (\mathbf{I}_{3 \times 3} - \mathbf{K} \phi') \mathbf{P}_n \quad (\text{III.38})$$

The same procedure is conducted repeatedly to update the parameter estimates. A derivation of this procedure for general models is described in [Franklin, 1990]. At each time step, t , the prediction of the future stochastic component is calculated from

$$\hat{d}_{t+l} = -\hat{a}_1 \hat{d}_{t+l-1} - \hat{a}_2 \hat{d}_{t+l-2} - \hat{a}_3 \hat{d}_{t+l-3} \quad (\text{III.39})$$

however, if $l \leq 1$, the following equation is used ;

$$\hat{d}_{t+l} = -\hat{a}_1 d_{t+l-1} - \hat{a}_2 d_{t+l-2} - \hat{a}_3 d_{t+l-3} \quad (\text{III.40})$$

An estimate of the disturbance magnitude at a future time is then the combination of the estimates from the trend prediction (i.e., 2nd scheme) and the stochastic model prediction.

Predicted Path Uncertainty Characterization

The predicted vehicle path is characterized by a 2nd order polynomial equation. This section is to develop a model to generate uncertainty ranges for the polynomial coefficients. The inputs are statistical characteristics of the measurement/prediction errors of the lateral velocity, yaw rate, and front wheel steering angle, which are assumed to be available from off-line characterization. The results are the covariance of the possible deviation of the coefficients from the actual values.

Suppose the discretized form of equation (III.5) is expressed as

$$\mathbf{x}_p(k+1) = \Phi_p \mathbf{x}_p(k) + \Gamma_p \mathbf{u}_p(k) \quad (\text{III.41})$$

where the subscript p means that its a model for prediction. Furthermore, assume that there is an initial condition measurement error, $\Delta \mathbf{x}_{p_0}$, and errors in the inputs, $\Delta \mathbf{u}_p(k)$. Then the prediction error at step k is

$$\Delta \mathbf{x}_p(k) = \Phi_p^k \Delta \mathbf{x}_{p_0} + \sum_{i=1}^k \Phi_p^{i-1} \Gamma_p \Delta \mathbf{u}_p(k-i) \quad (\text{III.42})$$

Thus, the autocorrelation of $\Delta \mathbf{x}_p(k)$ and $\Delta \mathbf{x}_p(k-m)$ is

$$\begin{aligned} & E[\Delta \mathbf{x}_p(k) \Delta \mathbf{x}_p^T(k-m)] \\ &= \Phi_p^k E[\Delta \mathbf{x}_{p_0} \Delta \mathbf{x}_{p_0}^T] (\Phi_p^{k-m})^T + \sum_{j=1}^k \sum_{i=1}^{k-m} \Phi_p^{j-1} \Gamma_p E[\Delta \mathbf{u}_p(k-j) \Delta \mathbf{u}_p^T(k-m-i)] (\Phi_p^{i-1} \Gamma_p)^T \end{aligned} \quad (\text{III.43})$$

where $E[.]$ represent the expected value of a random process. Generally, the influence from $\Delta \mathbf{x}_{p_0}$ is more substantial than that from $\Delta \mathbf{u}_p(k)$; thus the second term on the right hand side of equation (III.43) can be ignored for an initial rough estimate. From equation (III.43), the covariance of the lateral displacement prediction error, $E[\Delta y_r(k) \Delta y_r(k-m)]$, can be obtained. Since the coefficients of the polynomial equation for the predicted path is obtained by

$$\mathbf{C}_v = \Psi_v \mathbf{Y}_v \quad (\text{III.44})$$

where

$$\begin{aligned} \Psi_v &= (\mathbf{X}_v^T \mathbf{X}_v)^{-1} \mathbf{X}_v^T \\ \mathbf{C}_v &= \{c_{v_0} \quad c_{v_1} \quad c_{v_2}\}^T \\ \mathbf{X}_v &= \begin{bmatrix} 1 & x_{v_1} & x_{v_1}^2 \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ 1 & x_{v_n} & x_{v_n}^2 \end{bmatrix} \\ \mathbf{Y}_v &= \{y_{v_1} \quad \cdot \quad y_{v_n}\}^T \end{aligned} \quad (\text{III.45})$$

and $(x_{v,n}, y_{v,n})$ is a discrete point in the predicted vehicle path, the covariance of the coefficients can be obtained by

$$E[\Delta C, \Delta C,^T] = \Psi, E[\Delta Y, \Delta Y,^T] \Psi,^T \quad (\text{III.46})$$

where the $E[\Delta Y, \Delta Y,^T]$ was obtained previously.

Simulations And Discussion

This section presents a simulation result to show the validity of the linearized path projection. Then, simulation results for lateral velocity and external disturbances estimation with the steady state Kalman filter are discussed. The Kalman filter gain is designed for a typical highway geometry (i.e., typical curvature) and road surface featuring smooth asphalt (which accounts for 80% of the US highway [Sayer, 1986]) and a forward vehicle speed of 90 Km/h. The simulations show the performance of the Kalman filter for tracking different levels of lateral velocity and external disturbances. For this purpose, various external disturbances are considered. Then the same Kalman filter gain is applied to various other situations (e.g., road geometry, road surface, and vehicle velocity) to study the possibility of using only one Kalman filter gain for typical highway driving. Subsequently, different disturbance characterization schemes for predicting future disturbance values are discussed. The improvements in TLC assessment achieved with external disturbance characterization are also discussed. Finally, vehicle path prediction uncertainty is studied, with the uncertainty characterization scheme, in terms of its contribution to the TLC uncertainty.

To validate the linearized path projection model (i.e., equation (III.5)). Two TLC simulations under the same driving conditions (i.e., same vehicle dynamics and roadway geometry) are conducted using the nonlinear path projection equation (i.e. equation (III.2)) and the linearized model. Fig. III.2 shows the results of these two simulations, in which the positive values represent a tendency for left land boundary crossing and vice versa. From Fig. III.2 , one can see that the linearized model is adequate for path projection.

To study the performance of the Kalman filter for different external disturbances, three simulations are conducted. Among these simulations, the road geometry, road surface, and vehicle velocity are the same and the Kalman filter gain is designed for such conditions. However, they have different superelevation and wind force. The road geometry has a straight section which transitions to a curve, the road surface features a road roughness typical of smooth asphalt (which accounts for about 80 % the typical US highway), and the vehicle velocity is 90 Km/h. Figures III.3 to III.5 show the simulation results. The simulation shown in Fig. III.3 has a large superelevation and a strong wind force. Fig. III.4 shows results for a large superelevation but a small wind force. The results in Fig. III.5 are for a small superelevation and a small wind force. Then, the estimated lateral velocity and external disturbances are applied to the path prediction under the assumption that the disturbance magnitude holds constant during the path projection period. The results are compared with the predictions which use no estimation scheme. For the latter predictions, the vehicle lateral velocity and external disturbances are assumed to be zero. The comparison is shown in Fig. III.6 in terms of the resulting time to lane crossing estimation error from these two different path predictions in different cases. A detailed discussion of the time to lane crossing calculation methods is given in [Lin and Ulsoy, 1995]. The results show that for the first two cases, the time to lane crossing estimation error is significantly improved by the estimated values. However, for the third case, no improvement is seen. This is because under this condition (i.e., a mild external disturbance), the lateral velocity and external disturbance is already very small and the assumption of zero values is adequate.

A fixed Kalman filter gain which is designed for smooth road surface and a particular highway geometry (a straight section transitions to a pure curve with a clothoid in between) is next applied to other highway driving conditions to study the possibility of using one fixed Kalman filter gain for typical highway driving. The first simulation has the same conditions as in Fig. III.4 except that the road surface is much rougher than the

smooth asphalt (usually accounts for local roadway or some unrepaired highway). The result of the estimation is shown in Fig. III.7. The following two simulations study the effect of the road geometry (which also affect the lane marker location perception). The results in Fig. III.8 are for the same conditions as in Fig. III.4 but the radius of the curved section is much smaller (281 m in Fig. III.8 versus 380 m in Fig. III.4). On the other hand, Fig. III.9 has only a straight section during the simulation. Fig. III.10 shows the time to lane crossing estimation errors in these three simulations. Figure III.7 and the corresponding time to lane crossing error in Fig. III.10 show that the nominal Kalman filter gain works successfully even when the lane marker location perception is significantly affected by vehicle vibration. Similar results are obtained for straight road driving. However, if the vehicle encounters a relatively tight curve, such as the one shown in Fig. III.8, the performance of the Kalman filter is degraded. However, the time to lane crossing value is still better than that with no Kalman filter estimation.

Finally, the performance of the Kalman filter when the forward vehicle velocity is different from the nominal speed is studied. For this purpose, the Kalman filter gain designed for 90 km/h is applied to a vehicle dynamics simulation with a forward velocity of 110 km/h. The result is compared with the estimation with a Kalman filter gain designed for the 110 Km/h simulation. Fig. III.11 shows the result of this study, which shows similar estimation results for the two systems. Thus, the Kalman filter gain is not sensitive to the forward velocity variation.

After the external disturbances are estimated, the disturbance characterization schemes discussed in a previous section are applied to characterize the disturbances and to predict future values, which are then considered in the path projection. The results are shown in Fig. III.12, in which the perceived TLC's are compared with the true TLC. In order to see the benefit of disturbance characterization, the result with no disturbance characterization is also included. Fig. III.12 has a vehicle driven on a straight road and approaching a curve section under the effect of a substantial side wind force. When $t = 16$

sec., a fixed steering wheel is assigned to simulate a drowsy driver falling asleep. This situation becomes apparent at about $t = 21$ sec. and TLC starts to decrease to zero from that point on. Fig. III.12 shows that disturbance characterization significantly improves the perceived TLC; however, there is no substantial difference between the three characterization schemes. This is mainly due to the fact that the trend of the disturbance varies slowly; a piecewise constant model introduces only limited differences from the linear variation model. Furthermore, additional characterization of the stochastic component provide a better result than considering the steady disturbance trend only. However, the benefit is limited, and is not visually obvious in Fig. III.12. However, since the piecewise constant disturbance model scheme is satisfactory, no further study of modeling the stochastic component was conducted.

Finally, vehicle path prediction uncertainty is studied in terms of its contribution to the TLC uncertainty. Due to the fact that the influence on the predicted path uncertainty from the measurement/estimation errors of the vehicle lateral velocity, yaw rate, and front wheel steering angle (i.e. $\Delta \mathbf{x}_{p_0}$ in equation (III.42)), are more substantial than that from the prediction errors due to future disturbance (i.e. $\Delta \mathbf{u}_p(k)$ in equation (III.42)), only the uncertainty introduced by $\Delta \mathbf{x}_{p_0}$ is considered. Table III.1 shows the results, in which, $\sigma_{\Delta v}, \sigma_{\Delta r}, \sigma_{\Delta \delta}$ represent standard deviations of the measurement/estimation errors of the vehicle lateral velocity, yaw rate, and front wheel steering angle respectively. Furthermore, in these simulations, the road geometry perception is accurate. Thus, the contribution to the TLC uncertainty comes from vehicle path prediction uncertainty only. The results show that the TLC uncertainty values are large when the front wheel steering angle measurement errors are large. Thus, an accurate measurement on the front wheel steering angle is required or the results may not be useful for the proposed active safety system. If satisfactory measurement errors can not be obtained, an effective algorithm must be developed to reduced the uncertainty (e.g., check the consistency of the obtained TLC).

Table III.1 : Time To Lane Crossing Uncertainty Of Some Vehicle Path Prediction Uncertainties With A Nominal Time To Lane Crossing Of 3.0 sec

	$\sigma_{\Delta v} = 0.02(m/s)$	$\sigma_{\Delta v} = 0.02(m/s)$	$\sigma_{\Delta v} = 0.02(m/s)$
	$\sigma_{\Delta r} = 0.01(deg/s)$	$\sigma_{\Delta r} = 0.03(deg/s)$	$\sigma_{\Delta r} = 0.03(deg/s)$
	$\sigma_{\Delta \delta} = 0.01(deg)$	$\sigma_{\Delta \delta} = 0.03(deg)$	$\sigma_{\Delta \delta} = 0.05(deg)$
TLC Uncertainty $\sigma_{TLC} (sec)$	0.19	0.56	0.93

Summary And Conclusions

This chapter discusses a future vehicle path prediction scheme and an algorithm to characterize the prediction uncertainty. For the path prediction, a linearized model is utilized. Its validity is discussed. Furthermore, for an accurate path prediction, a steady state Kalman filter which uses perceived lane marker locations, obtained from an on-board near field sensor, is developed to simultaneously estimate vehicle lateral velocity and external disturbances. The possibility of using one fixed Kalman filter gain for typical highway driving is also investigated. Subsequently, different schemes for disturbance characterization to predict future disturbance inputs to the vehicle are discussed. Finally, an algorithm to characterize path prediction uncertainty is proposed at the end of the chapter, which assumes that the statistical characteristics of the measurement/estimation errors of the vehicle lateral velocity, yaw rate, and front wheel steering angle are available from off-line characterization.

Simulation result show that the linearized model is adequate for TLC calculation, which allows a more efficient computation of the TLC. Simulation results also show that, by applying the estimation scheme discussed in this chapter, the predicted future vehicle path, and consequently the time to lane crossing, are significantly improved for conditions where the vehicle is subjected to significant external disturbances. However, when the external disturbances are small, an assumption of negligible vehicle lateral velocity and external disturbance is adequate.

It is also shown that a Kalman filter gain which is designed for a typical road geometry (i.e., curvature), road surface (i.e., smooth asphalt accounting for 80% of the typical US highway), and forward vehicle speed at 90 Km/h can accommodate a much rougher surface (which introduces severe vehicle vibration), a forward speed variation up to 110 Km/h, and a road curvature smaller than the designated curve. However, another Kalman filter gain (or gain scheduling) is needed when the vehicle encounters a much tighter curve (e.g., an exit ramp on typical highway).

On the other hand, study results indicate that a piecewise constant model is the most appropriate for predicting future disturbance values. The main reason being the variation of the future disturbance trend is slow and prediction of the stochastic component can not be satisfactorily achieved with a simple order model. Finally, the TLC uncertainties corresponding to some path prediction uncertainties are studied. The results show that accurate front wheel angle measurement is required to limit the TLC uncertainty to a useful level.

Usage of lateral accelerometers rather than yaw rate sensors and a more complex model for the external disturbances for the Kalman filter may be suitable areas for future research. The prediction error for future disturbance inputs can also be studied to improve the prediction of the path prediction uncertainty.

References

- Box G.E.P., Jenkins G.M., Reinsel G.C., *Time Series Analysis-Forecasting And Control*, Prentice-Hall Inc., 1994.
- Dickmanns E.D., Zapp A., "A Curvature-Based Scheme For Improving Road Vehicle Guidance By Computer Vision", *Proc. of SPIE on Mobile Robots*, Vol. 727, pp. 161-168, 1986.
- Franklin G.F., Powell J. D., Workman M.L., *Digital Control of Dynamic Systems*, New York: Addison-Wesley Publishing Company; 1990.
- Gillespie T.D., *Fundamental of Vehicle Dynamics*, Society of Automotive Engineers, Inc., 1992.
- Godthelp H., Milgram P., Blaauw G., "The Development of a Time-Related Measure to Describe Driving Strategy", *Human Factors*, 26(3), pp. 257-268, 1984.
- Hou M. and Muller P.C., "Design Of Observers For Linear Systems With Unknown Inputs", *IEEE Trans. on Automatic Control*, vol. 37, no 6, June 1992.
- LeBlanc D.J., Venhovens P.J.Th., Pilutti T.E., Lin C. F., Ervin R.D., Ulsoy A.G., MacAdam C.C., "A Warning And Intervention System For Preventing Inadvertent Road Departure", *14th IAVSD Symposium on the Dynamics of Vehicles on Roads and Tracks*, Ann Arbor, MI, Aug. 1995.
- Lin Chiu-F. and Ulsoy A. Galip, "Calculation Of The Time To Lane Crossing And Analysis Of Its Frequency Distribution", *Proc. of American Control Conference*, Seattle, WA. 1995.
- Ljung L., *System Identification : Theory For The User*, Prentice-Hall Inc., 1987.
- MacAdam Charles C., "The Interaction of Aerodynamic Properties and Steering System Characteristics on Passenger car Handling", *Proceedings of the 11th IAVSD Symposium of the Dynamics of Vehicles on Roads and on Tracks*, Kingston, Ont., Canada, 1989.
- Maquin D., Gaddouna B., and Ragot J., "Estimation Of Unknown Inputs In Linear Systems", *Proc. of the American Control Conference*, Baltimore, June 1994.
- Okuno A. Fujita K., and Kutami A., "Visual Navigation Of An Autonomous On-Road Vehicle: Autonomous Cruising On Highways", *Vision-Based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Park Y. and Stein J.L., "Closed-Loop, State And Input Observer For Systems With Unknown Inputs", *Int. Journal of Control*, 48 (3), pp. 1121-1136, 1988.
- Sayer M., "Characteristics Of The Power Spectral Density Function For Road Roughness", *Symposium Of Simulation And Control Of Ground Vehicle*, ASME, 1986.
- Venhovens Paul J. Th., *Manual For An Road-Departure Prevention System Simulation Program*, The University of Michigan Transportation Research Institute, 1995.

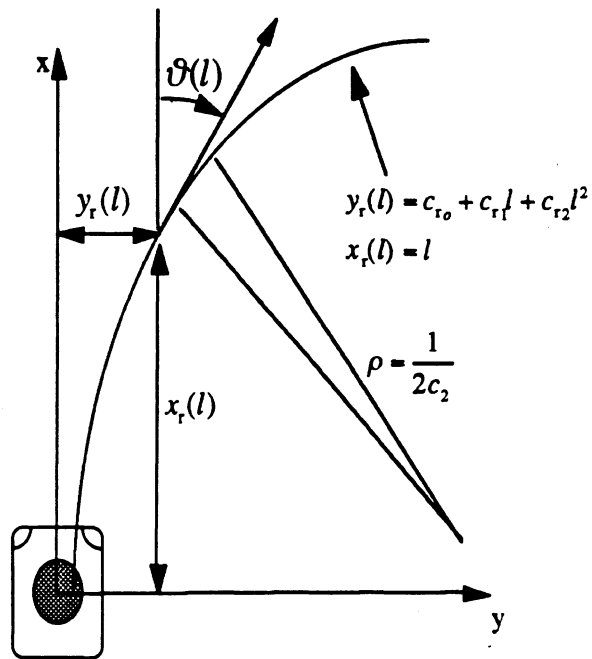


Figure III.1 : Geometrical Relation For A 2nd Order Polynomial Equation

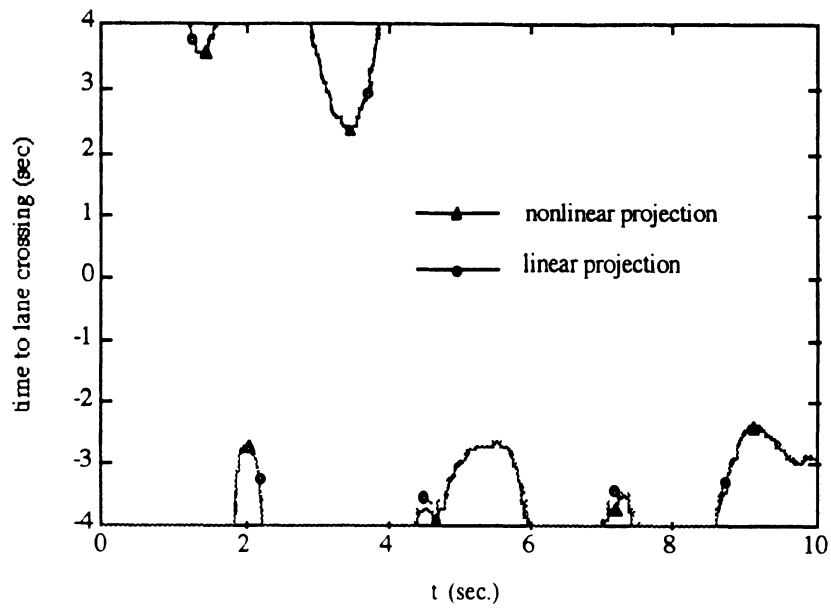


Figure III.2 : Comparison Of The Nonlinear And Linear Equations For Vehicle Path Projection In TLC Calculations

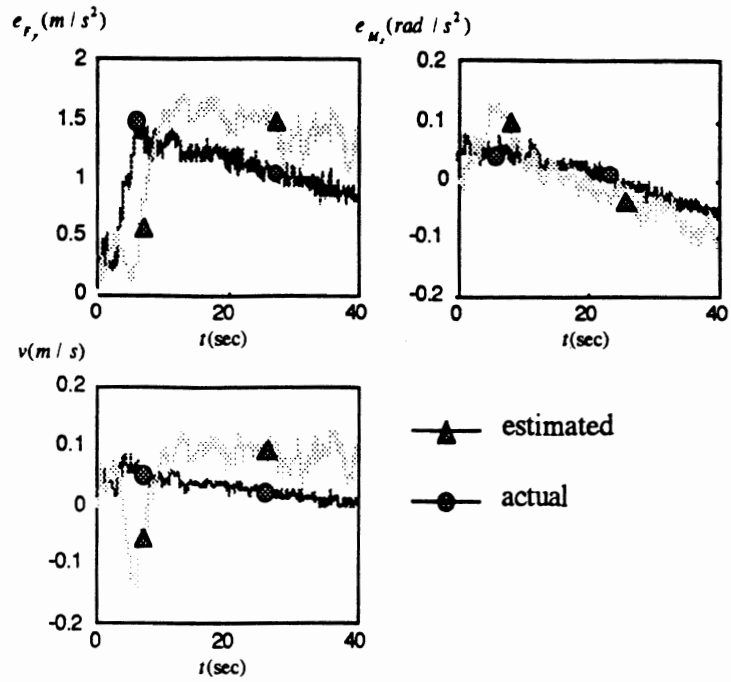


Figure III.3 : Lateral Velocity And External Disturbances Estimation In A Large Superelevation And Strong Wind Condition

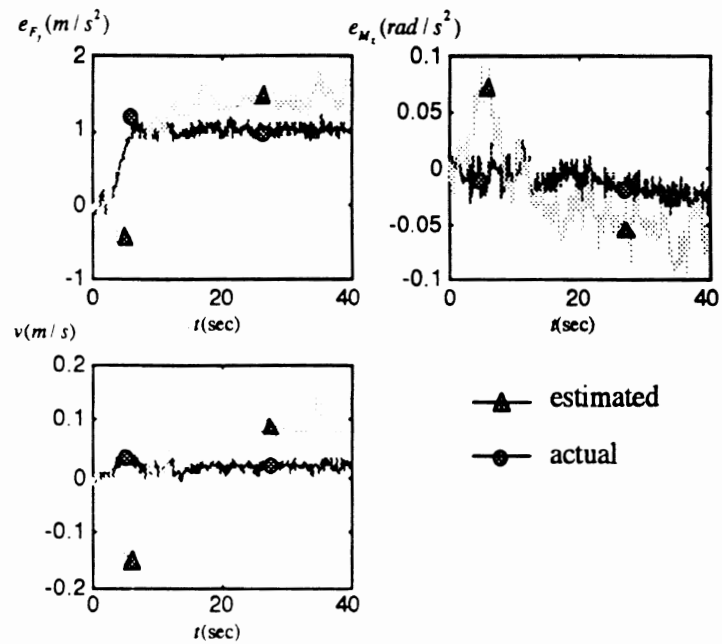


Figure III.4 : Lateral Velocity And External Disturbances Estimation In A Large Superelevation And Small Wind Condition

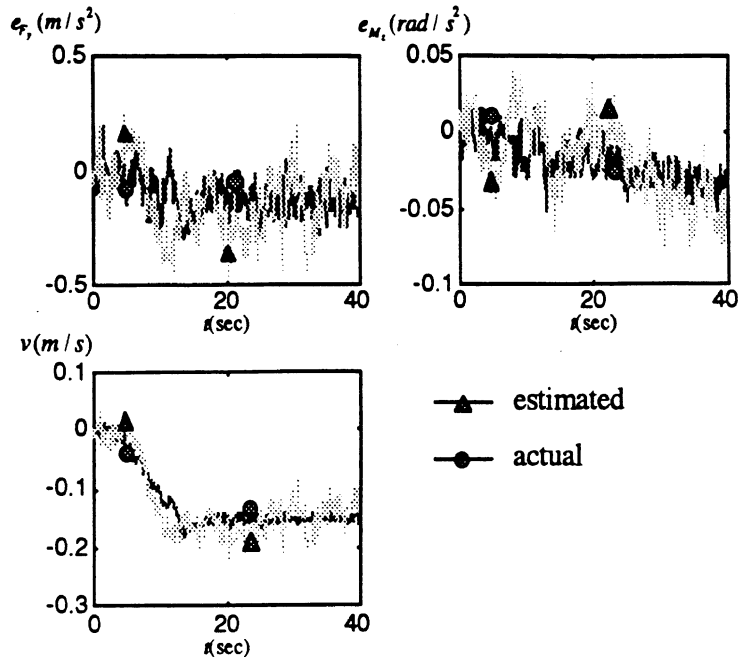


Figure III.5 : Lateral Velocity And External Disturbances Estimation In A No Superelevation And Small Wind Condition

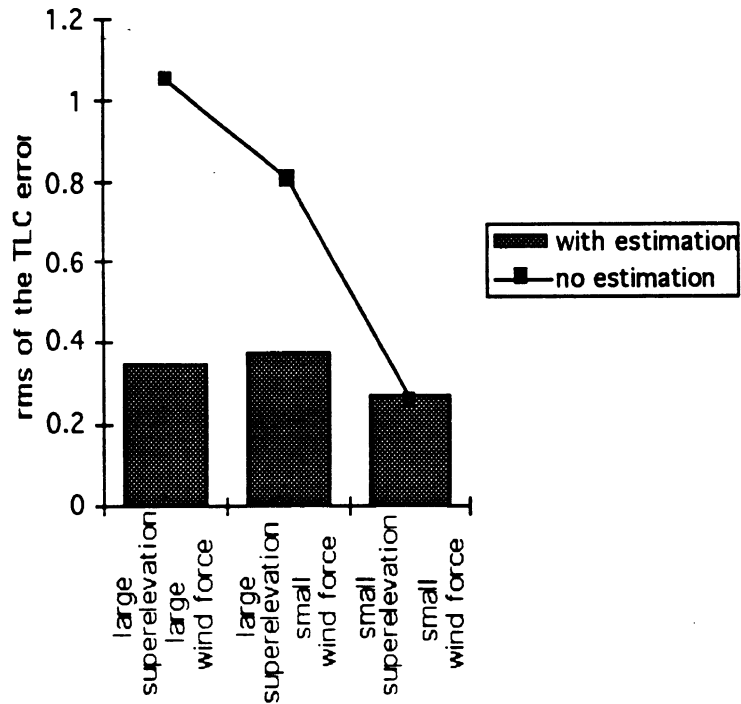


Figure III.6 : Time To Lane Crossing Estimation Errors Under Different External Disturbances

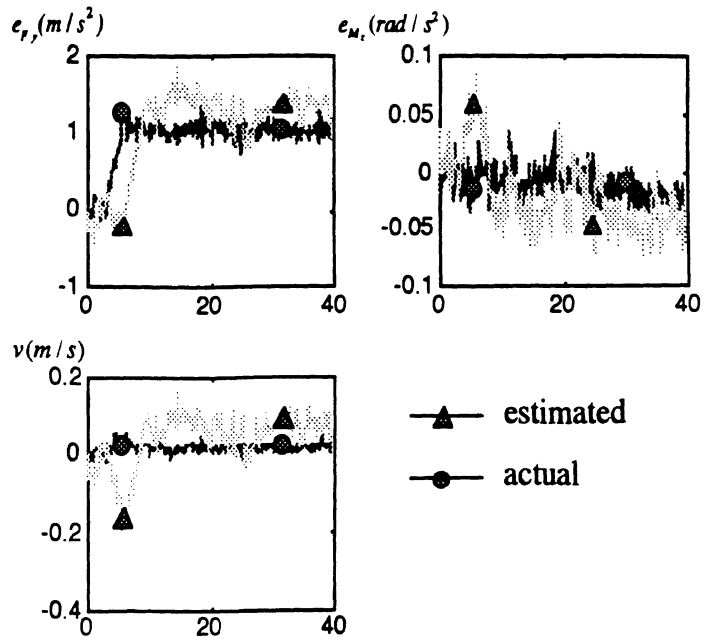


Figure III.7 : Lateral Velocity And External Disturbances Estimation Under Significant Vehicle Vibration Effect

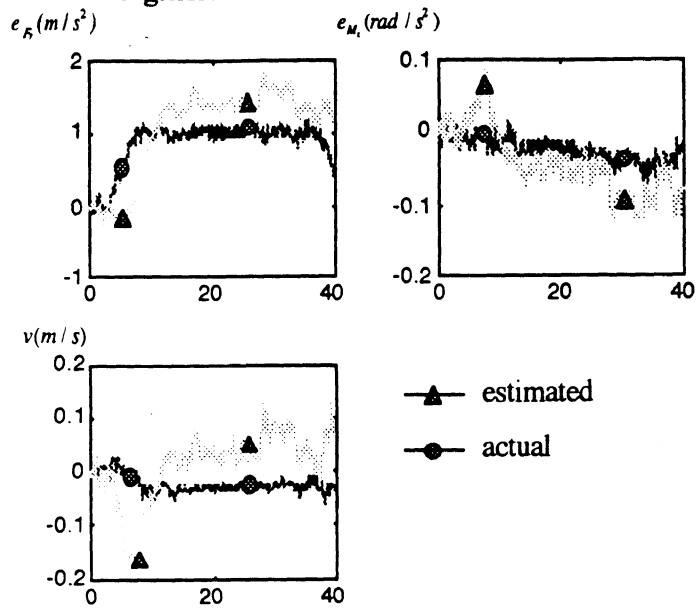


Figure III.8 : Lateral Velocity And External Disturbances Estimation In Tight Curve Driving

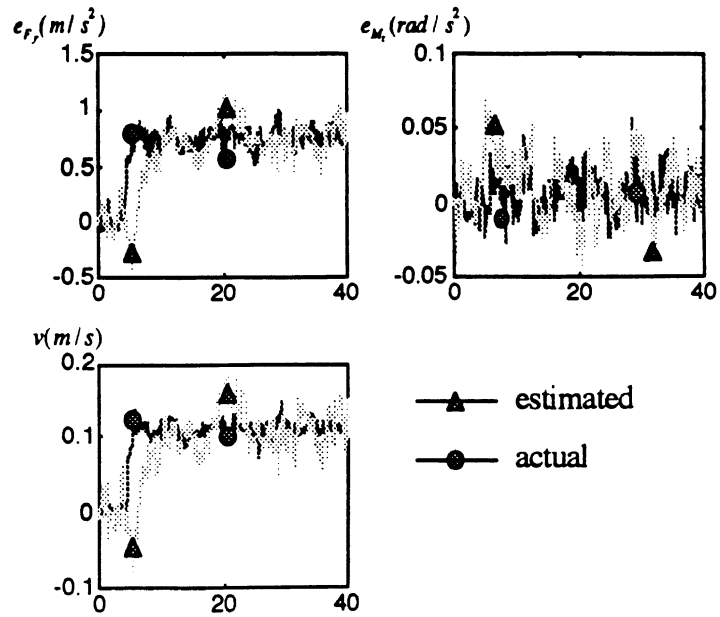


Figure III.9 : Lateral Velocity And External Disturbances Estimation In Straight Section Driving

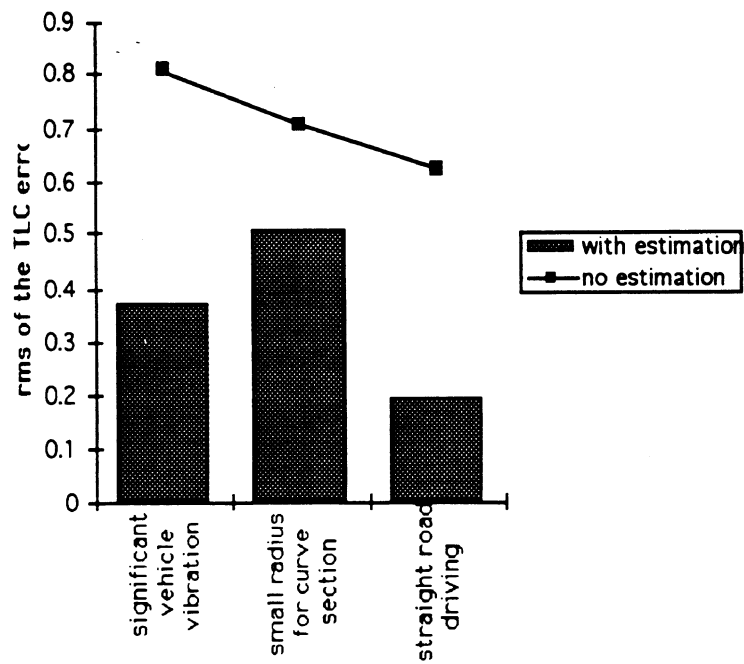


Figure III.10 : Time To Lane Crossing Errors Under Different Effects

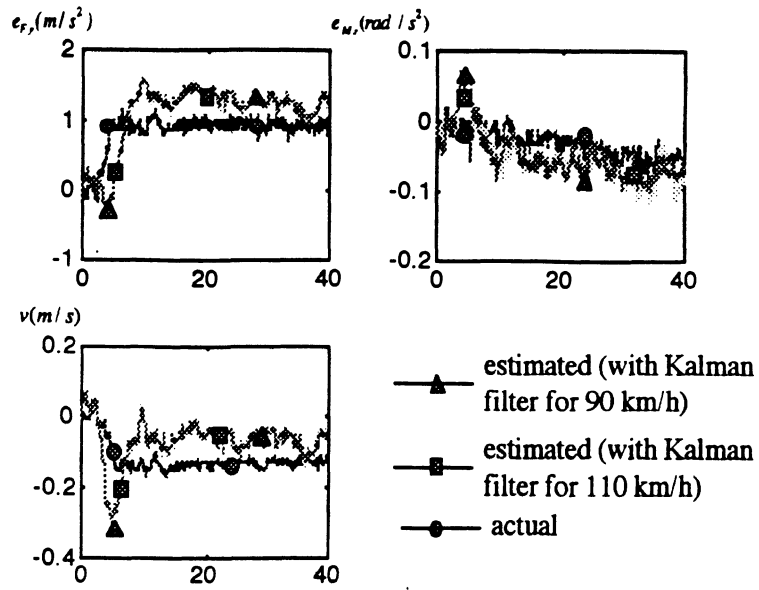


Figure III.11 : Lateral Velocity And External Disturbances Estimation Under Vehicle Forward Velocity Variation Effect

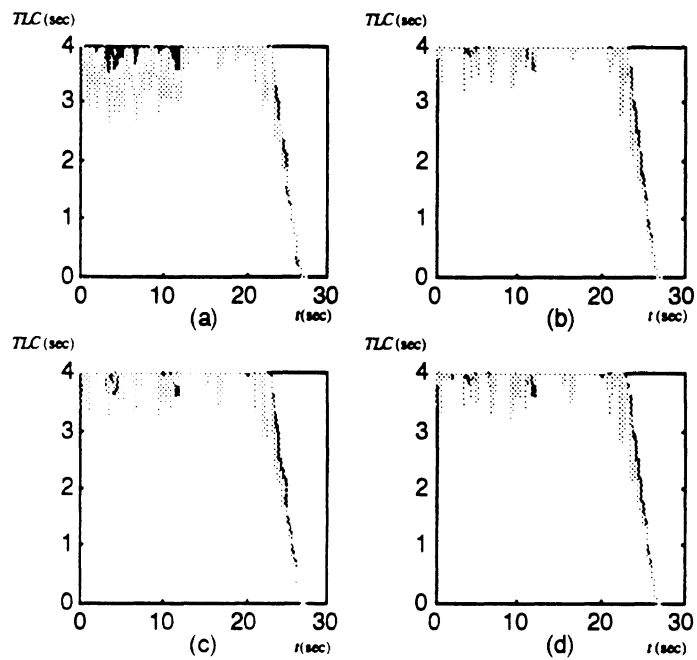


Figure III.12 : Comparison Of Different Disturbance Characterization Model For Future Value Prediction; (a) No Disturbance Estimation, (b) Piecewise Constant Model, (c) Linear Model, (d) Linear Model Plus Stochastic Model

CHAPTER IV

LANE GEOMETRY PERCEPTION AND THE CHARACTERIZATION OF ITS ASSOCIATED UNCERTAINTY

Abstract

This chapter presents a process for down range road geometry reconstruction from measurements (e.g., using computer vision) of lane markings. Here, lane markers refer to imaginary points along the lane edges (usually referred to the painted white lines). Lane marker information from both near-field and far-field cameras is assumed. Subsequently, an algorithm to characterize the road geometry perception uncertainty is proposed. The down range road geometry reconstruction is composed of three steps; 1) acquisition of lane marker locations (which is assumed to be available in this study), 2) perception range extension to alleviate the field of view problem associated with using a single-far-field sensor, and 3) road geometry modeling to generate a model for the down range road geometry. Two steady state Kalman filters (a 3rd order Kalman filter and a 4th order Kalman filter) and a least square curve fitting scheme are developed for road geometry modeling. For the least square curve fit and the 3rd order Kalman filter, the down range road geometry is expressed with a 2nd order polynomial equation. For the 4th order Kalman filter, a 3rd order polynomial equation is used to model the road geometry. Simulations are conducted with these three separate road modeling schemes and the results are compared in terms of their ability to reconstruct the down range road geometry. These results show that the Kalman filter approach offers significant advantages over least square curve fitting in smooth road geometry. However, a least square curve fit gives better results in transitions (between different curves). This is because a Kalman filter always has a lag in variable estimation due to its filtering effect. Furthermore, simulation results show

that a 4th order Kalman filter has better performance than a 3rd order Kalman filter because it assumes a road model which allows the curvature to linearly vary along the curve (a 3rd order Kalman filter assumes constant curvature for road geometry). Finally, an algorithm to characterize the road geometry perception uncertainty is proposed and validated. Road geometry perception uncertainty for typical highway driving is simulated using the proposed algorithm.

Introduction

This chapter discusses the problems of road geometry perception, by generating a model for the down range road geometry and characterizing the associated uncertainty, with lane marker locations acquired through image processing. Characterization of the down range road geometry is necessary for an active safety system being developed at the University of Michigan to estimate the “lane tracking margin” of a vehicle, and to prevent run-off-road accidents [LeBlanc *et.al.*, 1995]. For this purpose, an algorithm for down range road geometry reconstruction is developed. The procedure for down range road geometry reconstruction in the proposed active safety system is to first acquire lane marker locations from the image. Then a perception range extension algorithm is applied to assess lane marker locations outside the far-field sensor field of view and the result is an extended lane marker array. Subsequently, the extended lane marker array is used by a road geometry modeling module to reconstruct the road geometry. The results of the road geometry reconstruction are two polynomial equations representing right and left road edges. In the current study, it is assumed that lane marker locations will be available from an on-board sensing system and the focus is on the subsequent processes (i.e., development of a perception range extension algorithm and road geometry modeling algorithm). Furthermore, since an understanding of the reliability of the road geometry acquisition is also important, so that the reliability of the “lane tracking margin” can be assessed, uncertainty characterization of the perceived road geometry is then conducted

after the road geometry reconstruction. The results are the covariance of the possible deviation of the perceived polynomial coefficients from coefficients which represents the true geometry.

The reason for including a perception range extension algorithm is that the field of view of the sensing system is limited and, under certain circumstances, the desired objects (i.e., lane markers) lie outside of the perception range of the sensors. Such a situation can lead to a lack of sufficient information for road geometry reconstruction and subsequently, affect the robustness of the proposed active safety system. For this purpose, an algorithm to extend the down range road geometry perception range, by utilizing information from a 2nd near-field camera and the fact that the lane width is constant, is proposed.

On the other hand, since least square curve fitting and Kalman filtering approaches are typically used for road geometry modeling, three algorithms including 1) a 3rd order Kalman filter modeling the road geometry with a 2nd order polynomial equation, 2) a 4th order Kalman filter modeling the road with a 3rd order polynomial equation, and 3) a least square curve fitting scheme modeling road geometry with a 2nd order polynomial equation are developed and then compared in terms of their ability to recover road geometry.

In the research literature, two schemes for road geometry modeling are typically considered: (1) least square curve fitting and (2) Kalman filtering. The down range road geometry is usually represented by a low-order polynomial equation, or a set of polynomial equations. The main differences between specific implementations are typically in the order of the polynomial equations. Essentially, a first order polynomial equation assumes the road is straight, a 2nd order polynomial equation assumes constant curvature for the road geometry, and a 3rd order polynomial equation allows the curvature to vary linearly [Dickmanns, 1988]. The lowest polynomial order which is adequate for describing the previewed roadway in the field of interest is typically chosen; a large area generally requires a higher order. Both the least square curve fit and Kalman filtering approaches find an optimal polynomial equation (with a pre-selected order) which minimizes the variance

between the equation and the acquired lane marker locations. There are two ways to do least square curve fitting. One way is to simply fit a polynomial equation to the lane marker array contained in the current image. The other utilizes the knowledge of the vehicle kinematics to transform all the lane marker locations from several images to a global reference frame to create a lane marker array which contains historical information for curve fitting ([Kenue, 1989], [Thorpe, 1992]). A Kalman filter also utilizes the information from the previous images and the vehicle kinematics [Dickmanns, 1988]. However, the Kalman filter approach uses a recursive algorithm which is more time and memory efficient than the second least square curve fitting scheme ([Dickmanns, 1988], [Broida, 1986], [Rives, 1986], [Kriegman, 1989]). Therefore, for the proposed active safety system, the first least square curve fitting scheme (i.e., using only the information contained in the current image) and the Kalman filter are considered as candidates and then evaluated under different conditions (e.g., road geometry and road unevenness).

For the problem raised by the limited sensor field of view, a multiple-far-field-sensor system is usually implemented such that the area of interest can be covered ([Graefe, 1992], [Tsugawa, 1984]). For a single-far-field-sensor system, as will be implemented on the proposed active safety system, an algorithm is necessary to effectively extend the range of the perception. In this study, an algorithm, which uses the information provided by a single-far-field-sensor and a near-field-sensor and knowledge on highway construction, is proposed to improve the perception range.

Furthermore, the perceived down range road geometry deviates from the true geometry due to the limitations of the sensing system (e.g., resolution of the sensors, insufficient knowledge of sensor orientation). Since the perception errors in the down range roadway geometry are not deterministic, the possible deviation of the perceived geometry from the true geometry is usually characterized by an uncertainty range. The issue of road geometry perception uncertainty has never been studied. This is because the perception error in the range of interest of an autonomous vehicle system is usually small,

and its effect on the system performance can be successfully reduced by the control system. Most of the studies on perception uncertainty characterize the uncertainty range of a spatial point location ([Kriegman, 1989],[Matthies, 1987]), as needed by Kalman filters for mobile robot guidance. Such studies are for indoor environments; therefore, the uncertainty characterization does not consider vehicle pitch and roll effects since they are relatively small in this situation. In such studies, the uncertainty range is usually modeled in a statistical form and typically as Gaussian [Durrant-Whyte, 1988]. Other methods for uncertainty characterization are scalar weights [Moravec, 1980] and uncertainty manifolds [Brooks, 1985]. The Gaussian distribution is widely used because of its convenience for analysis.

The next section provides a detailed explanation of the procedure for down range road geometry reconstruction. The perception range extension algorithm is discussed in the subsequent section. Then, the least square curve fit, the 3rd order Kalman filter, and the 4th order Kalman filter for road geometry modeling are introduced. The discussions on road geometry modeling are followed by an introduction of an algorithm to characterize the perception uncertainty. These are then followed by a section containing comparative simulation results and discussions for the Kalman filter and least square curve fit. The same section also contains validation of the proposed uncertainty characterization algorithm and simulation results for typical geometry perception uncertainty in highway driving. The final section contains the main conclusions of this study.

Down Range Road Geometry Reconstruction

The down range road geometry reconstruction task is to generate models for the road boundaries in the vehicle previewed scene up to a range of interest. In this study, polynomial equations are chosen to model the down range road geometry for their simplicity to represent the road geometry and consequently their convenience to manipulate. Different orders (i.e., 2nd order and 3rd order) of the polynomial equations are studied to

see their adequacy for road modeling for “lane tracking margin” assessment purpose. The reconstruction finds the corresponding polynomial coefficients for the down range road boundaries.

The down range road geometry reconstruction involves three steps (see Fig. IV.1). The first step has a sensing system which captures the desired objects and locates these objects in the reference frame. Generally, the desired objects are imaginary points, referred to here as lane markers, along the painted white lines. In the proposed active safety system a sensor provides lane marker locations from the right and left lane edges in the far field and a second sensor provides lane marker location from the right lane edge in the near field.

Due to the field of view constraint, the perceived far field lane marker locations from each edge are not always sufficient to generate accurate models for the corresponding curves. Thus, an algorithm to extend the perception range is necessary. To extend the perception range along the right edge, the second step in the module (see Fig. IV.1) uses the information from the far field left edge and the near field right edge to extend the perception range. The result is a modified lane marker array for the right edge. A more detail explanation is included in the next section.

Finally, the modified lane marker array is fitted to a polynomial equation (either a 2nd order polynomial equation or a 3rd order polynomial equation). Three different algorithms are developed for road geometry modeling; 1) a 3rd order Kalman filter (which models road geometry as a 2nd order polynomial equation), 2) a 4th order Kalman filter (which models road geometry as a 3rd order polynomial equation), and 3) a least square curve fit (which models the road geometry as a 2nd order polynomial equation). The least square curve fit uses only the information from the current image. On the other hand, the Kalman filter further utilizes the knowledge of vehicle kinematics and the information from previous images. Both of these methods find an optimal polynomial equation which minimizes the variance between the perceived and predicted lane marker locations. The results of the curve fitting to the modified lane marker array are coefficients of the

polynomial equation which accounts for the right lane edge; for a 2nd order polynomial equation, it is

$$y(x) = c_{r0} + c_{r1}x + c_{r2}x^2 \quad (IV.1),$$

for a 3rd order polynomial equation, it is

$$y(x) = c_{r0} + c_{r1}x + c_{r2}x^2 + c_{r3}x^3 \quad (IV.2)$$

where the y and x are the coordinates of a point on the curve with respect to the vehicle-fixed coordinate system as shown in Fig. IV.2. The vehicle-fixed coordinate system is defined according to the Society of Automotive Engineering (SAE) standard coordinate system [Gillespie, 1992]. Suppose the road has only one lane, the left road edge can be approximated by

$$y(x) = c_{l0} + c_{l1}x + c_{l2}x^2 = (c_{r0} - w_{lane}) + c_{r1}x + c_{r2}x^2 \quad (IV.3)$$

for 2nd order polynomial equation road model and

$$y(x) = c_{l0} + c_{l1}x + c_{l2}x^2 + c_{l3}x^3 = (c_{r0} - w_{lane}) + c_{r1}x + c_{r2}x^2 + c_{r3}x^3 \quad (IV.4)$$

for 3rd order polynomial equation road model with the constant lane width assumption, where the w_{lane} represents the lane width. For a multi-lane roadway, the w_{lane} is multiplied by the number of the lanes.

Perception Range Extension Algorithm

The perception of the sensing system for each lane edge is sometimes insufficient for accurate reconstruction of the geometry of the curve. This is particularly true when the vehicle is maneuvered through a curve as shown in Figure IV.2. This section presents an algorithm to extend the perception range for an edge such that a more accurate model for the down range geometry can be obtained.

In Figure IV.2, the perceived right lane edge is between c and d and the perceived left lane edge is between a and b . The right edge markers are represented by the stars and the left edge markers are represented by the squares. With the constant lane width assumption the left marker locations (i.e., the squares) can be mapped into the right edge

and the results are the triangles. Furthermore, the information from the near field sensor can also be utilized to supplement the right lane marker array. The result is a modified array which ranges from e to f. Thus, the perception range for the right edge is extended from cd to ef.

To map the left markers to the right edge, the tangential angles at the markers must be known. This is accomplished by least square curve fitting a polynomial equation to the perceived left lane marker locations from the sensors. If the result is

$$y = f(x) \quad (IV.5),$$

then the tangential angle at a marker is determined from

$$\vartheta = \frac{d}{dx} f(x) \quad (IV.6).$$

Least Square Curve Fitting Scheme For Lane Geometry Modeling

This section discusses the third step (i.e., road geometry modeling) of down range road geometry reconstruction as mentioned previously (see Fig. IV.1). The input to this module is the modified right lane marker array from the second step (i.e., the perception range extension module) and the results are two polynomial equations representing the right and left lane edges.

Suppose a modified right lane marker array (i.e., $(x_1, y_1) \dots (x_n, y_n)$) is available from the second step, then the acquisition of an optimal 2nd order polynomial equation using a least square curve fit can be obtained by first calculating

$$\mathbf{S} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} \quad (IV.7)$$

where

$$\mathbf{S} = \{s_0 \quad s_1 \quad s_2\}^T \quad (IV.8),$$

and

$$\mathbf{X} = \begin{bmatrix} f_0(x_1) & f_1(x_1) & f_2(x_1) \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ f_0(x_n) & f_1(x_n) & f_2(x_n) \end{bmatrix} \quad (IV.9)$$

$$\mathbf{Y} = [y_1 \quad \cdot \quad \cdot \quad y_n]$$

and then obtaining the optimal coefficients from

$$c_0 + c_1x + c_2x^2 = s_0f_0(x) + s_1f_1(x) + s_2f_2(x) \quad (\text{IV.10})$$

In equation (IV.10), the c 's are the coefficients of the polynomial equation as shown in equation (IV.1). Furthermore, the $f_i(x)$'s are orthogonal basis functions used to avoid numerical ill-conditioning of the lane marker array. The orthogonal functions can be obtained using the three-term recurrence relation algorithm discussed in [Conte, 1972]. For the particular arrangement of the lane marker locations, the three orthogonal functions are

$$\begin{aligned} f_0(x) &= 1 \\ f_1(x) &= x - 50 \\ f_2(x) &= x^2 - 100x + 1700 \end{aligned} \quad (\text{IV.11})$$

Using orthogonal functions is important for this purpose because the x coordinate of the lane markers usually ranges from a few meters to several hundred meters but the y coordinate usually ranges from zero to less than 50 meters, leading to numerical ill conditioning.

Kalman Filter For Lane Geometry Modeling

An alternative for down range road way geometry modeling is to use a Kalman filter. Again, a lane marker array is assumed will be available from the perception extension module. The results are, again, the estimated polynomial equations for the right and left lane edges.

To implement the Kalman filter, two models which relates the perceived geometry and the vehicle kinematics are developed; one model corresponds to modeling the road geometry with a 2nd order polynomial equation and the other model corresponds to modeling the road geometry with a 3rd order polynomial equation. The model corresponding to the 2nd order polynomial equation is developed as

$$\begin{Bmatrix} \dot{c}_0 \\ \dot{c}_1 \\ \dot{c}_2 \end{Bmatrix} = \begin{bmatrix} 0 & u & 0 \\ 0 & 0 & 2u \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} c_0 \\ c_1 \\ c_2 \end{Bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} v \\ r \end{Bmatrix} \quad (\text{IV.12})$$

where v and r are vehicle lateral velocity and yaw rate in a body-fixed coordinate system, c_o, c_1, c_2 are the coefficients of the polynomial equation for the lane edge (see Equation (IV.1)), and u is the vehicle forward velocity.

In equation (IV.12), the 2nd order polynomial equation is relative to a vehicle fixed coordinate system (see Fig. IV.3) defined as the standard SAE coordinate system [Gillespie, 1992]. Essentially, a 2nd order polynomial equation assumes a constant curvature;

$$\kappa(l) = 2c_2 \quad (IV.13)$$

where l is a parameter along the curve, $\kappa(l)$ denotes the curvature at l , and c_2 is a constant value. As shown in Fig. 3, the radius of the curve, ρ , is constant along the curve. With the assumption of small curvature, the tangential angle at l is obtained by integrating equation (IV.13);

$$\vartheta(l) = \int_0^l 2c_2 d\lambda = c_1 + 2c_2 l \quad (IV.14)$$

Furthermore, the y coordinate of the curve at a location l is obtained from;

$$y(l) = \int_0^l \vartheta(\lambda) d\lambda = \int_0^l (c_1 + 2c_2 \lambda) d\lambda = c_o + c_1 l + c_2 l^2 \quad (IV.15)$$

In equations (IV.13) ~ (IV.15), l can be replaced by x , the x coordinate of the curve at l , under the assumption of small curvature, which changes equation (IV.15) to

$$y(x) = c_o + c_1 x + c_2 x^2 \quad (IV.16)$$

For a more detailed discussion regarding the derivation of equations (IV.13) ~ (IV.16) the reader is referred to [Dickmanns, 1986]. From equation (IV.16), one can observe that

$$c_o = y_{x=0} \quad (IV.17)$$

Thus,

$$\dot{c}_o = \dot{y}_{x=0} = (c_1 \dot{x} + 2c_2 x \dot{x})_{x=0} - v = c_1 u - v \quad (IV.18)$$

Furthermore, equation (IV.14) shows that

$$c_1 = \vartheta_{x=0} \quad (IV.19)$$

Therefore,

$$\dot{c}_1 = \dot{\vartheta}_{x=0} - r = 2c_2\dot{x} - r = 2c_2u - r \quad (IV.20)$$

Finally, because of the constant curvature assumption,

$$\dot{c}_2 = 0 \quad (IV.21)$$

Then, equation (IV.12) is obtained with the combination of equations (IV.18), (IV.20), and (IV.21).

On the other hand, the model corresponding to a 3rd order polynomial equation is developed as

$$\begin{Bmatrix} \dot{c}_0 \\ \dot{c}_1 \\ \dot{c}_2 \\ \dot{c}_3 \end{Bmatrix} = \begin{bmatrix} 0 & u & 0 & 0 \\ 0 & 0 & 2u & 0 \\ 0 & 0 & 0 & 6u \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{Bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{Bmatrix} v \\ r \end{Bmatrix} \quad (IV.22)$$

where v and r are vehicle lateral velocity and yaw rate in a body-fixed coordinate system, c_0, c_1, c_2, c_3 are the coefficients of the polynomial equation for the lane edge (see Equation. (IV.2)), and u is the vehicle forward velocity.

Essentially, a 3rd order polynomial equation assumes a linear variation for the curvature along the forward distance;

$$k(l) = 2c_2 + 6c_3l \quad (IV.23)$$

Again, l is a parameter along the curve, $\kappa(l)$ denotes the curvature at l , and c_2 and c_3 are constant values. With the assumption of small curvature along the curve, the tangential angle at l is obtained by integrating equation (IV.23) and yield

$$\vartheta(l) = \int_0^l (2c_2 + 6c_3\lambda) d\lambda = c_1 + 2c_2l + 3c_3l^2 \quad (IV.24)$$

Finally, the y coordinate of the curve at a location l is obtained from;

$$y(l) = \int_0^l \vartheta(\lambda) d\lambda = \int_0^l (c_1 + 2c_2\lambda + 3c_3\lambda^2) d\lambda = c_0 + c_1l + c_2l^2 + c_3l^3 \quad (IV.25)$$

under the assumption of small tangential angle along the curve with respect to the reference frame. With the same small tangential angle assumption, the parameter l can be replaced by x , the x coordinate of the curve at l , and yield

$$y(x) = c_0 + c_1x + c_2x^2 + c_3x^3 \quad (IV.26)$$

Using a similar derivation as that for the 3rd order Kalman filter, the following equations can be obtained;

$$\dot{c}_0 = c_1 u - v \quad (IV.27)$$

$$\dot{c}_1 = 2c_2 u - r \quad (IV.28)$$

$$\dot{c}_2 = 6u \quad (IV.29)$$

$$\dot{c}_3 = 0 \quad (IV.30)$$

Equation (IV.22) can then be obtained by combining equation (IV.27)~(IV.30). For Kalman filtering implementation in discrete domain, equations (IV.12) and (IV.22) are then discretized. Denote the results as

$$\mathbf{C}(k+1) = \Phi \mathbf{C}(k) + \Gamma \mathbf{u}(k) \quad (IV.31)$$

The implementation of the Kalman filter to obtain an optimal polynomial equation, which minimizes the variation between the perceived lane marker locations and the predicted locations, can be achieved using

$$\hat{\mathbf{C}}(k+1) = \Phi \hat{\mathbf{C}}(k) + \Gamma \mathbf{u}(k) + \Phi \mathbf{L}(\mathbf{Y}(k) - \mathbf{H} \hat{\mathbf{C}}(k)) \quad (IV.32)$$

where

$$\mathbf{Y}(k) = \{y_{x=10}(k) \ y_{x=30}(k) \ y_{x=50}(k) \ y_{x=70}(k) \ y_{x=90}(k)\}^T \quad (IV.33)$$

\mathbf{H} is the output matrix; for the 3rd order Kalman filter

$$\mathbf{H} = \begin{bmatrix} 1 & 10 & 10^2 \\ 1 & 30 & 30^2 \\ 1 & 50 & 50^2 \\ 1 & 70 & 70^2 \\ 1 & 90 & 90^2 \end{bmatrix} \quad (IV.34)$$

for the 4th order Kalman filter

$$\mathbf{H} = \begin{bmatrix} 1 & 10 & 10^2 & 10^3 \\ 1 & 30 & 30^2 & 30^3 \\ 1 & 50 & 50^2 & 50^3 \\ 1 & 70 & 70^2 & 70^3 \\ 1 & 90 & 90^2 & 90^3 \end{bmatrix} \quad (IV.35)$$

\mathbf{L} is the Kalman filter gain. The initial estimate can be obtained using the least square curve fit to the initial lane marker locations; this improves the convergence rate of the estimator.

The Kalman filter gain is obtained from a process noise covariance matrix and a measurement error covariance matrix. To obtain these two matrices, a more sophisticated model is utilized to simulate the error associated with the sensors and the accuracy of the model in equation (IV.12) and (IV.22) [Venhovens, 1995]. Essentially, the process noises come from the residuals of a polynomial equation (here, it is either 2nd order or 3rd order) to approximate the real road geometry (e.g., a straight section connected by a pure curve). The measurement noise is associated with the perception errors due to vehicle vibration and roadway superelevation. This covariance matrix is obtained from the lane marker locations perception error by the sensors in a simulation. The measurement in the near field (i.e., y_{x-10}) is important in order to obtain accurate lateral deviation estimate of the vehicle from the lane edge. Whereas, the measurements in the far field are significant for capturing the upcoming geometry variation while the vehicle is in a transition period (i.e., on a straight section approaching a curve or vice versa).

Characterization Of The Road Geometry Perception Uncertainty

The perceived road geometry deviates from the true geometry mainly due to the influence of vehicle vibration and superelevation. In the proposed active safety system, an algorithm is developed to estimate vehicle pitch and roll to compensate the perception error due to vehicle vibration [Venhovens, 1995]. However, it can only compensate part of the constant level error; there are still bias and stochastic components in the perception error. In this study, the bias is assumed to be negligible. The proposed uncertainty characterization algorithm only characterizes the perception error associated with the stochastic component. The results are the error covariance's of the perceived polynomial coefficients (i.e., the covariance's of the possible deviation of the perceived coefficients from the true geometry). Furthermore, the proposed algorithm is to characterize the steady state perception uncertainty associated with Kalman filtering process. This algorithm is validated in the following section.

Essentially, the estimate update of a Kalman filter can be divided into two steps; measurement update and time update [Brown, 1983]. The measurement update is made when a new set of data is acquired;

$$\hat{\mathbf{C}}^-(k) = \hat{\mathbf{C}}(k) + \mathbf{L}(\mathbf{Y}(k) - \mathbf{H}\hat{\mathbf{C}}(k)) \quad (\text{IV.36})$$

where $\hat{\mathbf{C}}(k)$ represents the best estimate using previously available information. Suppose

$$\mathbf{e}(k) = \mathbf{C}(k) - \hat{\mathbf{C}}(k) \quad (\text{IV.37})$$

in which $\mathbf{C}(k)$ represents the true values, then the error covariance matrix associated with the best estimate can be written as

$$\mathbf{P}(k) = E[\mathbf{e}(k)\mathbf{e}^T(k)] \quad (\text{IV.38})$$

and the update of the error covariance associated with equation (IV.36) is

$$\mathbf{P}^-(k) = \mathbf{P}(k) - \mathbf{P}(k)\mathbf{H}^T(\mathbf{H}\mathbf{P}(k)\mathbf{H}^T + \mathbf{R}_v)^{-1}\mathbf{H}\mathbf{P}(k) \quad (\text{IV.39})$$

where \mathbf{R}_v is the covariance matrix for the measurement noise. The time update is associated with the motion equations (i.e., equation (IV.31));

$$\hat{\mathbf{C}}(k+1) = \Phi\hat{\mathbf{C}}^-(k) + \Gamma\mathbf{u}(k) \quad (\text{IV.40})$$

The update of the error covariance matrix associated with equation (IV.40) is

$$\mathbf{P}(k+1) = \Phi\mathbf{P}^-(k)\Phi + \mathbf{R}_w \quad (\text{IV.41})$$

where \mathbf{R}_w is the process noise covariance matrix. Equations (IV.39) and (IV.41) are then combined to yield

$$\mathbf{P}(k+1) = \Phi(\mathbf{P}(k) - \mathbf{P}(k)\mathbf{H}^T(\mathbf{H}\mathbf{P}(k)\mathbf{H}^T + \mathbf{R}_v)^{-1}\mathbf{H}\mathbf{P}(k))\Phi + \mathbf{R}_w \quad (\text{IV.42})$$

which is a first order difference equation. To find the covariance matrix at every step, only the initial condition is required. The steady state value of equation (IV.42) can be obtained by solving the eigenvectors of

$$\mathbf{S} = \begin{bmatrix} \Phi^T + \mathbf{H}^T\mathbf{R}_v^{-1}\mathbf{H}\Phi^{-1}\Gamma\mathbf{R}_w\Gamma^T & -\mathbf{H}^T\mathbf{R}_v^{-1}\mathbf{H}\Phi^{-1} \\ -\Phi^{-1}\Gamma\mathbf{R}_w\Gamma^T & -\Phi^{-1} \end{bmatrix} \quad (\text{IV.43})$$

The steady state solution of equation (IV.42) is

$$\mathbf{P}_\infty = \Lambda_s \mathbf{X}_s^{-1} \quad (\text{IV.44})$$

where

$$\begin{bmatrix} \mathbf{X}_I \\ \Lambda_I \end{bmatrix} \quad (IV.45)$$

are the eigenvectors of \mathbf{S} associated with its stable eigenvalues.

Since the initial estimate of the state vector, $\hat{\mathbf{C}}(0)$, is obtained by least square curve fitting a polynomial equation to the acquired lane marker locations, the initial covariance matrix, $\mathbf{P}(0)$, is the error covariance matrix of the least square curve fit. This matrix is obtained by conducting a simulation with typical highway geometry and road surface conditions and then calculating the error covariance of the least square curve fit results with respect to the true geometry. In other words, suppose $\hat{\mathbf{C}}_{LSQ}$ is the estimate of the polynomial coefficients using least square curve fit, the initial covariance matrix for the Kalman filter is obtained from

$$\mathbf{P}(0) = E[(\mathbf{C} - \hat{\mathbf{C}}_{LSQ})(\mathbf{C} - \hat{\mathbf{C}}_{LSQ})^T] \quad (IV.46)$$

where the perception error vector, $\mathbf{C} - \hat{\mathbf{C}}_{LSQ}$, is assumed to be stationary and zero mean.

Results And Discussion

This section first discusses the simulation results obtained by implementing the three different road geometry modeling algorithms introduced in previous sections. For simulations, the road geometry (a typical highway geometry) and vehicle velocity are the same in all the cases. The vehicle is driven at 90 Km/h from a straight section to a curve with curvature typical of highway driving. Three different cases are conducted. The differences between the first two cases are the road unevenness; one has a typical highway road unevenness and the other one has relatively rough surface. The last case has a slight different geometry and relatively significant superelevation from the previous cases. The Kalman filter gain is designed based on the road geometry and vehicle velocity. Finally, road geometry perception uncertainty in typical highway driving is presented.

Figure IV.4 shows the results from the Kalman filter and least square curve fit for a relatively good (i.e., smooth asphalt and small superelevation) road condition. Such a road unevenness profile accounts for 80% of the typical US highway road unevenness. The

road geometry has a straight section connected by a curve with a transition in between.

Fig. IV.5 shows the deviation of the perceived curve from the true curve, which is calculated with the following equation;

$$\delta_r = \frac{1}{100} \sum_{x=1}^{100} |y_n(x) - y_p(x)| \quad (IV.47)$$

where

$$y_n(x) = f_r(x) \quad (IV.48)$$

is the location of the true road geometry evaluated at x and

$$y_p(x) = f_p(x) \quad (IV.49)$$

is the location of the perceived road geometry evaluated at x . The results indicate that the Kalman filter has better performance than the least square curve fit under steady state conditions. However, it is important to note that a lag in the geometry estimation is evident in the results for the Kalman filter. This estimation lag causes the Kalman filter to have worse performance than the least square curve fit in the transition period. This is because of the filtering effects of a Kalman filter; for the 3rd order Kalman filter, the constant curvature assumption (i.e., equation (IV.21)) for the previewed road geometry is the main reason and for the 4th order Kalman filter, the constant curvature changing rate (i.e., equation (IV.30)) is the main reason. However, due to the allowance of curvature variation along the curve, the 4th order Kalman filter has a better performance than the 3rd order Kalman filter in the transition period. Fig. IV.5 shows that the 4th order Kalman filter has a faster adaptation than the 3rd order Kalman filter. An overshoot of the 4th order Kalman filter larger than the magnitude of the 3rd order Kalman filter is evident. Such a lag in geometry estimation is not serious for the conditions in the current simulation; however, it introduces a substantial error when the down range road geometry exhibits rapid change. For example, if the down range road geometry has a straight section connected by a curve with small radius (e.g., a ramp), a substantial “lane tracking margin” assessment error

occurs in the transition period. Under such conditions, the results from a least square curve fit are also unsatisfactorily.

In the next case, a vehicle is again driven from a straight section to a curve. However, the road surface is relatively rougher than in the previous case to introduce significant vehicle pitch and roll. When the vehicle pitch/roll, an on-board sensor loses track of its orientation and consequently provides inaccurate lane marker locations. Fig. IV.6 shows the road geometry reconstruction between the three different algorithms and Fig. IV.7 shows the deviation of the perceived geometry from the true geometry. The same conclusions from the previous case can also be drawn from this case; 1) a Kalman filter has a better performance than a least square curve fit in steady state condition, 2) a least square curve fit is better in transition period than a Kalman filter, 3) a 4th order Kalman filter has a better result than the 3rd order Kalman filter in transition period. Furthermore, the simulations show that road geometry perception error due to significant vehicle vibration effect (usually of a high frequency (e.g., > 1 Hz)) can be successfully attenuated by the Kalman filter.

The last case has a road geometry similar to the previous cases; however, the superelevation gradually increases (from the straight section to the curve) up to a value of 6 degrees. Figure IV.8 shows the results of this simulation. Obviously, the Kalman filter has a performance superior to that of the least square curve fit in this simulation. The results in this simulation is mainly because of the fact that the Kalman filter has a recursive weighted least square curve fit. In other words, an appropriately recursive weighted least square curve fit would be expected to have similar performance as the Kalman filter in the similar simulation conditions. In the current case, the 3rd order Kalman filter imposes significant gains for errors associated with the near field data (i.e. y_{x-10} and y_{x-30}), which implies that for a steady state situation, only the near field data may be adequate for recovering the down range curve. The 4th order Kalman filter gain has a relatively larger weight on the far field data such that it has a better performance in the transition period.

The mechanism of a Kalman filter to cope with superelevation can be explained in more detail with Fig. IV.10. Essentially, superelevation causes a previewed curve to appear as a straight line to the sensors, as is shown by the result from the least square curve fit in Fig. IV.8. However, because of the knowledge of vehicle kinematics, and the recursive averaging of the images, the road curvature is detected by the Kalman filter. As the vehicle proceeds through the curve, the curve appears to the sensor as a straight line at every instant (i.e., the perceived road geometry is a straight line with respect to the vehicle-fixed coordinate system). However, since the vehicle kinematics is known, these straight lines have different orientations with respect to a global reference frame. Thus, it appears to the Kalman filter that a curvature exists in the previewed curve. Fig. IV.9 also shows that a 4th order Kalman filter can have a better performance than a 3rd order Kalman filter in steady state. This is simply because that a 3rd polynomial equation (associated with a 4th order Kalman filter) can have a better fit to the acquired data than a 2nd order polynomial equation (associated with a 3rd order Kalman filter).

Finally, the road geometry perception uncertainty characterization algorithm is validated and a typical result is simulated for both of the 3rd order and 4th order Kalman filters. For this purpose, two simulations using the 3rd order Kalman filter and the 4th order Kalman filter for road geometry reconstruction are conducted and the results are shown in Fig. IV.11 and Fig. IV.12 respectively. The simulations have road surface characterizing smooth asphalt unevenness which accounts for about 80% the typical US highway [Sayer, 1986]. As described previously, the assumption for the uncertainty characterization is that the perception error is stationary and zero mean. This says that if there is a bias in the road geometry estimation as shown in Fig. IV.11 and Fig. IV.12, it can be ignored. This assumption is validated with simulations which investigate how much the bias in the polynomial coefficients estimation affect the “lane tracking margin” assessment. For the proposed active safety system, Time to Lane Crossing (TLC) is used for the “lane tracking margin” assessment, which describes the expected time for the

vehicle to cross the lane boundary. The reader is referred to [Lin and Ulsoy, 1995] for a detailed description of the TLC estimation algorithm. Table IV.1 and Table IV.2 show the effect of the geometry perception bias shown in Fig. IV.11 and Fig. IV.12 to the TLC estimation at different TLC level respectively. The results show that it is reasonable to ignore the bias in geometry perception.

Table IV.1 : Validation Of Zero Mean Assumption For Geometry Perception Of A 3rd Order Kalman Filter

<i>TLC (sec)</i>	2.44	3.10	3.50	3.82
Bias In TLC Estimation With The Bias In Geometry Perception Of A 3rd Order Kalman Filter (sec)	0	-0.04	-0.08	-0.14

Table IV.2 : Validation Of Zero Mean Assumption For Geometry Perception Of A 4th Order Kalman Filter

<i>TLC (sec)</i>	2.19	3.27	3.75	4.14
Bias In TLC Estimation With The Bias In Geometry Perception Of A 4th Order Kalman Filter (sec)	0.02	-0.01	-0.08	-0.19

Subsequently, the ability of the proposed algorithm to predict the geometry perception uncertainty is validated. Simulations for the covariance matrix of the coefficient perception errors are conducted for the 3rd order and 4th order Kalman filters and the results are shown in Fig. IV.13 and Fig. IV.14, which corresponds to the geometry shown in Fig. IV.11 and Fig. IV.12 respectively. These results are then compared with the standard deviation of the steady state error in Fig. IV.11 and Fig. IV.12. Table IV.3 compares the results from the 3rd order Kalman filter and Table IV.4 compares the results from the 4th order Kalman filter. Then the effect of the differences between the predicted values and the true values to the TLC uncertainty prediction is studied. Table IV.5 compares the difference in TLC uncertainty characterization with the true standard deviation and the predicted standard deviation of the geometry perception error associated with the

3rd order Kalman filter. Table IV.6 shows the results associated with the 4th order Kalman filter. The results show that the proposed algorithm is satisfactory for predicting the geometry perception uncertainty. However, this algorithm can only characterize the geometry perception uncertainty in steady state condition. For the perception uncertainty characterization in transition period, a more sophisticated algorithm which can reset the covariance matrix when encounters a transition period is necessary. Furthermore, the results in this section only validate the proposed uncertainty characterization algorithm in a smooth road surface condition (which accounts for about 80% the typical US highway). Further study must be conducted for significant superelevation condition (which causes significant bias in the measurement and thus may introduce substantial bias in the Kalman filter estimation).

Table IV.3 : Comparison Of The True And Predicted Values Of The Coefficient Perception Errors Standard Deviation Of A 3rd Order Kalman Filter

	σ_{c_0}	σ_{c_1}	σ_{c_2}
True Standard Deviation Of A 3rd Order Kalman Filter	0.0052	4.9372e-04	1.9373e-05
Predicted Standard Deviation For The 3rd Order Kalman Filter	0.0072	4.9229e-04	2.6882e-05

Table IV.4 : Comparison Of The True And Predicted Values Of The Coefficient Perception Errors Standard Deviation Of A 4th Order Kalman Filter

	σ_{c_0}	σ_{c_1}	σ_{c_2}	σ_{c_3}
True Standard Deviation Of A 4th Order Kalman Filter	0.0029	3.1524e-04	1.3678e-05	1.3621e-07
Predicted Standard Deviation For The 4th Order Kalman Filter	0.0069	6.9875e-04	1.7612e-05	1.9368e-07

Table IV.5 : Comparisons Of The TLC Uncertainty Prediction Between The Results With The True Geometry Perception Uncertainty And The Predicted Geometry Perception Uncertainty Associated With A 3rd Order Kalman Filter

<i>TLC(sec)</i>	2.6000	3.1000	3.5000	3.8200
Predicted σ_{TLC} with True Geometry Perception Uncertainty (sec)	0.0674	0.1182	0.1782	0.2361
Predicted σ_{TLC} with Predicted Geometry Perception Uncertainty (sec)	0.0905	0.1601	0.2425	0.3223

Table IV.6 : Comparisons Of The TLC Uncertainty Prediction Between The Results With The True Geometry Perception Uncertainty And The Predicted Geometry Perception Uncertainty Associated With A 4th Order Kalman Filter

<i>TLC(sec)</i>	2.1900	3.2700	3.7500	4.1400
Predicted σ_{TLC} with True Geometry Perception Uncertainty (sec)	0.0361	0.1479	0.2557	0.3920
Predicted σ_{TLC} with Predicted Geometry Perception Uncertainty (sec)	0.0344	0.1514	0.2685	0.4190

Summary And Conclusions

This chapter presents a process for down range road geometry reconstruction with a sensing system which contains a far-field sensor and a near-field sensor. The road geometry reconstruction includes three steps : (1) lane marker location acquisition , (2) perception range extension, and (3) road geometry modeling. This context assumes the first step (i.e., the lane marker location acquisition) is available and proposes a perception range extension algorithm to solve the field of view problem. Furthermore, a 3rd order Kalman filter, a 4th order Kalman filter, and a least square curve fit are developed for road geometry modeling. Simulations with the Kalman filter and the least square curve fit are conducted under several different conditions. The results are compared. Finally, an algorithm to characterize the perception uncertainty for the down range road geometry is presented, its validity is discussed and perception uncertainty is simulated for a typical highway geometry and road surface.

The simulation results show that for a typical highway, the Kalman filters provide better road geometry estimation performances than the least square curve fit under steady state conditions. However, due to the lag introduced by the filtering effect of the Kalman filters, the least square curve fit is expected to perform better during transition periods. The performance of a 4th Kalman filter is better than a 3rd order Kalman filter in transition period because it relaxes the assumption of constant curvature for down range road geometry. The simulation results also show that a Kalman filter which can successfully cope with superelevation and vehicle vibration can be developed. The mechanism of a Kalman filter to cope with superelevation is its recursive averaging of the perceived road geometry which is essentially a straight line at every instance. Generally, a Kalman filter has better performance than a least square curve fitting scheme, and a much better performance than the least square curve fit in steady state. A 4th order Kalman filter can have a better performance than a 3rd order Kalman filter because it assumes a linear variation for the down range curvature, which improves the results in transition period, and it can be a better fit to the acquired data, which improves the results in steady state. Although the least square curve fit performs better during transitions, its advantages are limited. The main reason for the superiority of the Kalman filter is because it uses the information from both the image and vehicle kinematics and recursively averages the perceived road geometry.

Finally, the uncertainty characterization scheme is shown valid for predicting the standard deviation of the possible error of the road geometry perception for about 80% of the typical highway. The maximum error in predicting the TLC uncertainty with the proposed algorithm for a typical highway driving is about 0.09 sec for a 3rd order Kalman filter and is about 0.02 sec for a 4th order Kalman filter.

In future research, the issue of the performance of a Kalman filter using only the near field lane marker information can be investigated. It is expected that it would yield similar steady state results as the current design. However, its ability to recover road geometry in a transition period will be limited. Furthermore, a more sophisticated

algorithm to characterize the geometry perception uncertainty in transition is also interesting.

References

- Broida T.J., Chellappa R., "Estimation Of Object Motion Parameters From Noisy Image", *IEEE Transaction On Pattern Analysis And Machine Intelligence*, Vol. PAMI-8, No.1, pp 90-99, 1986.
- Brown R.G., *Introduction To Random Signal Analysis And Kalman Filtering*, John Wiley & Sons, Inc., 1983.
- Conte S.D., de Boor C., "Elementary Numerical Analysis", McGraw-Hill Book Company, 1972.
- Dickmanns E.D., Zapp A., "A curvature-based scheme for improving road vehicle guidance by computer vision", *Proc. of SPIE on Mobile Robots*, Vol. 727, pp. 161-168, 1986.
- Dickmanns E.D., and Graefe V., "Applications of Dynamic Monocular Machine Vision", in *Machine Vision and Applications*, New York, NY, Springer-Verlag, pp. 241-261, 1988.
- Gillespie T.D., *Fundamental of Vehicle Dynamics*, Society of Automotive Engineers, Inc., 1992.
- Graefe V., Kuhnert K.-D., "Vision-based Autonomous Road Vehicles", in *Vision-based vehicle Guidance*, New York, NY, Springer-Verlag, 1992.
- Kenue S.K., "LANELOK: Detection of Lane Boundaries and Vehicle Tracking Using Image-Processing Techniques-Part I: Hough Transform, Region-Tracking and Correlation Algorithm ", *Proc. SPIE Conference on Mobile Robots IV*, Philadelphia, PA., Nov. 1989.
- Kriegman D.J., Triendl E., and Binford T.O., "Stereo Vision and Navigation in Buildings for Mobile Robots", *IEEE Trans. on Robotics and Automation*, Vol. 5, No. 6, Dec. 1989.
- LeBlanc D.J., Venhovens P.J.Th., Pilutti T.E., Lin C. F., Ervin R.D., Ulsoy A.G., MacAdam C.C., "A warning and intervention system for preventing inadvertent road departure", *14th IAVSD Symposium on the Dynamics of Vehicles on Roads and Tracks*, Ann Arbor, MI, Aug. 1995.
- Lin Chiu-F. and Ulsoy A. Galip, "Calculation Of The Time To Lane Crossing And Analysis Of Its Frequency Distribution", *Proc. of American Control Conference*, Seattle, WA. 1995a.
- Rives P., Breuil E., Espian B., "Recursive Estimation Of 3D Features Using Optical Flow And Camera Motion", Hertzberger LO (ed) *Proc. of Intelligent Autonomous Systems*, Amsterdam, pp 522-532, 1986.
- Sayer M., "Characteristics Of The Power Spectral Density Function For Road Roughness", *Symposium Of Simulation And Control Of Ground Vehicle*, ASME, 1986.

Thorpe C., Hebert M., Kanade T., Shafer S., "The New Generation System for the CMU Navlab", in *Vision-based Vehicle Guidance*, New York, NY, Springer-Verlag, 1992.

Tsugawa S., Hirose T., and Yatabe T., "An Intelligent Vehicle With Obstacle Detection And Navigation Functions", *Proc. IECON*, Tokyo, pp 303-308, 1984.

Venhovens Paul J. Th., *Manual For An Road-Departure Prevention System Simulation Program*, The University of Michigan Transportation Research Institute, 1995.

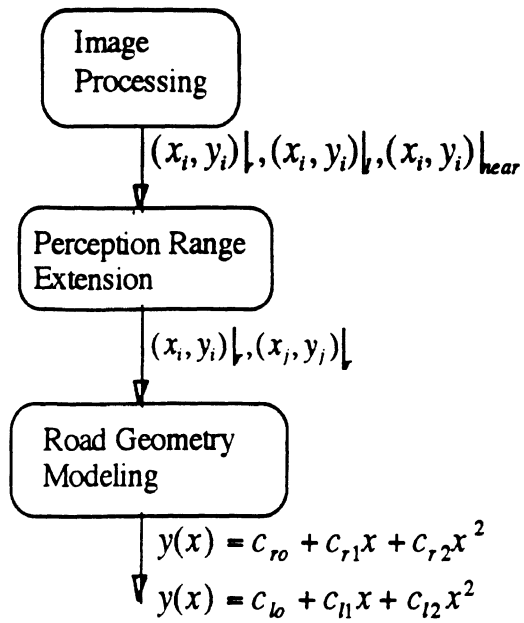


Figure IV.1 : Lane Geometry Reconstruction Procedures

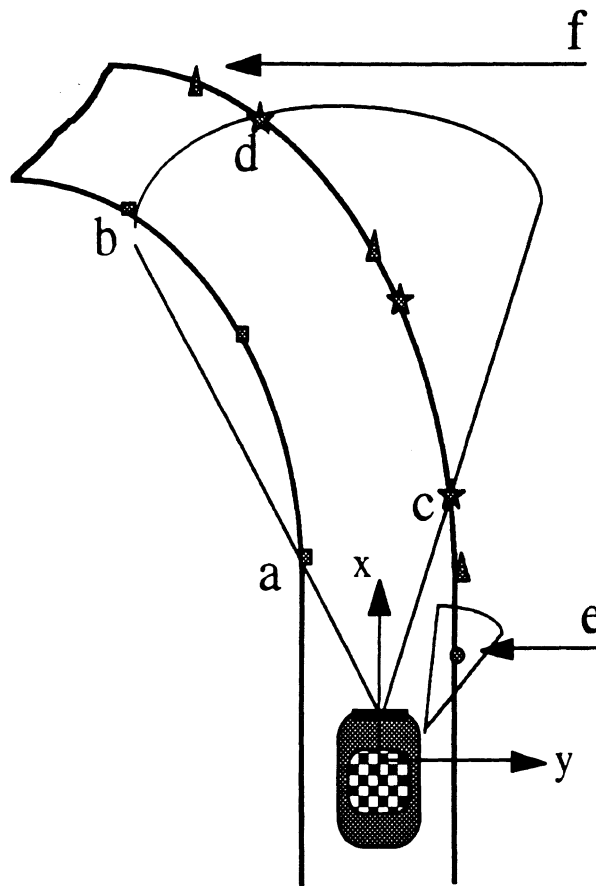


Figure IV.2 : Field Of View Constraint Of A Sensor

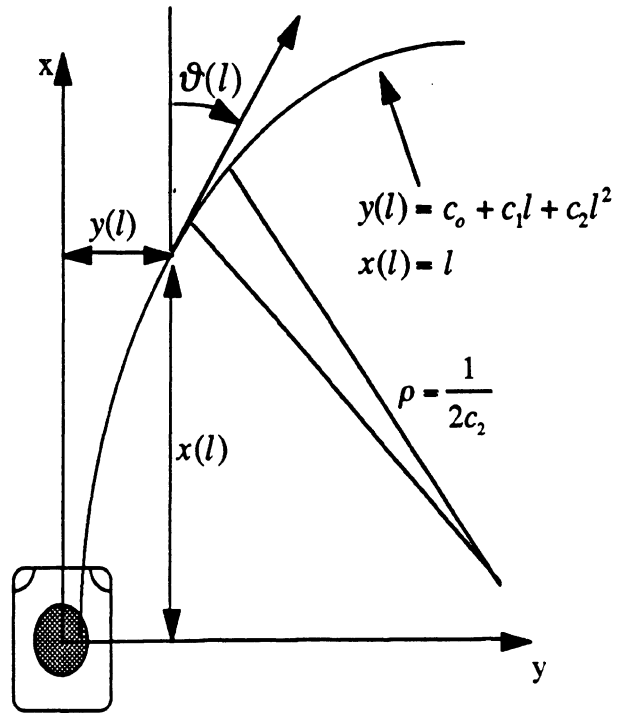


Figure IV.3 : Geometrical Relation For A 2nd Order Polynomial Equation

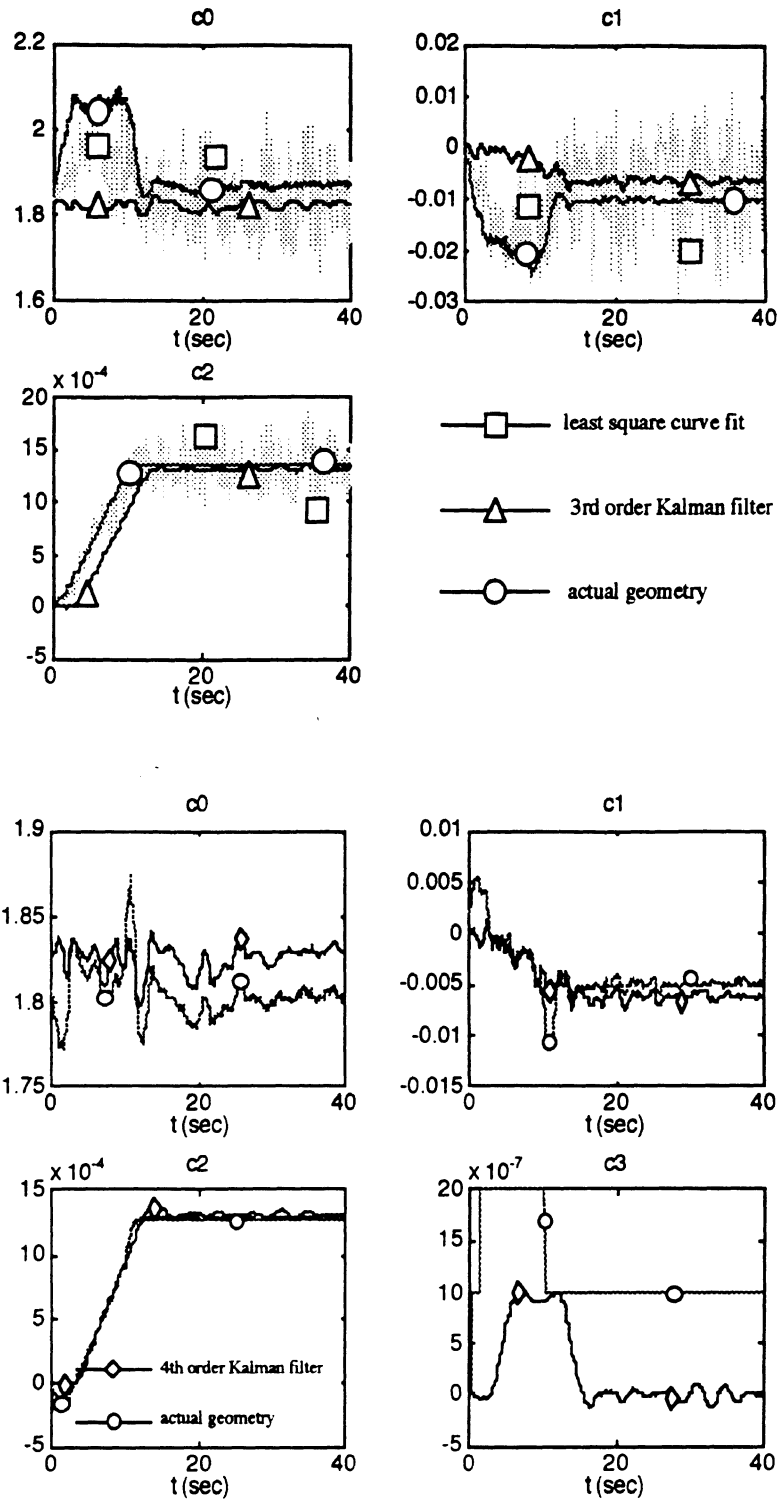


Figure IV.4 : Road Geometry Reconstruction Under A Good Road Condition

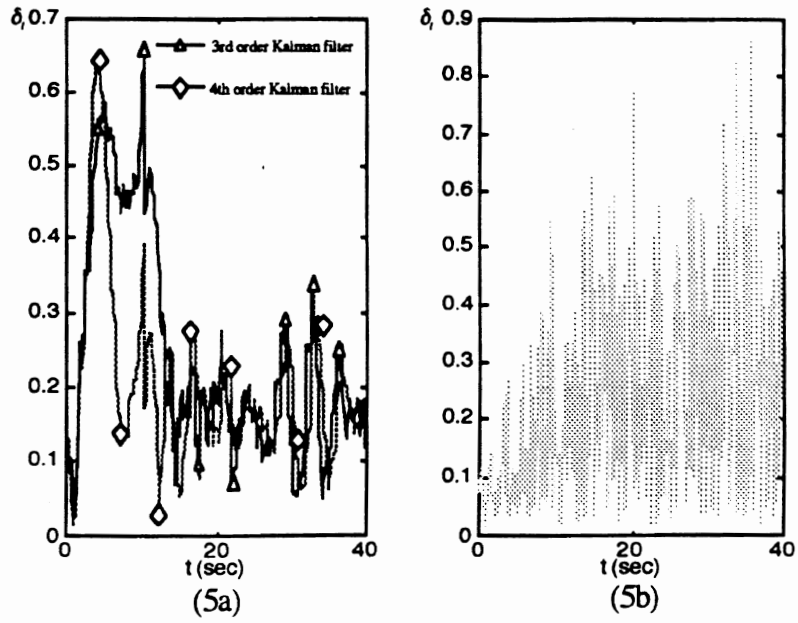


Figure IV.5 : Deviation Of The Perceived Road Geometry From the True Road Geometry Associated With The Road Geometry Reconstruction In Fig. IV.4; (a) Kalman filter, (b) least square curve fit

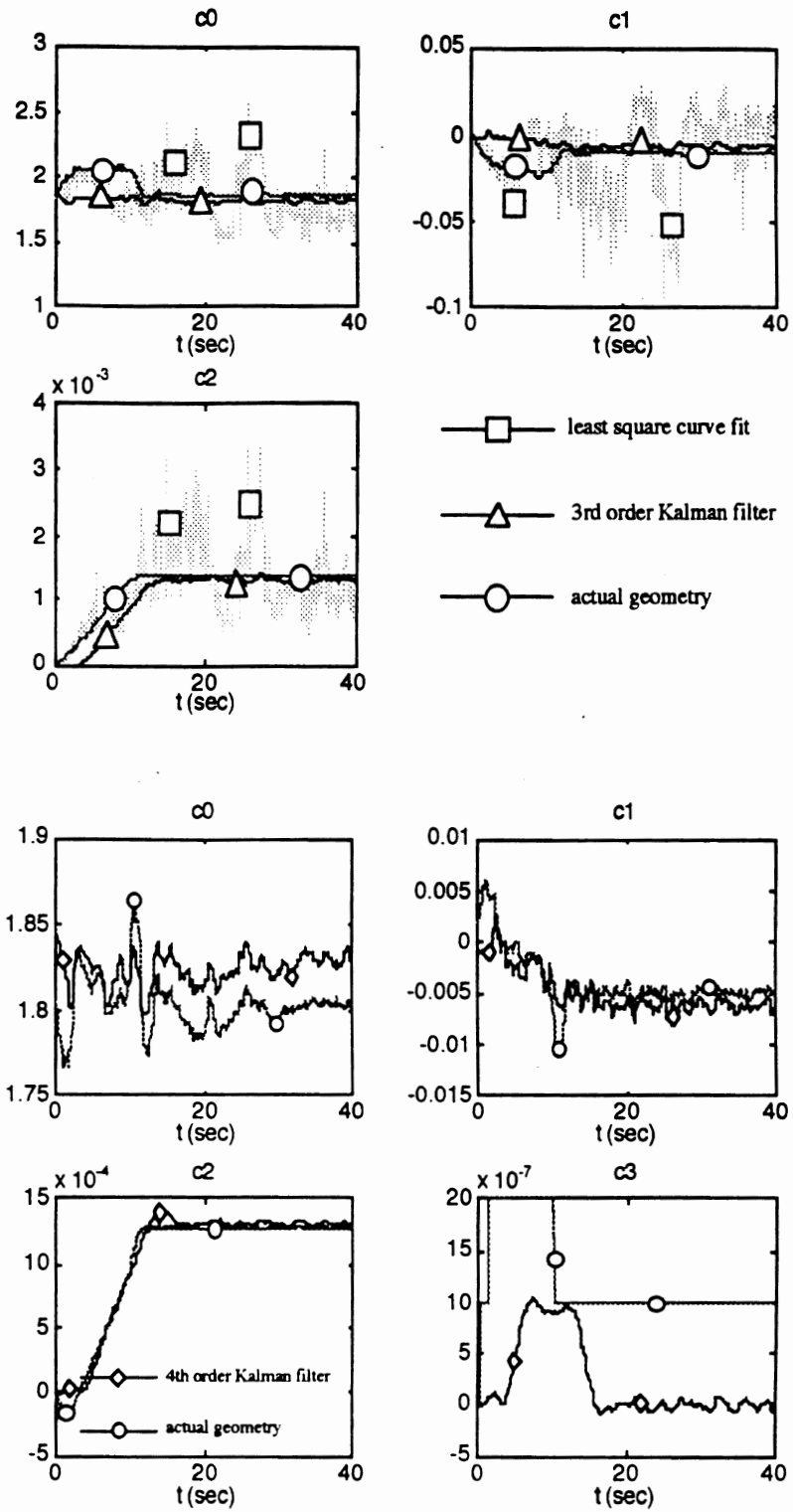


Figure IV.6 : Road Geometry Reconstruction Under A Rough Road Surface Condition

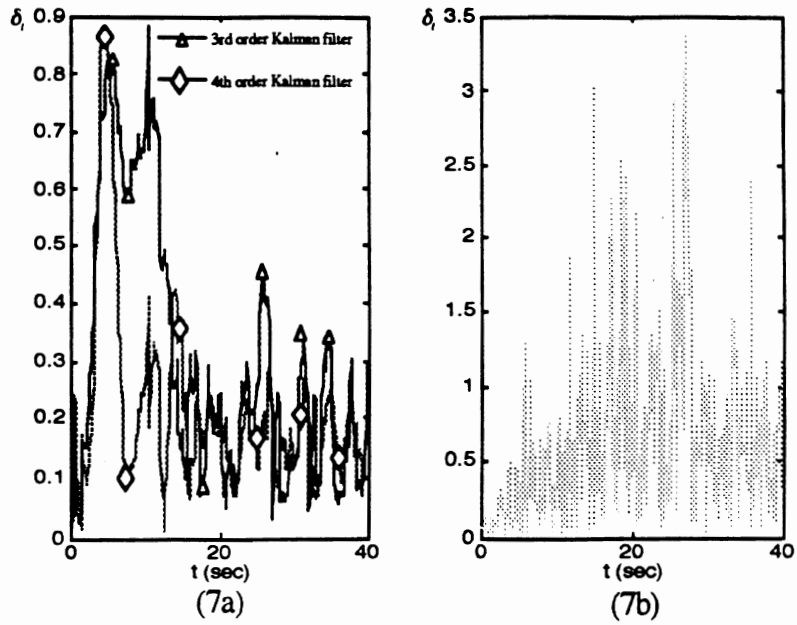


Figure IV.7 : Deviation Of The Perceived Road Geometry From the True Road Geometry Associated With The Road Geometry Reconstruction In Fig. IV.6; (a) Kalman filter, (b) least square curve fit

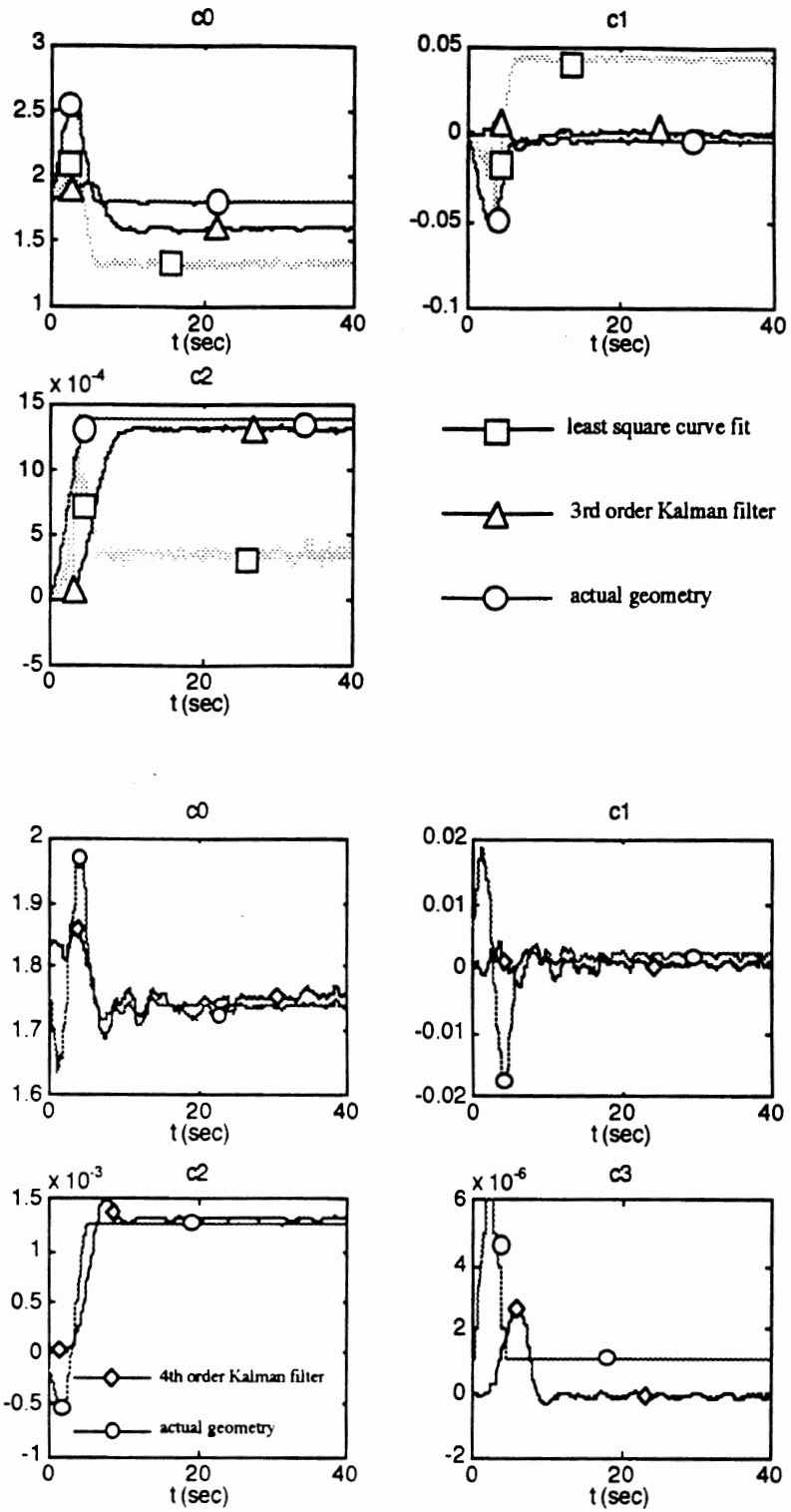


Figure IV.8 : Road Geometry Reconstruction Under A Superelevated Road Condition

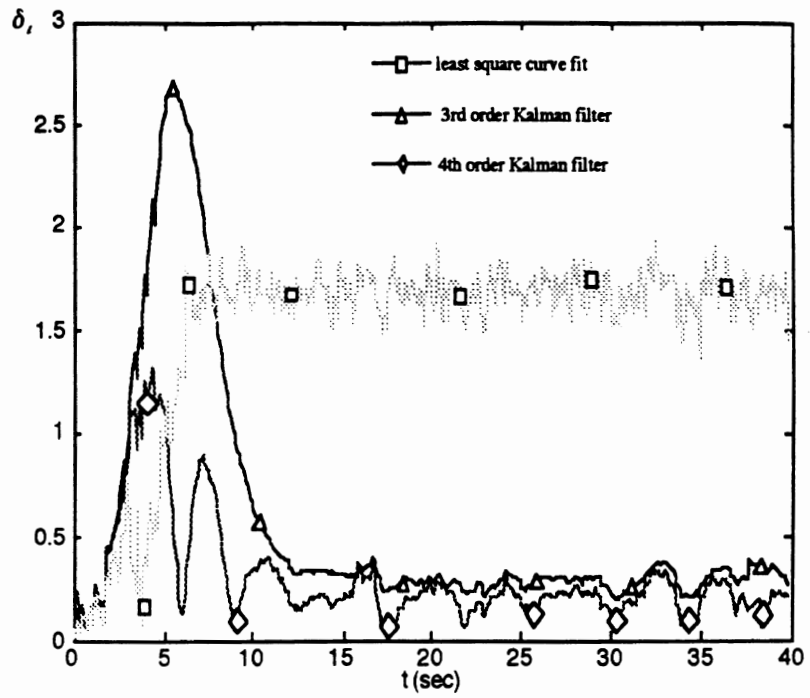


Figure IV.9 : Deviation Of The Perceived Road Geometry From the True Road Geometry Associated With The Road Geometry Reconstruction In Fig. IV.8

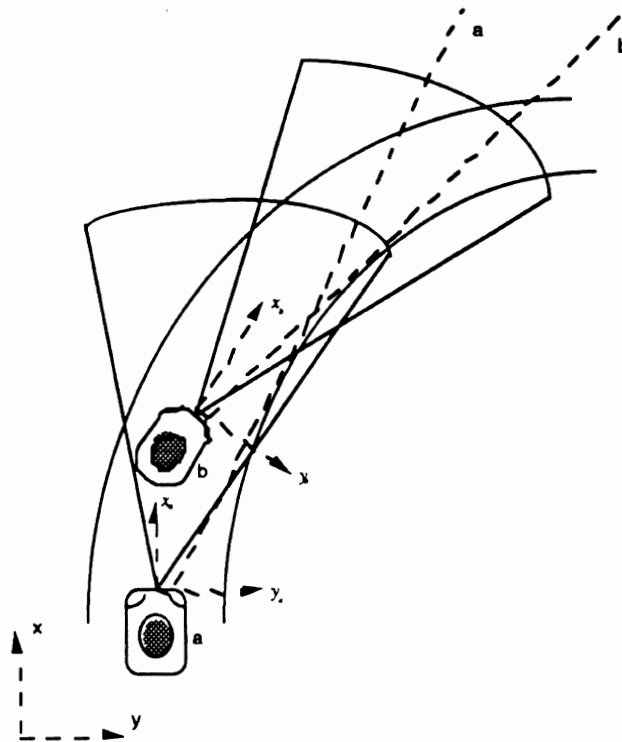


Figure IV.10 : Road Geometry Perception In A Superelevated Roadway

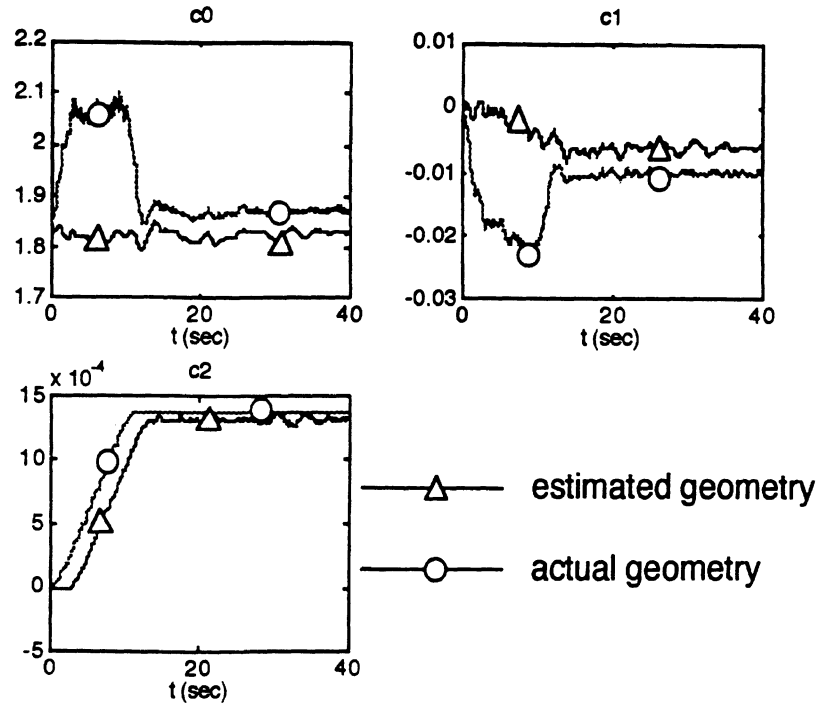


Figure IV.11 : Road Geometry Reconstruction With A 3rd Order Kalman Filter

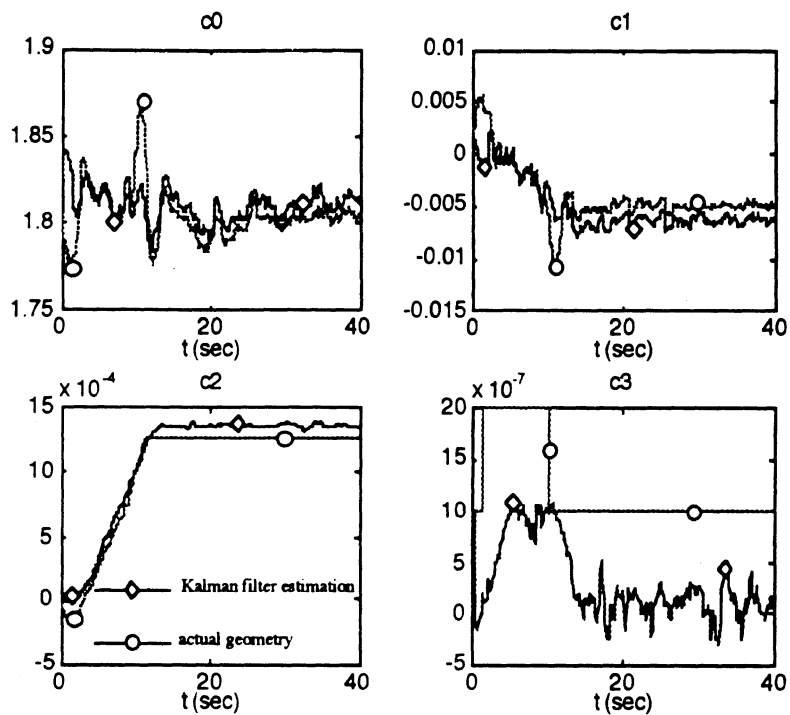


Figure IV.12 : Road Geometry Reconstruction With A 4th Order Kalman Filter

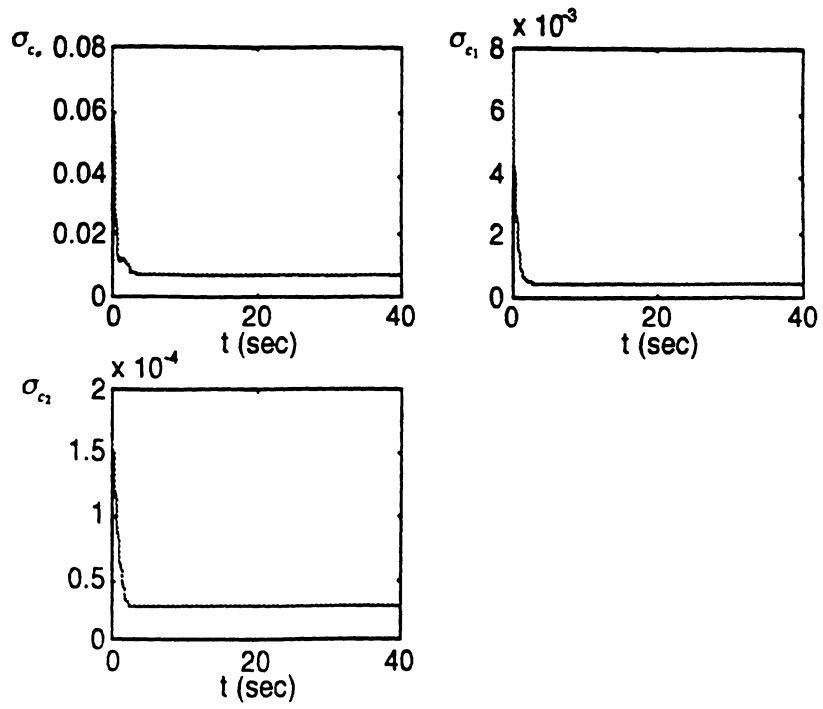


Figure IV.13 : Road Geometry Perception Uncertainty Associated With A 3rd Order Kalman Filter

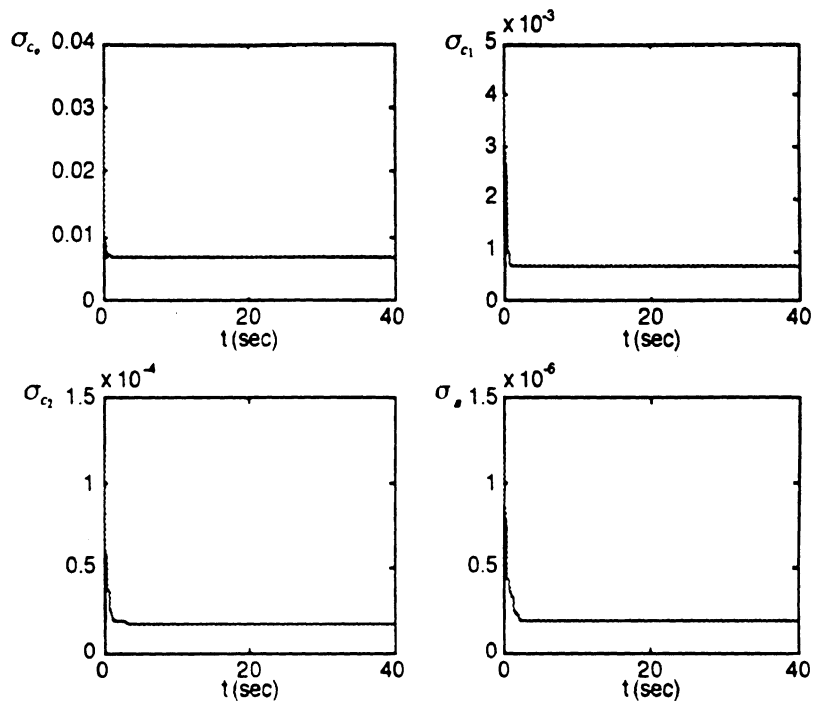


Figure IV.14 : Road Geometry Perception Uncertainty Associated With A 4th Order Kalman Filter

CHAPTER V

SUMMARY AND CONCLUSIONS

This chapter summarizes the contents of this dissertation. Conclusions from the study are discussed subsequently discussed.

Summary

This study focuses on the development of an algorithm to assess the lane tracking margin of a vehicle. This information will be utilized by an active safety system developed at the University of Michigan to prevent road-departure accidents. Such an algorithm is to calculate Time to Lane Crossing (TLC), a metric which predicts the time for a vehicle to cross the road boundary. In order to calculate time to lane crossing, future vehicle path and down range road geometry are needed. Therefore, an algorithm to predict future vehicle path and another algorithm to assess the down range road geometry are developed.

Furthermore, an algorithm to characterize the uncertainty of TLC estimation is also proposed. Such an algorithm requires the knowledge of the perception uncertainty of the roadway geometry and the prediction uncertainty of the vehicle path. Thus, algorithms which characterize the associated uncertainties of these two curves are also proposed. Before the developments of the uncertainty characterization algorithms, factors which substantially influence TLC accuracy are identified. Based on such information, algorithms are developed to compensate the errors in the predicted vehicle path and the perceived road geometry and consequently reduce the error in the TLC. Then, algorithms are developed to characterize the uncertainties associated with the residual effects.

Essentially, the TLC estimation scheme assumes a 2nd order polynomial equation to model the down range road geometry. The predicted vehicle path is obtained by projecting the vehicle path forward in a “brute force” manner. The time to lane crossing is obtained by first identifying the point where the vehicle path intersect with the road boundary and then calculating the time for the vehicle to reach the intersection. Since the projection time span of the vehicle path affect the accuracy of the TLC, this error is attenuated by introducing an interpolation scheme to increase the accuracy of the intersection and subsequently the accuracy of the TLC. Beside the development of the calculation algorithm, the frequency content of TLC for some highway driving are also studied to identify its bandwidth and subsequently the required sampling speed for TLC(i.e., the required speed for TLC calculation). On the other hand, the TLC uncertainty characterization scheme uses the knowledge of the uncertainty of the perceived down range road geometry and uncertainty of the predicted vehicle path. In other words, it uses the covariance matrices which characterize the possible deviation of the perceived coefficients from the true coefficients. In the system, the predicted vehicle path is also modeled as a 2nd order polynomial equation. The result of the TLC uncertainty characterization is the standard deviation of the possible deviation of the perceived TLC from the true value.

The vehicle path prediction algorithm has a linearized model for path projection, which includes a 2 Degree of Freedom (DoF) vehicle dynamics model such that the transient response can be accounted. Such a model is compared with a more sophisticated model in terms of the accuracy for path projection to see its validity. For path projection, it is assumed that the front wheel steering angle, vehicle yaw rate, and lane marker locations in the near-field of the vehicle will be provided. Since a study show that the future external disturbance must be considered for path prediction, a Kalman filter which uses the information of the near-field road geometry variation to estimate the vehicle lateral velocity and simultaneously estimate the external disturbance acting on a vehicle. The possibility of using one fixed Kalman filter gain for typical highway driving is also studied.

Furthermore, the estimated disturbances are characterized such that the future values can be predicted. Three different models (i.e., a piecewise model, a linear model, and a model containing a linear model and a stochastic model) are considered. The results of path prediction using these models are compared. On the other hand, the uncertainty characterization scheme for the predicted vehicle path assumes that the statistical characteristics of the measurement/estimation errors of the vehicle lateral velocity, yaw rate, and front wheel steering angle are available from off-line characterization. The result is an error covariance matrix for the 2nd order polynomial coefficients for the predicted vehicle path.

Finally, the down range road geometry perception module reconstructs the geometry with lane marker locations from an on-board sensing system which includes a far-field and a near-field sensors. The road geometry is model with a polynomial equation (either a 2nd order polynomial or a 3rd order polynomial equations). The road geometry reconstruction includes three steps ; 1) acquisition of the lane marker locations through image processing, 2) extension of the perception range of a single-far-field-sensor system to enhance the robustness of the system, and 3) road geometry modeling using the extended lane marker array. The results are the coefficients of the polynomial equation. The perception range extension algorithm utilizes the knowledge of constant lane width for typical highway and the information from the near-field sensor. For road geometry modeling, three different algorithms are developed: 1) a least square curve fit modeling the road geometry as a 2nd order polynomial equation, 2) a 3rd order Kalman filter modeling the road geometry with a 2nd order polynomial equation, and 3) a 4th order Kalman filter modeling the road geometry with a 3rd order polynomial equation. Simulations are conducted using these three road modeling algorithms and the results are compared. Since the active safety system will be demonstrated on superelevated road which introduces significant difficulty for the sensor to perceive correct lane marker locations, simulations is conducted to see the ability of the different road modeling schemes to cope with the

superelevation. Furthermore, an algorithm is developed to characterize the steady state geometry perception uncertainty associated with Kalman filtering process. Off-line simulations are conducted to obtain the steady state uncertainty. The result is an error covariance for the perceived polynomial coefficients. This algorithm is validated and simulation for a typical highway driving is conducted.

Conclusions

The results from the study of TLC calculation algorithm show that the interpolation scheme improves the TLC error by about 40%. Results also show that among the factors considered, vehicle vibration, superelevation of the roadway, external disturbances, and path projection initial condition measurement errors significantly influence TLC error. Among these four factors, vehicle vibration, which cause the sensing system to lost its orientation, is the most significant factor. Vehicle vibration causes about twice the TLC error as the path projection initial condition measurement errors do. Therefore, an accurate road geometry perception will be more important than an accurate vehicle path projection. However, it is still necessary to develop an accurate vehicle path algorithm to reduce the TLC uncertainty for the TLC to be successfully utilized by the active safety system. Furthermore, a 2.5 Hz bandwidth is identified for typical TLC for highway driving. This bandwidth leads to a requirement of at least 10 Hz for the TLC sampling rate (i.e., TLC must be calculated every 0.1 sec). It is also shown that the proposed uncertainty characterization scheme is able to predict the standard deviation of the possible time to lane crossing estimation error in a 10% accuracy on typical highway driving. Finally a simulation result shows that when an "intoxicated" driver maneuvers a vehicle on a typical highway geometry and experiences a strong side wind gust, the minimum TLC is about 2.0 sec. and the maximum TLC uncertainty is about 0.25 sec when the TLC is about 4.0 sec. These values are obtained under the assumption that the vehicle dynamics (i.e., lateral

velocity and yaw rate) and external disturbances and the front wheel steering angle can be accurately assessed.

For the study of the vehicle path prediction, it is shown that the linearized model is adequate for TLC calculation, which allows a more efficient computation of the TLC. Simulation results also show that, by applying the external disturbance estimation scheme, the predicted future vehicle path, and consequently the time to lane crossing, are significantly improved for conditions where the vehicle is subjected to significant external disturbances. However, when the external disturbances are small, an assumption of negligible vehicle lateral velocity and external disturbance is adequate.

It is also shown that a Kalman filter gain which is designed for a typical road geometry (i.e., curvature), road surface (smooth asphalt, which accounts for about 80% the typical US highway), and forward vehicle speed at 90 Km/h can accommodate a much rougher surface (which introduces severe vehicle vibration), a forward speed variation up to 110 Km/h, and a road curvature smaller than the designated curve. However, another Kalman filter gain is needed when the vehicle encounters a much tighter curve (e.g., an exit ramp on typical highway).

On the other hand, study results indicate that a piecewise constant model is the most appropriate for predicting future disturbance values. The main reason being the variation of the future disturbance trend is slow and the predicting of the stochastic component can not be satisfactorily achieved with a simple order model. A higher model to characterize the stochastic component of the disturbance for prediction may improve the result. However, a higher order model needs more computational time and may not be appropriate for on-line application.

For the study of the road geometry reconstruction, it is shown that for typical highways, the Kalman filters has a better performance than the least square curve fit under steady state conditions. However, due to the lag introduced by the filtering effect of the Kalman filter, the least square curve fit performs better for transition periods. Simulation

results also show that a Kalman filter which can successfully cope with superelevation and vehicle vibration can be developed. Furthermore, simulation results show that a 4th order Kalman filter (which models the road geometry as a 3rd order polynomial equation) can yield better performance than a 3rd order Kalman filter (which models the road geometry as a 2nd order polynomial equation). The reason being that a 3rd order polynomial equation assumes a linear varying curvature along the down range curve. Also, for superelevated roadway, a 4th order Kalman filter performances better than a 3rd order Kalman filter because a 3rd order polynomial equation fits the acquired data better than a 2nd order polynomial equation. Generally, a Kalman filter has better performance than a least square curve fitting scheme. The main reason for the superiority of the Kalman filter is because it uses information from both the image and vehicle kinematics and recursively averages the perceived road geometry.

On the other hand, the uncertainty characterization scheme is shown be the valid for predicting the road geometry perception uncertainty for typical highways. Such validation is performed with simulations featuring smooth road surface and low superelevation. Its validity must be further studied for significant superelevation and road unevenness situations.

Finally, several simulations shows that time to lane crossing, along with the existing decision module, driver physical status assessment module, and the intervention controller, seems to be capable of preventing many of the road-departure accidents. However, the capability of the whole system must be studied in a more systematic way. Many worst case conditions must be considered. A comparison between using time to lane crossing and other metrics for the decision module in terms of the computational issue and the simplicity of the decisions rule base is also interested.

