Title
Transit Integrated Collision Warning System Volume I: System Development

Authors
Chang, Joanne
Dicky, Susan
Duncil, Bart
et al.

Publication Date
2007-11-01

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Transit Integrated Collision Warning System Volume I: System Development

California PATH Program
Carnegie Mellon University - Robotics Institute

California PATH Research Report
UCB-ITS-PRR-2007-19

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation, and the United States Department of Transportation, Federal Highway Administration.

Final Report for RTA 65A0150

November 2007

ISSN 1055-1425

Transit Integrated Collision Warning System Volume I: System Development

Prepared by:

University of California at Berkeley
PATH Program
1357 South 46th Street
Richmond, CA 94804

Carnegie Mellon University
Robotics Institute
5000 Forbes Ave
Pittsburgh, PA 15213

Prepared for:

California Department of Transportation
U.S. Department of Transportation
Federal Transit Administration

Final Report for RTA 65A0150

ACKNOWLEDGEMENTS

This report presents the results of a research effort undertaken by the the University of California PATH Program and Carnegie Mellon University Robotics Institute under funding provided by the Federal Transit Administration, California Department of Transportation and Pennsylvania Department of Transportation under Federal ID # 250969449000 through RTA 65A0150. The direction of Brian Cronin is gratefully acknowledged.

The people who participated directly in this research include (in alphabetical order):

PATH: Joanne Chang, Susan Dicky, Bart Duncil, Scott Johnston, Paul Kretz, Thang Lian, Xiaoyun Lu, David Marco, David Nelson, Steven Shladover, Wei-Bin Zhang, Yongquan Zhang

CMU: Dave Duggins, Jay Gowdy, Martial Hebert, John Kozar, Rob MacLachlan, Christoph Metz, Aaron Steinfeld, Arne J Suppe, Chuck Thorpe

SamTrans: Frank Burton (Project Manager)

PAT: Dan DeBone, Rick Snyder

Special thanks are also due to the state transportation agencies for providing additional funding and contractual assistance. Specifically, the California Department of Transportation (Caltrans) and the Pennsylvania Department of Transportation (PennDOT) were instrumental in the progress of this work.

Also this work would not have been possible without the cooperation of the local transit agencies. Specifically, the Port Authority of Allegheny County (PAT) and the San Mateo County Transit District (Samtrans).

We would also like to acknowledge the work of Clever Devices. We have learned a lot and hopefully applied the lessons learned from their work of designing and installing their obstacle detection systems on transit buses.

The feedback of Eric Traube of Mitretek has also been very beneficial to the effort of this research and evaluation program.

ABSTRACT

Based on the foundation of the frontal and side collision warning systems, the Frontal Collision Warning System (FCWS) and Side Collision Warning System (SCWS) teams joined efforts to improve the collision warning algorithms. The objective of the ICWS Program is to study how frontal and side collision warning system might interface with each other, and to develop prototype ICWS systems on two buses, one at Samtrans and the other at PAT. The prototype ICWS buses have been in revenue operation in the Bay Area and Pittsburgh to collect field operational data and driver responses. The results of the ICWS design, build, and integration efforts as well as an analysis of early data collections to evolve the warning algorithms are documented in this final technical report. Evaluation and performance analysis are currently being finalized and will be issued in a separate report.

Keywords: Integrated Collision Warning System, low speed collision warning, Transit bus safety

EXECUTIVE SUMMARY

This final technical report documents technical developments conducted under the Integrated Collision Warning System Program (ICWS). It is a continuation of the development programs for the individual frontal and side collision warning systems for transit buses. The goal of the ICWS program is to integrate the advanced frontal and side collision warning systems into a unified collision warning system. A single Driver Vehicle Interface (DVI) is being developed that can effectively display warnings from both frontal and side collision warning systems and signal the driver in a manner that is effective in helping the driver avoid crashes.

Vehicle collisions have been a significant concern for transit operators. They not only result in property damage, service interruptions and personal injuries, but also affect transit efficiency, revenue and public perception. In addition to collision damage, passenger falls resulting from emergency maneuvers also contribute to an increased potential for passenger injuries and liability. A transit collision ripples through the agency and consumes additional resources to settle claims and results in significant loss of good will. Transit operators and industry stakeholders actively seek solutions to avoid collisions and have recommended that studies be conducted under the US DOT's Intelligent Vehicle Initiative (IVI) to develop transit collision warning technologies. The primary goal of the Transit IVI program is to develop technical and performance specifications for collision warning systems which can identify hazards that may potentially lead to collisions in complex urban environments and warn drivers accordingly. Based on the recommendations, Federal Transit Administration initiated the Transit IVI Program in 2000. As part of the Transit IVI Program, substantial efforts were carried out to develop frontal and side collision warning systems that can deal with the urban driving environment.

The research efforts on Frontal Collision Warning Systems (FCWS) were carried out by the San Mateo County Transit District (SamTrans), University of California PATH Program (PATH), California Department of Transportation (Caltrans), and Gillig

Corporation. Most of the San Francisco Bay Area transit agencies are participating in the project in an advisory role and have provided significant inputs to the project. The team conducted in-depth study of accident data for 35 transit agencies. The team obtained a better understanding of the causes of transit frontal collisions and the conditions in which crashes may potentially occur through field testing and data collection using instrumented buses. Human factors researchers also closely interacted with SamTrans drivers to understand their needs and expectations. Based on the accident data analysis and field data collection, the FCWS team developed sensing schemes, obstacle detection and collision warning algorithms and a DVI design. Prototype collision warning systems were instrumented onto three Samtrans buses that include radar and lidar sensors, obstacle detection and collision warning algorithms, and a DVI. These prototype FCWS systems address imminent crashes and warning needs for smoother maneuvering. As the final product, preliminary requirement specifications were developed and experimentally verified through field testing using the three buses equipped with the prototype warning system.

The research efforts on Side Collision Warning Systems (SCWS) were carried out by the Port Authority of Allegheny County (PAT), Carnegie Mellon University Robotics Institute (CMU-RI), the Pennsylvania Department of Transportation, and Clever Devices. Similar to the research on FCWS, the side collision warning team has collected field data to study the hazards on both side of the bus while the bus is in motion and has developed approaches for tracking the movement of vehicles and pedestrians using scanning laser rangefinders mounted on the sides of the bus. While vehicle collision avoidance is an important goal of SCWS, much of the emphasis of this study is placed upon pedestrian detection by assessing the movement of pedestrians relative to the sidewalk. A prototype side collision warning system was first installed on a test vehicle platform and later on a PAT bus for field experiments. Based on the test results, preliminary requirement specifications for an SCWS were developed.

Based on the foundation of the frontal and side collision warning systems, the FCWS and SCWS teams joined efforts to improve the collision warning algorithms. The objective of the ICWS Program was to study how frontal and side collision warning system might

interface with each other, and to develop prototype ICWS systems on two buses, one at Samtrans and the other at PAT. The prototype ICWS buses have been in revenue operation in the Bay Area and Pittsburgh to collect field operational data and drivers' responses. Evaluation and performance analysis are still being conducted.

This report mainly describes the following:

(1) ICWS introduction and overview.

As a driver assistance system, the primary goal of the ICWS is to predict imminent potential crashes, or collisions with objects so that it can warn the transit operator when necessary. To achieve this goal, first of all, the system needs to be capable of gathering information from both the subject vehicle and the surrounding environment (Transit bus and object sensing and detection), it then needs to track the objects (around both front and side), predicts their trajectories and assesses the threat based on all knowledge available, finally, it needs to be able to issue warnings to the operator via the Driver Vehicle Interface. These functions are implemented by the system hardware, software and algorithms.

(2) ICWS hardware, software and algorithm.

The ICWS hardware includes power adaptors, host-bus sensors, object sensors, engineering computers, cameras, video recorders and DVI. Host bus sensors measure bus speed, accelerations, yaw rate, brake pressure, throttle position, windshield wiper status, back up light, turn signals, GPS location, etc. Object sensors include frontal Lidars, two additional Radar sensors used as alternative sensors in harsh weather, one curb detector and two side laser scanners. Three PC104 engineering computers are used for ICWS data acquisition, archiving and warning algorithm generation. Recorders were developed to save the video streams from cameras that are installed to provide different views around the bus. These recorders are part of the research system and are not necessary for the final system. The FCWS is connected with the SCWS using serial ports. The system hardware provides the platform for system software and application algorithms.

The ICWS computers are running QNX (FCWS) and Linux (SCWS). Although the specific implementations are different, a "single-writer multiple-reader" model is used as the basic protocol for inter-process communications in the system software. The FCWS exchanges data with the SCWS via a custom-built protocol. Built on the system hardware and software are the ICWS application algorithms.

The ICWS algorithms include system modeling, object tracking and threat assessment, system fault detection and recovery.

The biggest challenge for the FCWS is that buses usually serve in urban/suburban environment where too many objects may trigger false alarms. Hence it is a difficult problem to detect real imminent crashes and give drivers timely warnings while suppressing excessive false alarms. The third generation algorithm PATH developed for forward collision warning has five unique features (1) Modeling moving targets with non-holonomic constraints. (2) Taking into account the driver's role in the system. (3) Eliminating Coriolis effect. (4) Suppressing finite size object effect. (5) Using required deceleration as threat measure. All these features address reducing the nuisance alarms to a great extent as shown in the data analysis and field testing.

The SCWS uses the linear feature tracker combined with history-based track validation and is able to generate reasonably accurate velocity estimates for cars and pedestrians in a cluttered urban environment, while giving a low rate of spurious motion indications that can cause false alarms. The estimation of acceleration and turn rate appears to improve prediction of future positions.

The ICWS has four categories of fault from the system point of view: power fault, sensor fault, DVI fault, and engineering computer fault. The practical fault detection algorithms and detection strategies are proposed and system fault reporting and system recovery methods are introduced.

(3) DVI development.

The DVI for the ICWS was an extension of the UC Berkeley PATH experience over the previous two years on transit bus operation. The design of the DVI took into consideration the characteristics of the bus design and special needs for transit drivers. Many discussions with and feedback by Foster Miller, Inc and members of the transit community (SamTrans, PATH and a dozen of transit agencies in the Bay Area) were also used for the current design. The current design of the DVI will be evaluated in simulation by PATH and SamTrans as part of this program and evaluated by transit operators. These results will be incorporated into the final performance specifications for an ICWS.

(4) ICWS field testing, data analysis and system evaluation.

The prototype system has undergone detailed testing and analysis. Simulation tools and playback tools for ICWS were developed to analyze the raw data, as well as test and evaluate the system. The simulation tools regenerate all intermediate variables and trace back each detail of the processing performed. Playback tools are used to show the video files together with all engineering data so that we can have a comprehensive understanding of the scenarios.

Series of tests were conducted at both RFS and Crows Landing to test the FCWS. A leading vehicle and a bus were the main focuses of the testing. A fifth wheel, an accelerometer and a string pot were installed on the leading vehicle and synchronized with the FCWS. A string connected to the bus was used to measure the distance between the leading vehicle and the bus. The leading vehicle ran at low/medium speed with the bus following it at a reasonably safe distance. The estimations (from the FCWS algorithm) of the essential variables: relative positions, target speed and acceleration were compared with the raw measurement from the Lidar, the string (when applicable), the fifth wheel or the accelerometer on the leading vehicle. The result is a good match as shown later in this document.

The FCWS warning scenarios were categorized and analyzed using a three-step quantitative approach. The three scenarios include: moving/stopped target ahead on straight road; stationary target roadside on curved road; overhead obstacles on

declining/flat road are analyzed. Improvements were made to the algorithm to include features that turn the nuisance warning to a friendly reminder. With the road geometry information (e.g., more precise GPS and digital map system), driver status information, target properties and crash data analysis, some of the nuisances induced by curved roads and overhead obstacle problems could be overcome.

Bench tests were also conducted for the SCWS system to verify the resolution, the accuracy of the object sensors and the accuracy of velocities measured by DATMO. Closed Course testing was conducted to verify the warning algorithms by constructing situations in which cardboard objects came in contact with the bus to verify the true positives and look at the relative timing of the incident prediction and DVI activation.

The majority of the positive alarms SCWS issues are understandable by the transit operator. Many of the false positives are not very seriously false (velocity off slightly), and the driver might not even consider them nuisances. When a large amount of false positives are seen by the operator, the problem can be traced back to sensor failures (e.g. laser scanner not level due to the bus tilting or road variation and picking up ground returns). The number of serious false positives which will be present even if all the sensors work correctly is small and due primarily to velocity outliers.

(5) Transit CWS Simulator.

The SamTrans FAAC™ simulator is being modified to incorporate CWS functions, which will allow us to create specific scenarios of interest, including scenarios too dangerous to test on real buses, to which large numbers of drivers can be exposed, providing us with a much more extensive data set than we could obtain from in-service operation of two buses. From the simulator experiments, more extensive data sets will be obtained, which will be used to analyze driver behavior changes due to the introduction of ICWS and for further optimization of the warning algorithms and DVI.

(6) ICWS commercialization and further research recommendations.

As more advanced sensors and more powerful computers are available, together with further integration of the FCWS and the SCWS, the ICWS will have fewer sensors needed to maintain the same or even higher sensing capability and process all functions using only one computer, resulting in smaller volume and less cost. More research is being conducted to improve the ICWS tracking algorithm and threat assessment algorithm. Research on use of GPS/Digital map and sensor data fusion will also be introduced to help ICWS performance improve.

Table of Contents

List of Figures

List of Tables

INTRODUCTION

1.1. Background

The Federal Transit Administration has been funding work over the last five years to shorten the commercialization and deployment cycle of collision warning systems for the transit industry. FTA developed initial cooperative agreements with San Mateo Transit Authority (Samtrans), California Department of Transportation (Caltrans), University of California at Berkeley PATH Program (PATH) and Gillig Cooperation to develop Frontal Collision Warning Systems (FCWS), and with Port Authority of Allegheny County (PAT), Pennsylvania Department of Transportation (PennDOT), Carnegie Mellon University Robotics Institute (CMU RI) to develop Side Collision Warning System (SCWS) and with Ann Arbor Transit Authority and Veridian Engineering Division to develop Rear Collision Warning Systems (RCW). The focus of these efforts was to fund technology development to the point where a commercial system could be developed. In addition, existing Side Object Detection systems using ultrasonic automotive sensors were put into operational field tests to learn how to introduce technology onto a transit platform in a way that made it acceptable to operators, maintenance personnel and management. Initial results of this work were an advance in the technology usable for collision warning systems, specifications for advanced collision warning systems, and the evaluation of 100 commercially available side object detection systems in revenue operation.

The next step in this program was to determine what it would take to field an integrated advanced frontal and side collision warning system and conduct a more limited field test on ten commercial systems. The objectives for this work were as follows:

1. Develop a Functional ICWS
2. Create System Acceptable to Operators
3. Prove Technical Feasibility Through Field Test of Prototype System
4. Demonstrate a Potential for Reduction in the Severity and Frequency of Collisions

In 2002, FTA entered into cooperative agreements with a consortium that included San Mateo Transit Authority (Samtrans), Port Authority of Allegheny County (PAT), California Department of Transportation (Caltrans), Pennsylvania Department of Transportation (PennDOT), University of California PATH Program and the Carnegie Mellon University Robotics Institute. Prototype hardware designs and algorithm research were focused early in the project to field an advanced Integrated CWS. This report documents the results of this research prior to the evaluation of the prototype advanced ICWS. The final evaluation report for this Integrated CWS will be produced in June 2005

1.2. Scope

As detailed in the Preliminary ICWS Performance Specifications, the primary goal of an integrated collision warning system is to predict imminent potential crashes, or collisions with objects and warn the transit operator. To achieve this goal the collision warning system has the sensing capability to gather information from both the subject vehicle and the surrounding environment (Transit bus and object sensing) and display it to the operator via the Driver Vehicle Interface. The ICWS fulfills eight functions as illustrated in Figure 1, including object sensing, transit bus sensing, the basic signal and data processing functions shown within the dotted lines and the Driver Vehicle Interface (DVI). At the beginning of this program, these functions were examined to see what research needed to be done to accelerate the deployment of commercial systems.

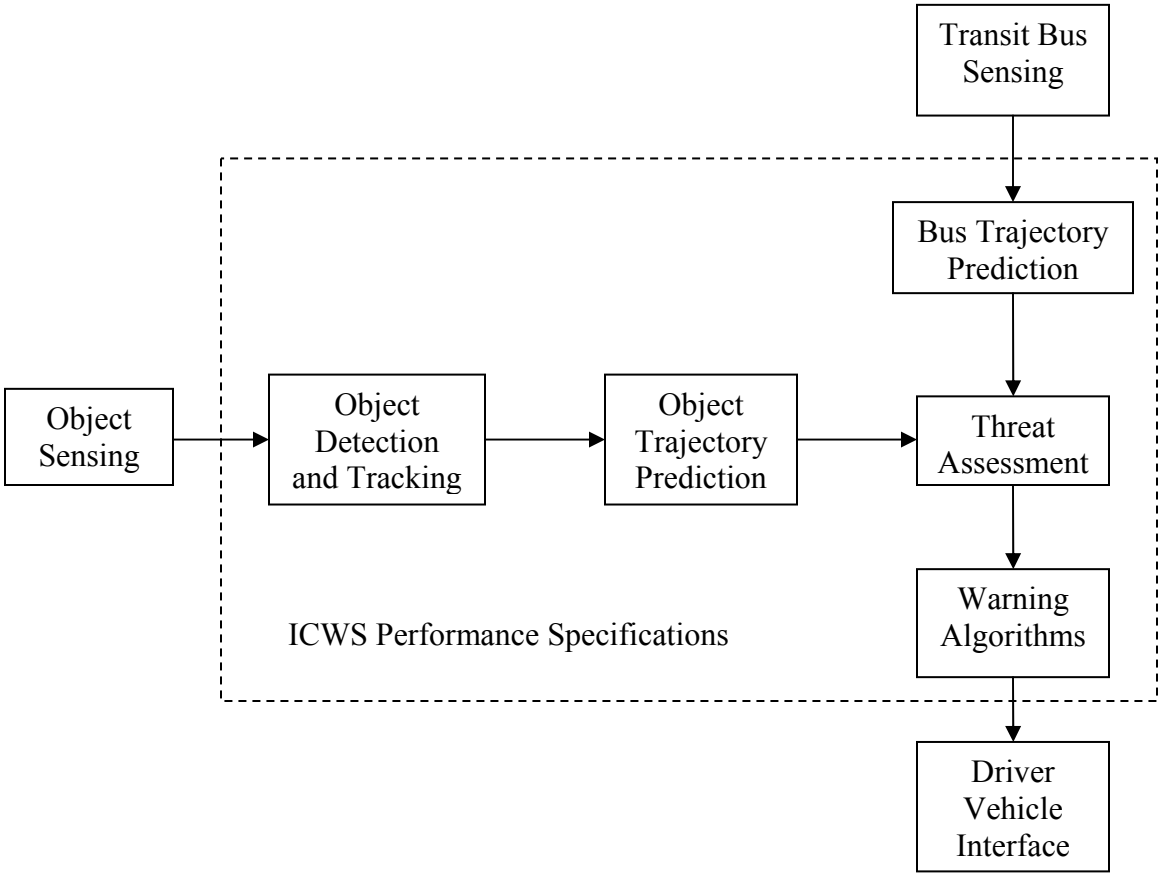


Figure 1 - Basic ICWS Algorithms

1.2.1 Transit bus and object sensing

Subject vehicle status sensing refers to the acquisition of information on operator actions and the current kinematic states of the bus. Examples of subject vehicle status sensors are: speedometers, accelerometers, brake pressure sensors, steering angle sensors, and GPS receivers. Commercial sensors exist in this area and only need to be specified and incorporated into a commercial system. The goal of this program was to determine what sensor information is necessary and should be defined in the ICWS Specifications.

Object sensing refers to the acquisition of information from the environment (for example, road curvature), the presence of other objects (for example, vehicles and pedestrians) and the current kinematic states of the objects. Examples of sensors for object status sensing are microwave RADARs, laser LIDARs, imaging sensors and

ultrasonic sensors. The sensors used in this early prototype ICWS were the more expensive and higher performance ones in order to determine where the performance level should be set for a commercial system. The development of a cheaper sensor for a commercially viable system is discussed more fully in the Recommendations Section.

1.2.2 ICWS Algorithms - Signal and data processing

The main research component of this program involved developing the algorithms necessary to process the incoming data and generate warnings to a transit operator. Research was accomplished in each of the five algorithm areas defined below.

The function of object detection and tracking is to tell if there is an object within the monitoring coverage of the collision warning system. The state of the art in object tracking was not sufficient to develop ICWS systems that could accurately and in a timely fashion present objects to be tracked. The conversion of sensor data to object data represented a large challenge to developing these systems and significant effort was devoted to this cause.

The function of object trajectory estimation is to determine the present and future kinematic states of an object. The states included such information as spatial position, velocity and acceleration of an object. The algorithms for predicting the trajectory are straightforward and did not need to be researched, but the importance of each of the states for the warning algorithms were examined and the results incorporated in the current set of ICWS Specifications.

The function of bus trajectory estimation is to determine the present and future kinematical states of the transit bus. The states included such information as spatial position, velocity and acceleration of the bus. Once again, the algorithms for predicting the trajectory are straightforward and did not need to be researched, but the importance of each of the states for the warning algorithms was examined and the results incorporated in the current set of ICWS Specifications.

The function of threat assessment is to determine the likelihood of collision between the transit bus and an object by assessing such factors as the probability of a collision, time to collision and the likely severity of a collision. These factors form the basic data used in the warning algorithms. As such, they were a primary part of the research and used as metrics in the evaluation phase of this program.

The warning algorithms determine the safety level of the transit bus and its environment based on the threat assessment. One important aspect of the warning algorithms is to use heuristics based on threat assessment, object location and timing to minimize the number of nuisance alarms. A framework for these heuristics was developed which allows future heuristics to further tune the system based on data obtained during revenue service during the evaluation part of this program.

1.2.3 Driver-vehicle interface (DVI)

The DVI is a critical component of the ICWS, which displays the outputs of the ICWS to the operator for appropriate corrective action. These signals are presented via displays whose modalities include visual and the capability for auditory. An effective DVI must be able to bring the driver's attention to the hazardous situation while he/she performs a variety of driving and non-driving tasks and does not pose additional workload or distraction. The DVI for the ICWS was an extension of the UC Berkeley PATH experience over the previous two years on transit bus operation. The design of the DVI took into consideration the characteristics of the bus design and special needs for transit drivers. Many discussions with and feedback by Foster Miller, Inc and members of the transit community (SamTrans, PATH and a dozen of transit agencies in the Bay Area) were also used for the current design. The current design of the DVI will be evaluated in simulation by PATH and SamTrans as part of this program and evaluated by transit operators. These results will be incorporated into the final performance specifications for an ICWS. A more thorough discussion of the DVI in Section 7 of this report titled DVI Development.

1.3. Organization of Content

This report documents the research undertaken as part of this program by two universities, with each one describing their respective parts of the system. As such each major section is structured to discuss fully either the side component of that section or the frontal component. Care has been taken to title the subsections sufficiently to show this distinction so as not to confuse the reader. This document is divided as follows:

1 Introduction (Background, Scope, and Organization of Content)
2 Integrated Collision Warning System (System Description and Integrated ICWS)
3 System Overview
4 Hardware Development
5 System Software
6 Algorithm Development
7 DVI Development
8 Data Analysis and Evaluation
9 Calibration and Testing
10 Transit CWS Simulator
11 Recommendations

Appendix A: Acronym Definitions
Appendix B: Related Documents
Appendix C: Published Papers
Appendix D: Conversion Tables

1.4. System Description

The integrated collision warning system is functionally divided into FCWS and SCWS processors dealing with frontal and side collision detection and warning signal generation. This modularity makes it easier to specify and integrate a rear collision warning system in the future for 360 degree situational awareness. The warning information is presented to the transit operators through an integrated Driver Vehicle Interface. Additionally, data collected through each processor is shared with the other processor and stored for easier data analysis. The elements of the ICWS include:

- Vehicle state estimation and bus signals interface – this includes the common infrastructure that each collision warning system needs such as vehicle position, speed, heading, door open/close, turn signals, etc.
- Frontal collision processor –Includes sensors for detecting frontal obstacles and vehicle status information for determining risk levels and for generating warning outputs to integrated DVI. Appropriate sensory information is exchanged with the side collision processors.
- Left and Right Side collision Processors - Includes sensors for detecting side obstacles and vehicle status information for determining risk levels and for generating warning outputs to the integrated DVI. Appropriate sensory information is exchanged with the frontal collision detection processor.
- Integrated DVI – to display the warning to the operator
- Data storage – Stores video and digital data for later analysis and evaluation. The data collected by both frontal and side collision detection systems are stored with time synchronized data formats for post processing analysis.

A top level overview showing the general configuration of the ICWS and a more detailed hardware / architectural layout are shown in the next two figures.

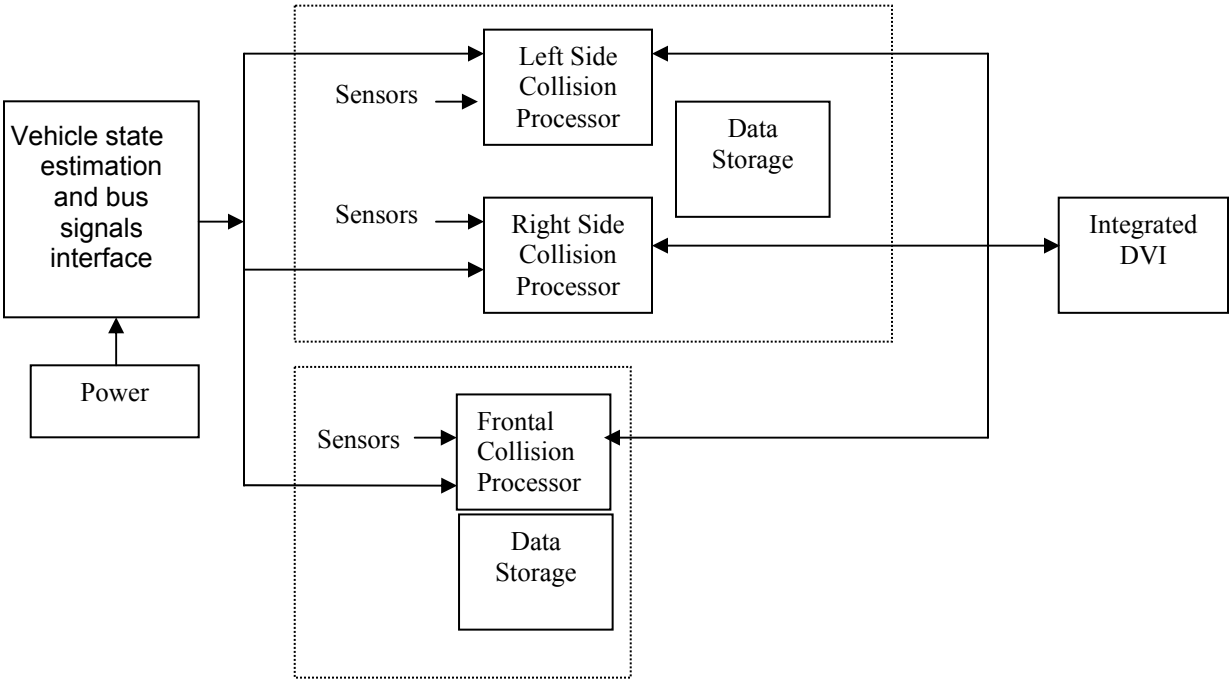


Figure 2 Configuration of Integrated Collision Warning System

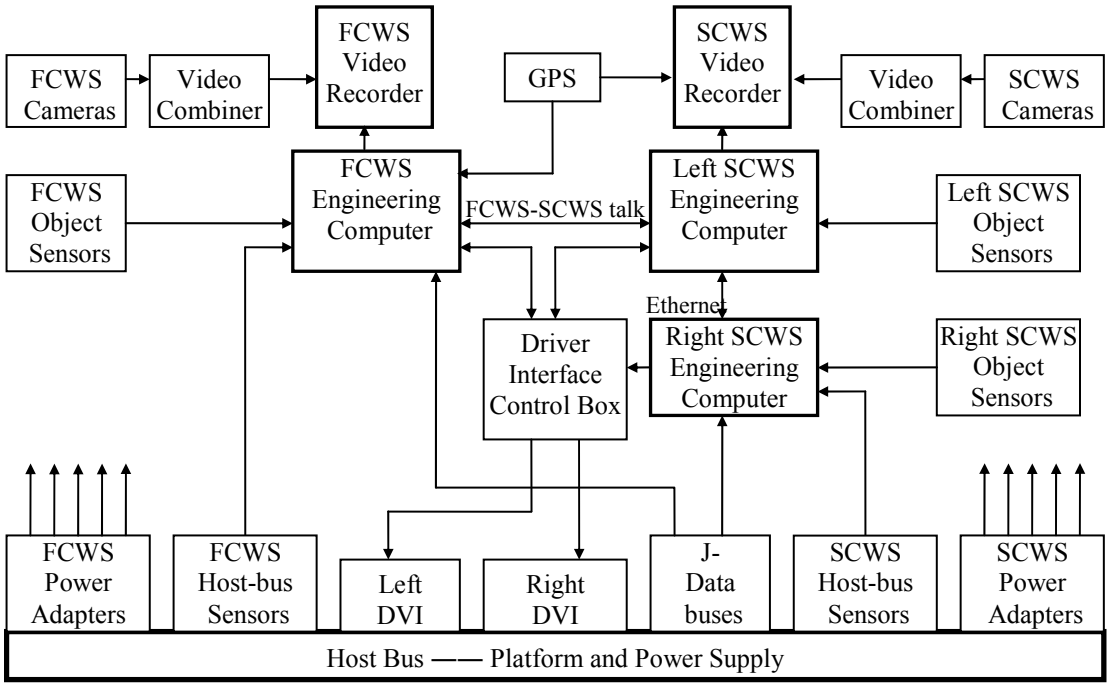


Figure 3 System Architecture

1.5. Integrated ICWS

What do we mean by an "integrated" ICWS? This question would naturally occur as you look at the functional configuration figure above, since it appears that each of these systems is operating independently. The overarching design philosophy for this early prototype combining the FCWS and SCWS was that the frontal and side collision warning systems should be closely integrated through information integration. In implementing the hardware, we wanted to ensure that each system can operate even if the others go down. With separate computing systems this dictates a level of independence that does not need to be reflected in the end commercial product. This integrated prototype is integrated at the information level primarily through the RS232 interface and the time synchronization of data streams to allow integrated post processing data analysis.

A visual integration occurs though the common Driver Vehicle Interface and the Driver Interface control box. This display to the operator integrates the warnings by displaying them on a single set of DVI's. Lastly, a common coordinate system has been defined to allow the meaningful passage of data between the FCWS and SCWS systems.

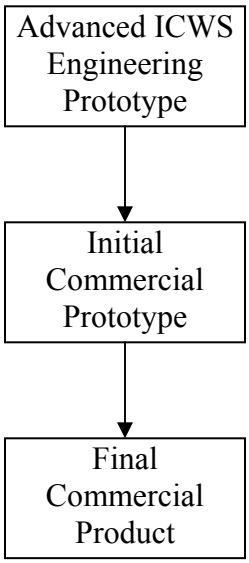


Figure 4 Three stages towards a commercial product

As shown in the above figure, this early prototype of an ICWS allows us to test the concepts and develop an integrated set of performance specifications for a commercial prototype of an ICWS. Offline data analysis will show additional levels of integration potential by revealing the benefit of real time information transfer between systems. Human factors testing will determine if a transit operator can assimilate the current DVI information. This familiarity and experience with the combined systems will show additional areas for further integrated specifications.

Once these integrated specifications are released, there are still two stages left to developing the final commercial ICWS. The first is the initial commercial prototype and the second is the commercial product itself. The commercial prototype will involve the integration of hardware subsystems, elimination of redundant components and interfaces, common software modules and overlapping sensors. This additional step before the development of a final commercial product is necessary in order to provide for the integration of the forward and side collision algorithms using a common algorithm base. This is discussed more fully in the Recommendations Section of this report.

1.6. Sensing Needs

The farthest detectable range in the same lane is 100m (330ft). The closest detectable range in the same lane is no greater than 3m (10ft). The maximum detectable side-looking angle from the front bus corners is 30 degrees. The detectable lateral position for the forward sensors is over 6m (20ft). The side looking sensors will closely track objects that are within 3m of the bus however, objects will be detected as far as 50 meters away.

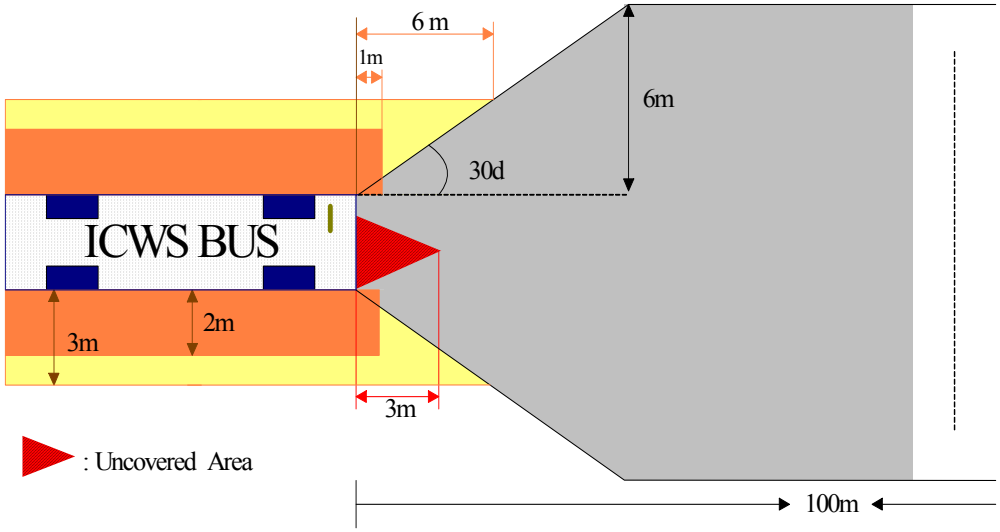


Figure 5 Integrated system spatial coverage illustration

SYSTEM OVERVIEW

1.7. FCWS System Overview

1.7.1 The goals of the FCWS

The goals of the transit Frontal Collision Warning System (FCWS) under the context of this project include:

1. Address imminent crashes.
2. Provide warnings for smoother maneuvering.
3. Provide warnings when a bus is too close to a forward vehicle.

1.7.2 FCWS functions

The operation environment for ICWS is significantly different from the environment that automobile CWS deals with in the following two ways. First, most of the transit frontal crashes occurred in urban areas while previous studies on collision warning and collision avoidance have mostly focused on highway applications, freight trucks, and light-duty passenger cars. The urban environment presents considerable challenges with respect to the diversity of obstacles to be detected and different traffic patterns. The transit FCWS must be able to deal with the complex urban environment besides the one that current commercial CWS address. Second, transit bus drivers are professional and experienced drivers who may have different needs for a FCWS. Transit drivers have also expressed concern regarding the presentation of warnings that can be seen by passengers. Bus passengers might find warnings for advance cues of potential threats to be annoying and potentially alarming. There is still a great need for human factors research in FCWS within the transit environment.

Despite the differences between the collision warning applications, the FCWS for transit buses requires the same functional elements that are required by other CWS. The principal functional element of a CWS is sensing and detection of presence of hazardous objects. Furthermore, this function must be able to match the complex urban environment. The second functional element is warning generation. It processes the sensor information to “detect” the targets that may potentially be dangerous to the bus,

then determines the threat level and generates warnings at a reasonable good timing if necessary. The third functional element is the Driver Vehicle Interface (DVI) which issues the warning message to the driver. The figure below depicts the functional description of the collision warning system:

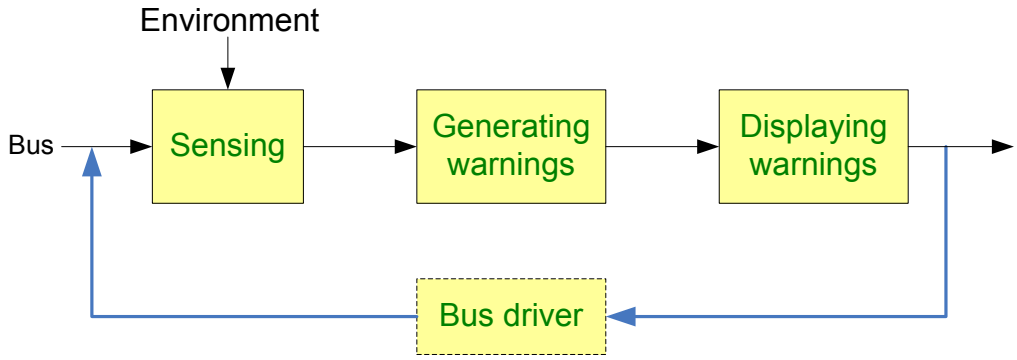


Figure 6 Functions of frontal collision warning system

The figure below shows the architecture of the FCWS system PATH developed:

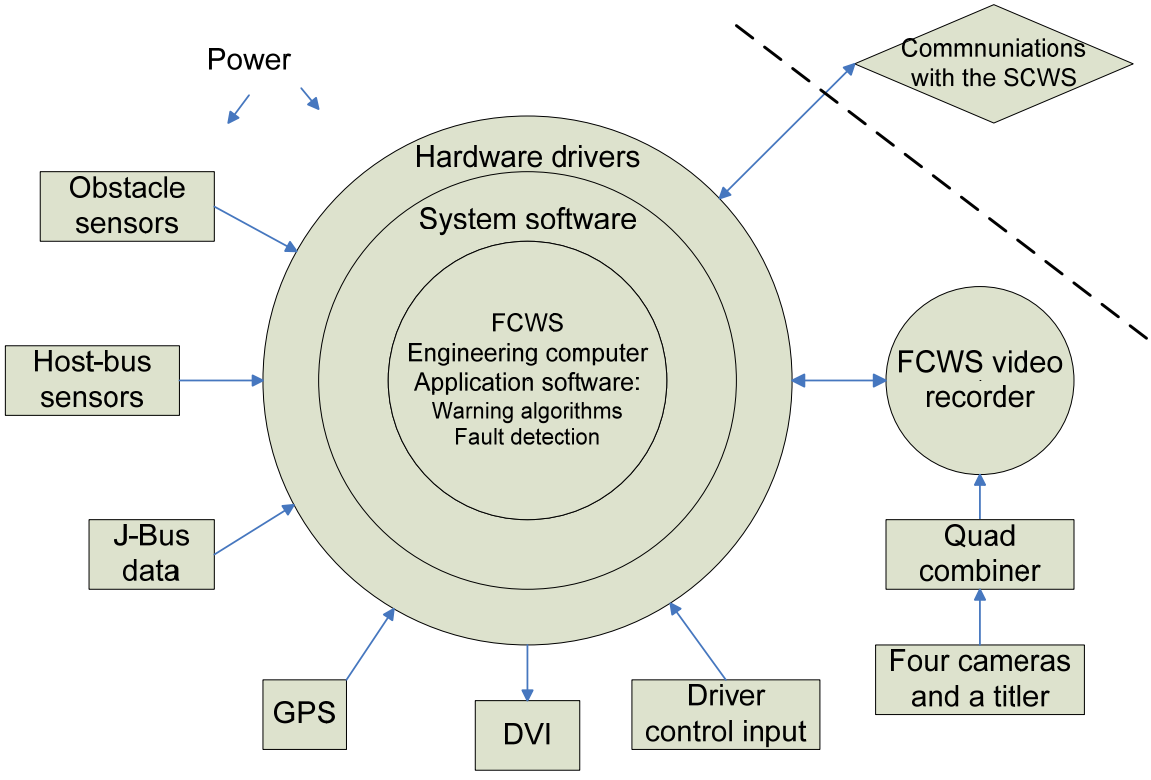


Figure 7 FCWS system architecture

1.7.3 FCWS system hardware

The FCWS system hardware consists of power and adapters, two PC104 computers (one is an engineering computer and the other is a video recorder), sensors (including five obstacle sensors, and host-bus sensors) and cameras (frontal looking camera, driver-side looking camera, passenger-side looking camera and interior looking camera), and human machine interface (including a driver control box and two DVI bars).

The engineering data, which mainly includes the obstacle sensor data and the host bus sensor data, is recorded and processed by an engineering computer which is a PC104 computer running QNX operating system. The obstacle sensors selected by PATH to capture the environment around the bus include commercially available mono-pulse millimeter-wave RADARs and scanning infrared lasers. Both the RADAR and scanning laser measure distance and azimuth angle of multiple targets. The RADAR units are mounted on the front bumper, one on each end, pointing forward. The Denso LIDAR unit is mounted near the center of the bumper, pointing forward. Host bus sensors measure the bus status, including bus speed, accelerations, yaw rate, brake pressure, throttle position, windshield wiper status, back up light, turn signals. Other sensors include a GPS system and a driver control box, which controls the brightness of the DVI bars and the system sensitivity level.

Video streams from four cameras are combined together with a titler by a quad combiner and recorded by another PC104 video-recording computer running QNX 6. It is synchronized with the engineering computer in real time through RS232 serial ports.

The FCWS and the SCWS communicate with each other through RS232 serial ports. The two systems exchange information that the other party may need.

1.7.4 FCWS system algorithms

The prototype FCWS algorithm was developed based on the data fusion and decision making model developed by the Joint Directors of Laboratories (JDL) data fusion sub-panel.

1.7.4.1 The JDL data fusion process model

The JDL data fusion model provides a top-level framework of data fusion systems, and defines terms commonly used in different areas. The top level of the JDL data fusion process model is shown in the figure below:

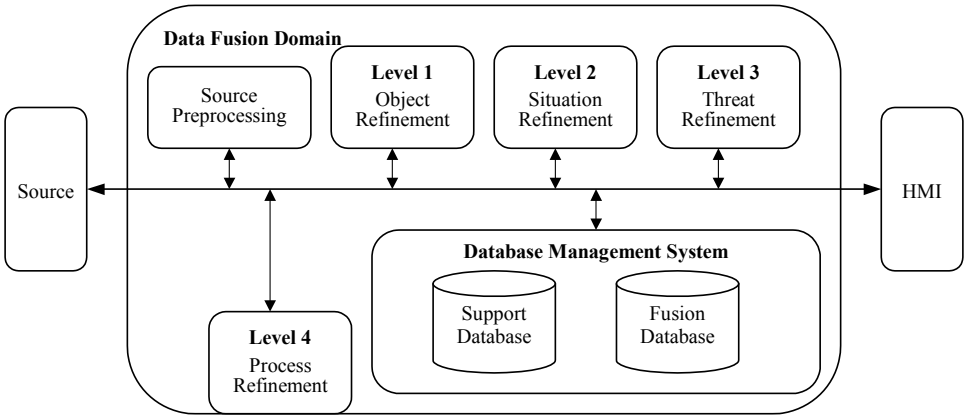


Figure 8 JDL data fusion process model

The JDL model is a generic model for common understanding and discussion. It has defined levels of processes to identify functions and techniques. The model has built a common base for researchers and system developers working in different areas. With the help of this model, we can adopt a lot of approaches and techniques developed for other applications, such as robotics, Computer Integrated Manufacturing Systems (CIMS), airport surveillance and air traffic control, to develop a CWS.

SOURCE	The sources provide information at a variety of levels ranging from sensor data to *a priori* information from databases to human input.
PROCESS ASSIGNMENT	Source preprocessing enables the data fusion process to concentrate on the data most pertinent to the current situation as well as reducing the data fusion processing load. This is accomplished via data pre-screening and allocating data to appropriate processes.
OBJECT REFINEMENT (Level 1)	Level 1 processing combines locational, parametric, and identity information to achieve representatives of individual objects. Four key functions are: • Transform data to a consistent reference frame and units

	• Estimate or predict object position, kinematics, or attributes • Assign data to objects to permit statistical estimation • Refine estimates of the objects identity or classification
SITUATION REFINEMENT (Level 2)	Level 2 processing attempts to develop a contextual description of the relationship between objects and observed events. This processing determines the meaning of a collection of entities and incorporates environmental information, *a priori* knowledge, and observations.
THREAT REFINEMENT (Level 3)	Level 3 processing projects the current situation into the future to draw inferences about the enemy threats, friendly and enemy vulnerabilities, and opportunities for operations. Threat refinement is especially difficult because it deals not only with computing possible engagement outcomes, but also assessing an enemy's intent based on knowledge about enemy doctrine, level of training, political environment, and the current situation.
PROCESS REFINEMENT (Level 4)	Level 4 processing is a meta-process, i.e., a process concerned with other processes. The three key level 4 functions are: • Monitor the real-time and long-term data fusion performance • Identify information required to improve the multi-level data fusion product, and • Allocate and direct sensor and sources to achieve mission goals.
DATABASE MANAGEMENT SYSTEM	Database management is the most extensive ancillary function required to support data fusion due to the variety and amount of managed data, as well as the need for data retrieval, storage, archiving, compression, relational queries, and data protection.
HUMAN-COMPUTER INTERACTION	In addition to providing a mechanism for human input and communication of data fusion results to operators and users, the Human-Computer Interaction (HCI) includes methods of directing human attention as well as augmenting cognition, e.g., overcoming the human difficulty in processing negative information.

Table 1. JDL data fusion process model

The JDL model however, is not a universal architecture for practical applications. It does not specify the level of data fusion. Data fusion level is an application-specific problem. To define the collision warning system architecture, analysis of the system function requirements is needed.

1.7.4.2 Requirements of the transit FCWS

All the functions defined in the JDL model except level four are requirements of transit FCWS. First of all, the source preprocessing must be performed to eliminate the

unwanted signals and to detect the objects of interest. The sources here may include object sensors such as RADARs, LIDARs, CAMs, GPSs, and subject vehicle sensors such as speedometers, accelerometers, yaw rate and braking pressure sensors. Sensors are used to convert the measurable elements of the physical processes of the environment into electric parameters. The process to convert the physical process elements into electric parameters is observation. Some unwanted signals, such as pavement clutter, road-side trees and traffic signs, etc., and interference from the same kind of sensors mounted on other vehicles or from other sources, as well as noise from internal components of the sensor, must be suppressed in order to pickup the real object signals. The preprocessing is the process to figure out, from one or more observations, whether an object exists or not, and to measure the status of the existing object.

The process of finding out whether an object exists or not is defined as detection. It is a probabilistic test of hypotheses. In the simplest situation, we have two hypotheses, H1 and H0, representing the object's presence and absence respectively. The probability of being H1 while the object does exist, viz. probability of correct detection (P_d), is always less than 1. The probability of being H1 while the object does not exist, viz. probability of false alarm (P_{fa}), is always greater than zero.

The process to measure the object status, such as location and velocity, from the observations, is defined as estimation. The estimated parameters are random variables, because they are calculated from observations and the observations are random samples from a probabilistic set.

The results of detection and estimation are called measurements in this report. A measurement comes from single or multiple observations. Measurements, as functions of time, are stochastic processes in reality. Level 1 processing should then be performed to detect the processes and to estimate parameters of the processes. It is assumed in most cases that false alarms are less possible than real objects to form continuous processes. The detection of the process will eliminate the false alarms and determine when a process begins and when it ends. The estimation of the process will refine the measurements. The

results of detection and estimation of processes are called tracks. The process to initiate, manipulate and end tracks is called tracking.

A track represents a stochastic process converted by a sensor from the physical process of an object. The parameters of a stochastic process are correspondent to the parameters (as functions of time) of an individual object. To develop a description of the current relationship among multiple objects and events in the context of their environment, level two processing is needed. Tracks from different sensors may represent the same object. These tracks must be fused into one track. This process is called track-to-track fusion, and the fused track is called the system track. After fusion, a system track becomes a refined unique representation of an object. The history of the tracks and the relationship among the tracks as an aggregation represent the traffic scenario. Once the scenario is described, level three processing is needed to assess the threats. Threat assessment is the process whereby the current situation is projected into the future to assess the severity of a potential traffic accident. Knowledge about vehicle kinematics, traffic, and the environment is needed for the assessment. Human behavior may also be used for this assessment. Once a potential threat is detected and exceeds the threshold, a warning will be sent to DVI. Level four processing is not needed in an FCWS, because the developers of the system and the vehicle drivers will perform this function outside of the system.

1.7.4.3 Architecture of the transit FCWS warning algorithm

Studies on collision warning/avoidance during the past few years have built a good foundation for the bus FCWS design. Sensors such as RADARs and LIDARs for automobiles have been developed. Some sensors have been integrated with built-in Digital Signal Processors (DSP) which can perform source preprocessing with some also able to perform level one processing. It is convenient to adopt these intelligent sensors in the bus FCWS. Threat assessment algorithms have been studied and various severity measures have been proposed, e.g. TTC, warning distance, warning boundaries.

To develop a collision warning algorithm architecture from the JDL model, one of the key issues is to decide where to fuse the data in the data flow. We prefer the track-to-

track fusion that matches the state-of-the-art technology of the sensors and helps us focus on higher level processing. The figure below is the block diagram of the transit FCWS warning algorithm architecture. Details of the warning algorithm are described in the algorithm chapter.

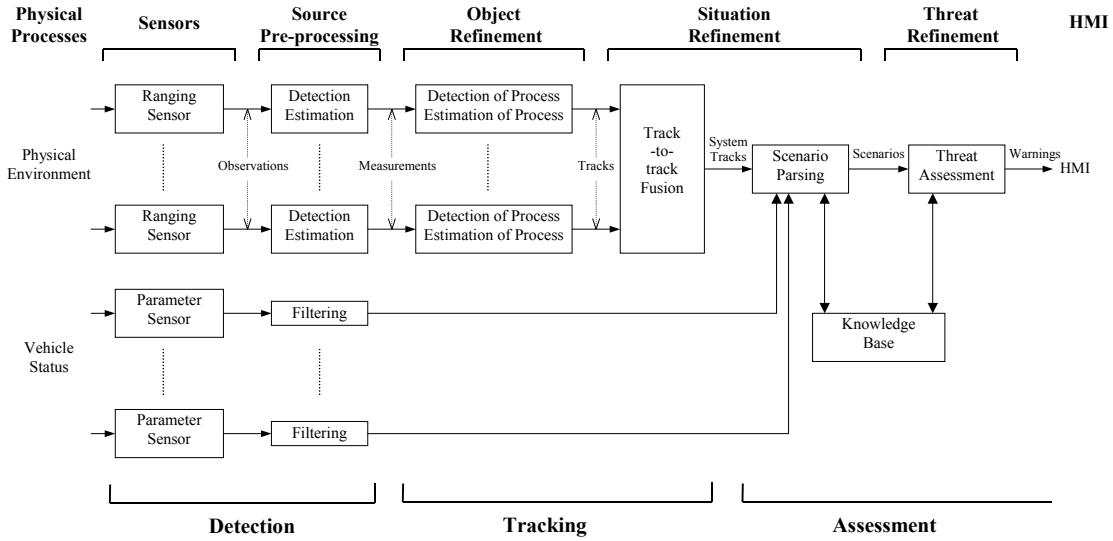


Figure 9 The architecture of the transit FCWS warning algorithm

1.8. SCWS System Overview

The computer systems on the bus have four major tasks:

1. Data collection
2. Data processing
3. Data storage
4. User interface

All items in the above list are critical to a functioning side collision warning system except for the data storage task, which is a necessary research tool. The major data processing task is the detection and tracking of objects from the range data from the SICK Laser ranger. Early in our design process, it was recognized that this task would consume the largest share of our processor power. However, the tracking problem of objects on either side of the bus easily lends itself to a bilateral partitioning, leading us to an architecture of two semi-independent computers.

While the bulk of the processing is object tracking, most of the data collected by the system comes from a set of external cameras that allow the algorithm designer to better interpret the rather abstract range finder scans. This video data is not part of the core SCWS. Since it would be impractical to move so much data over the in-vehicle network, we added a third computer that serves as a central data repository and digital video recorder.

One further observation is that the right side of the bus is more important that the left since it faces the curb and is most often near pedestrians. For this reason, all the crucial vehicle state sensors are connected directly to the right processor and shared with other computers via an Ethernet network. The right computer is also the master clock in the system, allowing the different computers to properly interpret the shared data. Finally, the right computer is the central code repository. Source code, executables, and configurations all reside on the right computer's hard disk, but are transparently shared via NFS (Network File System).

1.8.1 SCWS data acquisition and communication

There are 5 major data sources in the system.

1. Vehicle State
2. LIDAR Data
3. Video Data
4. Ancillary Data
5. FCWS/SCWS Interface

Two CPUs collect and process left and right side LIDAR data, respectively Vehicle state is a critical component of the SCWS, and is therefore attached to the right processor, the more important of the two, and then shared with other computers in the system. Vehicle state includes odometry and IMU data, both of which are instruments that connect to a serial port.

While both the SCWS and FCWS require a pose estimate, they each compute their own estimate and do not share this information. This increases the independence of the system

at the expense of redundant hardware. However, this is more than justified by eliminating the additional downtime that would come with complete interdependence. The two systems do share a physical connection to a PATH installed odometer.

Video data sources include a curb detector and a forward looking camera that serves as a curb predictor. Since the curb detector is more reliable than the predictor, this instrument is attached to the right processor, with the predictor camera on the left. Only samples of the raw video data from either instrument are saved.

Where possible, we have tapped into existing vehicle systems to supplement our understanding of what the driver and vehicle are doing. The J1708 data bus broadcasts engine related information, such as vehicle speed and accelerator pedal position. The DINEX data bus broadcasts the status of turn signals, warning lights, head lamps, and tells us which doors are open and whether a passenger has requested a stop.

The DINEX system is not present on all buses, in which case we rely on instrumentation installed by PATH. Since this data is not critical, the FCWS collects this data and shares it via a serial link. This serial link is the only form of communication between the two systems. It is also used to hand off objects tracked with one system that are moving into the field of view of the other.

1.9. FCWS Obstacle Detection Sensors

1.9.1 FCWS obstacle sensors

The figure below shows the layout of obstacle sensors and video cameras (Front view). The positions of each sensor/camera are measured in a FCWS reference frame. The frame is originated on the ground under the center point of the frontal bumper with positive directions of x-, y- and z- axes pointing to driver-side, upward, and forward respectively.

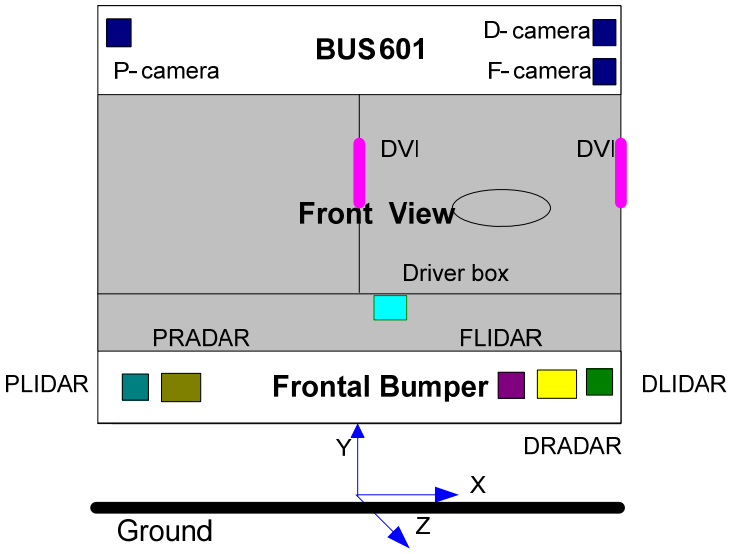


Figure 10 Layout of sensors, cameras and HMI

For convenience, the following abbreviations are used:

F-	Frontal-looking or frontal
D-	Driver-side-looking or driver-side
P-	Passenger-side-looking or passenger-side
I-	Interior-looking
LIDAR	Laser scanning RADAR
RADAR	Micro-wave RADAR
CAM	Camera

For example, F-CAM represents "frontal-looking camera", and D-RADAR stands for "driver-side micro-wave RADAR".

The numbers that are given in the following table are the obstacle-sensor positions of FCWS on the Samtrans bus.

Sensor/Parameter		**Host bus/Value**
Description	**Parameter**	**Bus 601**
F-LIDAR	X (lateral, mm)	768
	Y(vertical to ground, mm)	445
	Z(longitudinal to frontal face of the bumper, mm)	-25
	Angle (°)	0
D-LIDAR	X (lateral, mm)	1150
	Y(vertical to ground, mm)	435
	Z(longitudinal to frontal face of the bumper, mm)	-38
	Angle (° to the left)	20
P-LIDAR	X (lateral, mm)	-1180
	Y(vertical to ground, mm)	445
	Z(longitudinal to frontal face of the bumper, mm)	-76
	Angle (° to the right)	20
D-RADAR	X (lateral, mm)	965
	Y(vertical to ground, mm)	445
	Z(longitudinal to frontal face of the bumper, mm)	-51
	Angle (°)	0
P-RADAR	X (lateral, mm)	-965
	Y(vertical to ground, mm)	440
	Z(longitudinal to frontal face of the bumper, mm)	-51
	Angle (°)	0

Table 2. Location and orientation of obstacle detection sensors

1.9.1.1 Cameras

Camera (Bus 603)	X(m)	Y(m)	Z(m)
P-CAM	-0.60	2.59	0.27
D-CAM	0.95	2.69	-0.26
F-CAM	0.93	2.70	-0.13

Table 3. Locations of cameras

1.9.1.2 LIDARs (DENSO Corporation)

The table below shows LIDAR specifications.

Detection range	0-120m
Detection angle	40deg (lateral, ±20deg)
Detection angle	4.4deg(elevation)
Update rate	100ms
Laser wave length	850nm
Laser beam size	0.2deg(lateral) 0.9deg(elevation)
Number of detection points	265(lateral),6 (elevation) total: 1590points/cycle

Table 4. LIDAR specifications

The power supply of LIDARs is controlled by a speed-controlled relay. Whenever the bus speed measured is below 3m/s and the creeping detector detects the bus is not moving, a LIDAR control signal is set inactive to turn off the power to the LIDARs. When the bus speed measured is greater than 3m/s or the creeping detector detects the bus is moving, the LIDAR control signal is active and the LIDAR power is resumed.

(This relay may be removed if the sensor manufacturer improves the design to make the sensor eye-safe.)

1.9.1.3RADARs (EVT-300)

The table below shows RADAR specifications.

Detection range	0.3-110m
Detection angle	12deg (lateral,±6deg)
Update rate	65ms

Table 5. RADAR specification

1.9.2 FCWS Host-bus sensors

Vehicle speed is measured by listening to the vehicle's SAE J1939/1708 data bus and also by tapping off of an analog speed signal directly from the transmission. This speed signal from the transmission is filtered and conditioned by an electronic circuit.

Vehicle yaw rate is measured using a fiber optic rate gyro. This unit is mounted in a waterproof enclosure under the floor near the rear axle. This transducer has an RS232 interface.

Brake pressure is measured using a pressure transducer mounted on a spare port of the air brake system under the floor of the driving area. A proximity sensor, which is used to determine if the bus is moving at speeds lower than 2-3 miles per hour, is mounted near a universal joint on the drive shaft. Turn signal activation and backing light status is recorded by tapping off the existing turn signal circuit and backing lights. Windshield wiper activation is determined with a proximity sensor mounted on the windshield wiper mechanism. The host-bus state signals, including brake pressure, turn signals and back up light status , windshield wiper signal, creeping detector status and sensitivity level are filtered before going to A/D converters.

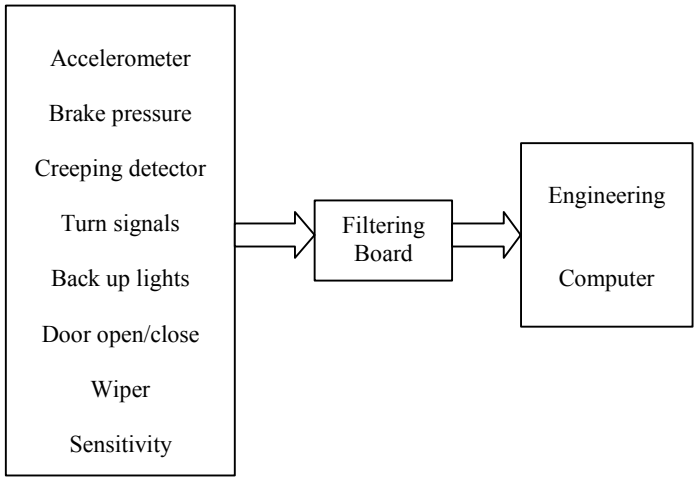


Figure 11. Interface between the engineering computer and host-bus sensors

The GPS antenna is mounted on the rear of roof near the exhaust for the HVAC, the GPS computer is mounted in a waterproof enclosure near the HVAC evaporator unit in the rear of the bus. The GPS and CDPD modem antenna are mounted on the rear of roof near the exhaust for the HVAC, the GPS and CDPD modem computers are mounted in a waterproof enclosure near the HVAC evaporator unit in the rear of the bus.

1.9.3 FCWS Battery / Ignition monitoring and shutdown circuitry

Two relays control the master power supplies: one is the master relay; the other is a time-delay relay. After ignition is on/off, the master relay turns on/off the switch. The switch will trigger the time-delay relay counter. Once the counter reaches a preset value, the time-delay relay will turn on/off the 12V and 24V bus Bars. The purpose of the time-delay is to avoid noise-triggered false on/off of power supplies, and give some additional time to the computers to save files before exit the program after the ignition is off. The master relay on/off signal is sent to the engineering computers to indicate the ignition operation.

1.10. SCWS Side Sensors

1.10.1 SCWS Laser Scanner

A laser ranging scanner manufactured by SICK, Inc is mounted on each side of the bus, behind the front wheel wells, below floor level. These laser scanners are use for object detection. This is a commercially available LIDAR that has been used extensively in the field of robotics and engineering for many years. A detailed error analysis of this sensor is included in the testing section of this document. The specifications for this LIDAR far exceed what is necessary for this application. It's usage for the research prototype system allowed the collection of high quality range data which can be used to show what is possible with collision warning systems. Any commercial collision warning system will not need to use as high performance LIDAR as we used. With the data collected it is easy to down sample and add noise to see how future algorithms perform.

Figure 12. SICK Laser Scanner and Retraction Assembly

1.10.2 Laser scanner retraction system

Each laser is mounted in a box approximately 18” H x 12” W x 12” D which mount in an opening in the sheet metal side panel of the bus. In operation, each laser extends approximately 4” beyond the side panel of the bus, but they are retracted below flush with the side panel when not in use. A small air cylinder on top of each enclosure box actuates the retract / extend motion of the laser, which swings on an arm pivoted near the front of the box. If the laser comes in contact with obstacles in the environment, the compliance of the air cylinder allows the laser to be pushed back into the enclosure to minimize damage to the laser and environment.

The actuation system comprises:
(2) Bimba air cylinders, 1-1/2” bore x 3” stroke;
(2) Automatic 4-way, spring-return, 24VDC, 8W solenoid valves;
(1) Watts filter-regulator;
(1) 2-way shutoff (ball) valve;
125PSI air supply with storage tank (On bus);
Plastic air tubing, ¼” OD and push-lock fittings.

With system air and/or electrical power off, the laser will be held retracted into the box by a return spring. To operate the actuation system, the ball valve (located at the bus air tank) will be opened allowing compressed air to pass through the filter-regulator (which will be adjusted to achieve appropriate force from the air cylinders) to the two solenoid valves located near the cylinders. Spring return on the solenoid valves will normally hold the cylinders retracted, even with zero air pressure. When the valves are activated based on a command signal from the control computer, air will flow through the valves to the cylinders, causing them to extend and push the laser support arms against a mechanical stops to precisely position the lasers for operation. When the valves are deactivated, air plus the return spring will cause the arms to swing back into the box against a back stop, such that the lasers will be within the bus envelope. Speed controls on the cylinders will be adjusted to give appropriate extend and retract speeds. Cylinder volume is 5.3 cubic inches (each); for the worst case of 125PSIG (max. available pressure; normal operating

pressure is 50 PSIG), each cylinder stroke will consume about 50SCI (standard cubic inches) of air, or about .029SCF (standard cubic feet). Under normal operation, we expect operation of the cylinders to occur not more than a few times per hour, so total air consumption should be small. The bus air tanks each hold about 1 cubic foot or air, so the volume consumed by actuation of two cylinders going full-stroke should be less than 1% of the tank volume. Leakage (through cylinder seals, etc.) is negligible.

1.11. SCWS Curb Detector

For the SCWS we need to determine the location of the sidewalk in order to better assess the situation. Pedestrians on the sidewalk are considered safer than if they are not on the sidewalk. We used a laser line striper (LLS) to detect the position of the curb at the front of the bus. The technical details of the LLS are presented in a paper.[1] Here we give an overview over its working principle and illustrate, how it was mounted on the bus.

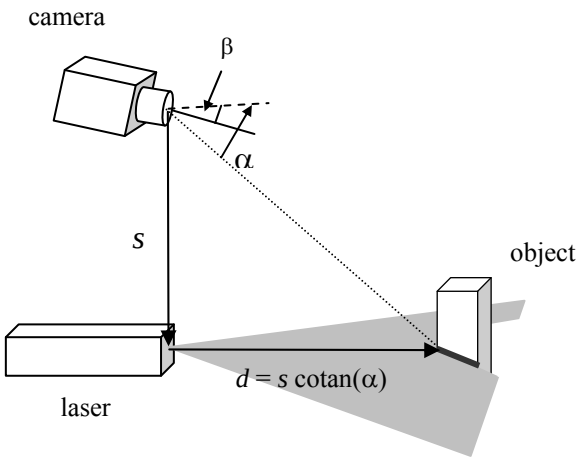


Figure 13 Schematic of a laser line striper.

The LLS projects a pattern of light into the scene that is imaged by a camera (see Figure 13) and the appearance of the pattern is used to compute distance to objects in the environment. The LLS is attractive because of its relative small size and robustness. In addition, computation of range is very low cost compared to other optical methods such as stereovision that requires high computation.

[1] Mertz, Kozar, Miller, Thorpe. "Eye-safe Laser Line Striper for Outside Use." IV 2002, Proceedings of the IEEE Intelligent Vehicle Symposium (IV2002). June 2002. http://www.ri.cmu.edu/pubs/pub_3890.html

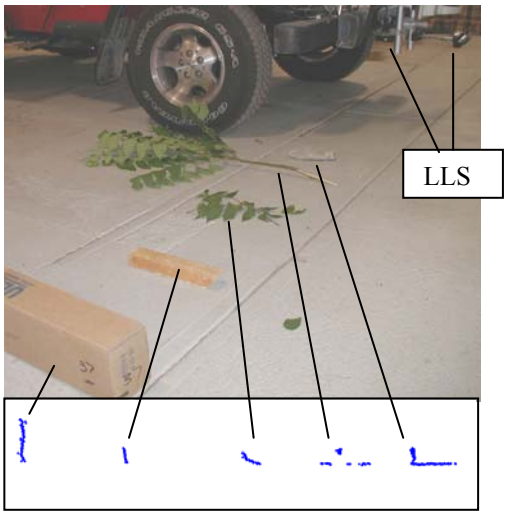


Figure 14 A LLS mounted on a vehicle looking to the side at various objects. The return from the sensor is shown in the lower part of the figure.

We have built and employed such a sensor where the light pattern is a plane of NIR light and the appearance on the object is a line (see Figure 14). The novelty of our sensor is that it can work outside in bright sunlight even though the power output of the laser is limited by eye safety. The background from ambient sunlight is suppressed by synchronizing a pulsed laser with a fast shutter, employing a narrowband filter, and some image analysis. The figure shows our LLS mounted on a vehicle and looking at various objects on the side of the vehicle. In the lower part of the figure, one can see the output of the sensor.

The range and resolution are dependent on the sensor configuration. In the following table they are shown for three different field-of-views:

field of view [deg]	30	55	105
angular resolution [deg]	0.05	0.09	0.16
max. range (ideal) [cm]	700	520	300
max. range (typical) [cm]	300	200	130
range resolution [cm]	1.4	2.6	5.0

Table 6. Range and Resolution of Laser Line Striper

The maximum range is for ideal conditions (high reflectivity objects, etc.). For typical conditions, it is about half that distance. The range resolution is for a 2 m distance, resolution varies with the square of the distance.

The LLS was mounted inside the front bumper of the bus with a field-of-view perpendicular to the side of the bus (see Figure 15).

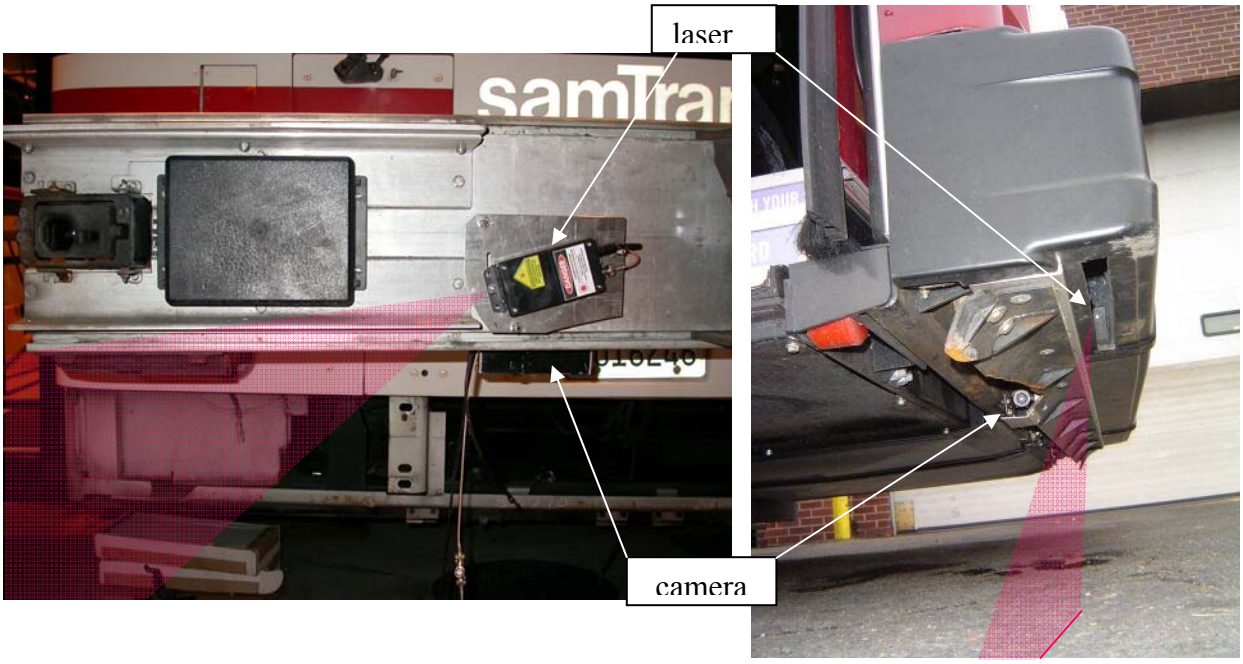


Figure 15 LLS mounted in the front bumper of the bus. On the left side is a frontal view of the SamTrans bus with the rubber skin of the front bumper removed. The laser can be clearly seen, the camera is occluded by the holding bracket. On the right is a side view of the PAT bus. The red semitransparent area indicates the location of the laser plane.

The LLS returned the cross section profile of the environment besides the bus. If there is a curb besides the bus, the profile looked like the one shown in Figure 16.

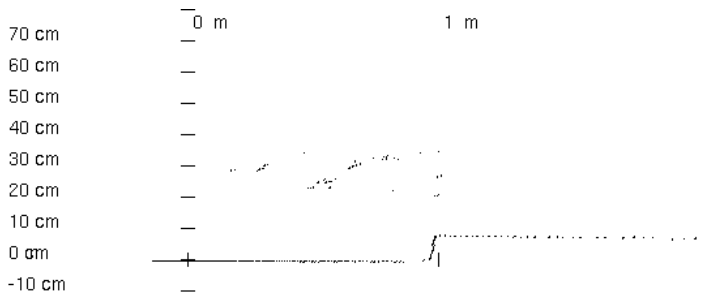


Figure 16 Profile of the road and curb observed by the laser line striper. Some erroneous readings can be seen above the road.

Finally, the location of the curb was extracted from the profile by a histogram method (see paper [2] for details).

During the operation of the bus with the LLS we had following experiences with environmental conditions:

Temperature: The temperature range for operation in Pittsburgh was between 0^o F and 120^o F. The upper range is 20^o F above the ambient temperature. This temperature was added because the laser is located within the black enclosure of the front bumper and above the black pavement. We needed to add a heater to the laser in order to reduce this range of temperatures.

Water: The LLS can be exposed to water through rain or the bus wash. We had to do some extra waterproofing to the camera.

Mechanical: The camera and laser needed to be tightly screwed to the frame of the bumper to keep its alignment. No mechanical damage occurred during the time of operation.

Dirt: Only a minimal amount of dirt accumulated on the lens of the camera or the exit window of the LLS, it did not affect its operation.

[2] Aufrère, Mertz, and Thorpe. "Multiple Sensor Fusion for Detecting Location of Curbs, Walls, and Barriers," Proceedings of the IEEE Intelligent Vehicles Symposium (IV2003). June 2003.

1.11.1 Sensor fusion of curb detector, bus state and video

In the previous section we described how we detect the curb next to the front bumper with a laser line striper. In this section we discuss how we determine the location of the curb alongside the bus and in front of the bus. Details of this method can be found in a technical paper.[3] The section below is a summary.

1.11.1.1 Tracking the curb alongside the bus

The movement of the bus is recorded and therefore it is possible to transform past curb measurements into the current reference frame. The collection of these past curb measurements in the current reference frame gives us the curb alongside the bus. An example can be seen in Figure 17, where the curb alongside the bus is indicated as a blue line.

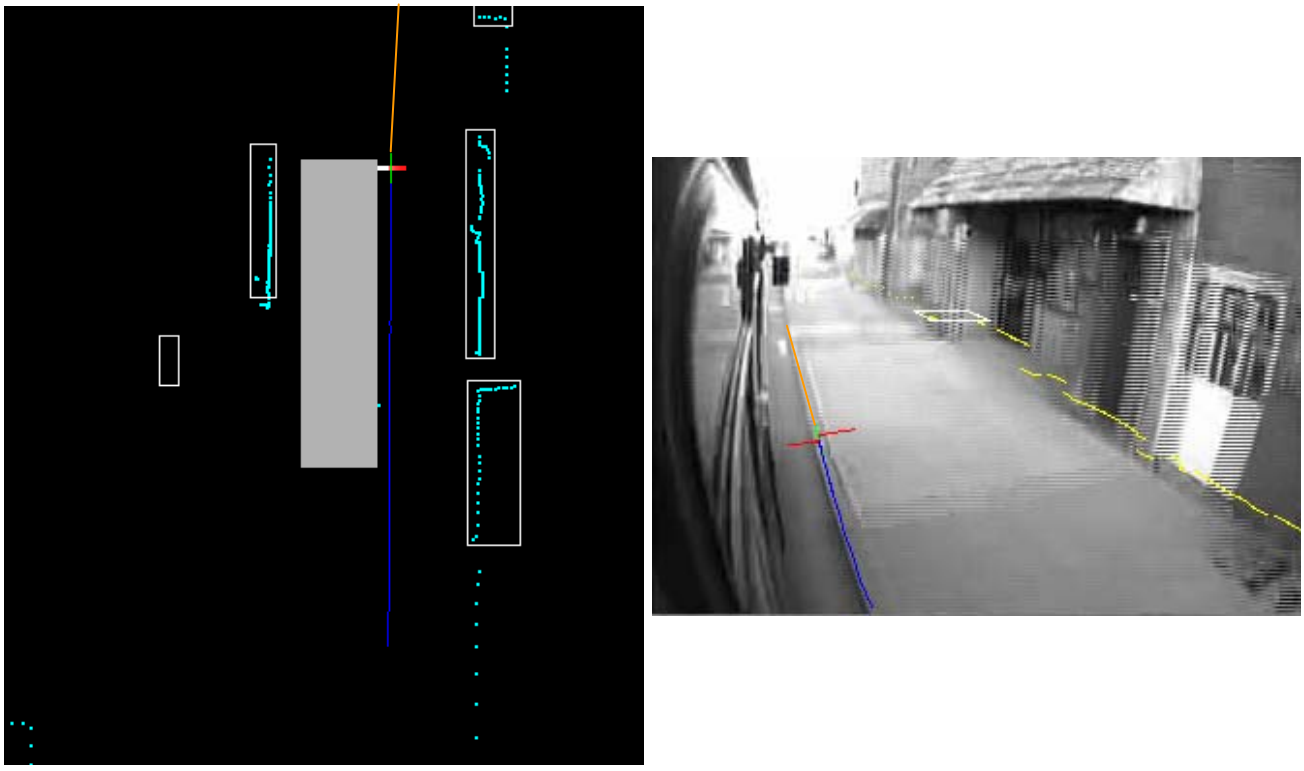


Figure 17 Curb alongside the bus. On left is a birds-eye-view and on the right is a view from the right camera. The raw data from the LLS is indicated with red color, the curb position extracted from that raw data is in green and the tracked curb is in blue.

[3] op. cit.

1.11.1.2 The curb in front of the bus

To detect the curb in front of the bus we use a vision process which is initialized by the knowledge of the curb position we already have. The right image in Figure 17 shows the view of the right, forward looking camera. We already know where the curb is and in which direction it proceeds. This is used for the initial start and direction of a search for an edge in the image. This search is continued till we reach a preset limit in the image, reach the edge of the bus, or reach an object. The position of the object is known from the laser scanner data. An orange line in Figure 17 indicates the curb ahead of the bus in our example.

1.11.1.3 Calibration of sensors

The above mentioned method of determining the position of the curb in front of the vehicle requires careful calibration of four sensors: Bus state, curb detector (LLS), camera, and laser scanner.

The bus is the reference frame and therefore the location and orientation of the other three sensors need to be determined with respect to the bus. The locations of the sensors are all measured using measuring tape. The laser scanner and the LLS are mounted on the bus in such a way that the orientation of their internal reference frames is either parallel or perpendicular to the axis of the bus reference frame. This way the rotation from one to the other coordinate system is easy to be determined.

Only the orientation of the camera is not trivial. First we want to define the rotation matrix. We use roll angle φ, pitch angle θ, and yaw angle ψ.

Equation 1

$$R(\varphi) = \begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Equation 2

$$P(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Equation 3

$$Y(\psi) = \begin{pmatrix} \cos(\psi) & 0 & -\sin(\psi) \\ 0 & 1 & 0 \\ \sin(\psi) & 0 & \cos(\psi) \end{pmatrix}$$

Equation 4

$$C = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

The matrix which rotates the camera coordinates to the vehicle coordinates is:

Equation 5

$$A = C \cdot R \cdot P \cdot Y$$

The matrix C takes care of the different conventions for the orientation of the axis. For the vehicle the x-axis is forward, the y-axis points to the right, and the z-axis points down. For the camera the z-axis is in the forward direction, the x-axis points to the right, and the y-axis points down.

The goal is now to find the three angles which describe the orientation of the camera. For this one can make use of the image provided by the camera. The distance between two pixels in the image correspond to about 1/10 of a degree in the real world and therefore one can measure angles quite accurately with the help of the image. Each point in the image has one horizontal and one vertical angle. Since we need to determine three angles (roll, pitch, and yaw), we need at least two points in the image and their corresponding points in the real world. To simplify the problem we choose three points with following properties:

1. The first point has the same y-position (in bus coordinates) as the camera.

2. The second and third points are vertical to each other in the bus coordinate frame.

Figure 18 Example of three points chosen for the calibration. Point *a* is on the ground at the front wheel at the same y-position as the camera. Points *b* and *c* are edges at the lower and upper part of the open front door.

If the three angles are small, the solution is to a good approximation:

Roll = difference in angle between the vertical and the line *b*-*c*.

Yaw = horizontal angle of *a*.

Pitch = vertical angle of *a* after correction of fact that *a* is not at the same height as the camera.

We found out that this approximation is not always good enough and we worked out the exact solution.

The three points are expressed in homogenized coordinates:

Equation 6

$$a = \begin{pmatrix} a_1 \\ a_2 \\ 1 \end{pmatrix},\ b = \begin{pmatrix} b_1 \\ b_2 \\ 1 \end{pmatrix},\ c = \begin{pmatrix} c_1 \\ c_2 \\ 1 \end{pmatrix}$$

All vectors in the following section are homogenized, *i.e.* they are always divided by their own third component so that the third component becomes 1.
The approximate yaw angle is:

Equation 7 $$\psi_a = \tan^{-1}(a_1)$$

To get the approximate pitch angle, we need to do following calculation:

Equation 8 $$a_y = Y(\psi_a) \cdot a$$

Equation 9 $$\theta_a = \tan^{-1}(a_{y2}) + \tan^{-1}(\Delta z / \Delta x)$$

where Δz and Δx are the distances in the z and x directions (bus coordinates) between the camera and point a.

Now we can construct following three points by rotating the points a, b, and c:

Equation 10 $$a_{yp} = P(\theta_a) \cdot Y(\psi_a) \cdot a$$

Equation 11 $$b_{yp} = P(\theta_a) \cdot Y(\psi_a) \cdot b$$

Equation 12 $$c_{yp} = P(\theta_a) \cdot Y(\psi_a) \cdot c$$

Now we need to find a rotation which will make the line b_{yp}-c_{yp} vertical while leaving a_{yp} unchanged, *i.e.* rotate around a_{yp} Therefore we need to solve following equation for the rotation angle φ_a:

Equation 13 $$(S(\varphi_a) \cdot b_{yp})_1 - (S(\varphi_a) \cdot c_{yp})_1 = 0$$

where the index 1 means the first component of the vector (which of course has been homogenized). The rotation S is defined as

Equation 14 $$S(\varphi_a) = P^{-1}(\alpha) \cdot R(\varphi_a) \cdot P(\alpha)$$

Equation 15 $$\alpha = \tan^{-1}(a_{yp2})$$

Equation 13 is the condition that the two points become vertical, Equation 14 is the rotation around the point a_{yp}.

Putting all the numbers into the equations and doing the multiplications is tedious but straightforward until one reaches an equation of the form:

Equation 16
$$\cos(\varphi_a)\cdot x - \sin(\varphi_a)\cdot y = z_1$$

The solution to this equation is:

Equation 17
$$\varphi_a = \pm\cos^{-1}\left(\frac{x\cdot z_1 + y\cdot z_2}{\sqrt{(x^2+y^2)(z_1{}^2+z_2{}^2)}}\right)$$

With

Equation 18
$$z_2 = -\sqrt{x^2+y^2-z_1{}^2}$$

The angle φ_a is negative if

Equation 19
$$\cos(\varphi_a)\cdot x - \sin(\varphi_a)\cdot y - z_1 > 0$$

Now one can construct the full rotation matrix:

Equation 20
$$A = C\cdot P^{-1}(\alpha)\cdot R(\varphi_a)\cdot P(\alpha)\cdot P(\theta_a)\cdot Y(\psi_a)$$

Notice that this equation does not contain the usual roll angle φ, pitch angle θ, and yaw angle ψ. If desired, one can determine them in the following way:

Equation 21
$$\varphi = \tan_2{}^{-1}(a_{32}, a_{33})$$

$$\theta = -\sin^{-1}(a_{31})$$
$$\psi = \tan_2{}^{-1}(a_{21}, a_{11})$$

where a_{ij} are the components of the matrix A and $\tan_2{}^{-1}(x,y)$ is the inverse tangent which takes appropriate care of the sign with respect to the four quadrants.

The solution has been implemented in a Matlab™ program. This Matlab™ program includes an interface which lets you choose the points by clicking on the image. It does the calibration on full images as shown in Figure 18 or on quad-images. The calibration is also used to do image overlays as one can see in Figure 17 on the right.

1.12. PC-104 Platforms

1.12.1 SCWS PC-104 platforms

All computers contain processor boards from various manufacturers, all based on the Intel Pentium III with speeds ranging from 700 MHz to 1.26 GHz. This part selection was motivated by the need for the most powerful processor available at the time in a rugged PC/104 form factor that is capable of withstanding extreme temperatures.

All computers run Red Hat Linux 7.2 with a 2.4.18 kernel and patched to reduce kernel latencies. This ensures that a heavily loaded computer is more responsive and allows us to use Linux for this data driven very soft real-time task. The benefit of this, as opposed to a real time system, is in ease of use.

The left and right computers are almost identical, physically. Minor hardware and software changes are all that is required to interchange the two.

PC/104 Stack	Left	Right
CPU 2 Serial Ports 100 Mbps Ethernet 40 GB Notebook Hard Disk	J1708 Interface	IMU
Frame Grabber	Forward Looking Curb Detector	Striper/Curb Detector
High Speed Serial Ports (4)	Left SICK Data, FCWS Interface, DINEX	Right SICK Data, Odometry
Sound Card	(N/A)	Driver Vehicle Interface
Digital I/O	Left SICK Power and Retraction, Left DVI	Right SICK Power and Retraction, Striper Power, Right DVI
Power Supply		

Table 7. Configuration of Left and Right SCWS Computers

Measurement of CPU loading on each of the three computers indicates that our 600MHz left computer is 65% loaded, our 1.2GHz right computer is 45% loaded, and our 700MHz data logging computer (Digital Video Recorder) is 60% loaded.

1.12.2 FCWS PC-104 platforms

The FCWS system hardware is composed of sensors, an engineering data recording-processing computer and a video-recording computer, as illustrated in the figure below. The engineering data that sensors send out is recorded and processed by a PC104 computer system. Besides regular ports for a PC104 computer, it has a digital I/O card, an Analog/Digital I/O card, a CAN card which reads J data bus, a Serial Port card and a counter/timer card.

A sensor arrangement for FCWS is designed to include sensors to detect frontal and frontal corner obstacles and to monitor steering angle movement, brake pressure, throttle position, vehicle velocity and acceleration. Video data from the cameras is recorded using another PC104 computer.

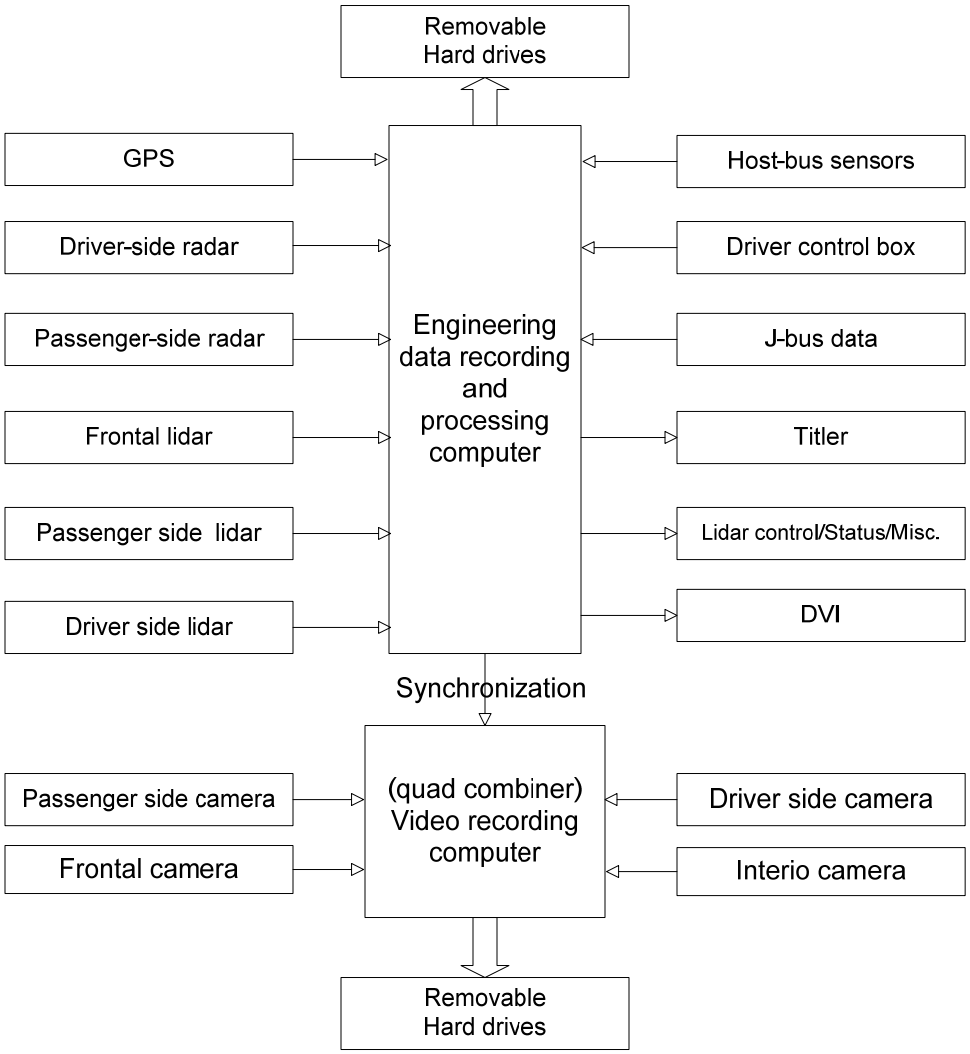


Figure 19. FCWS System Architecture

The figure below shows the layout of the computer enclosure. The computer enclosure contains an engineering computer, a video recorder, electronics circuits including battery ignition monitoring and shutdown circuitry, power adapters, bus power bars and cable connectors.

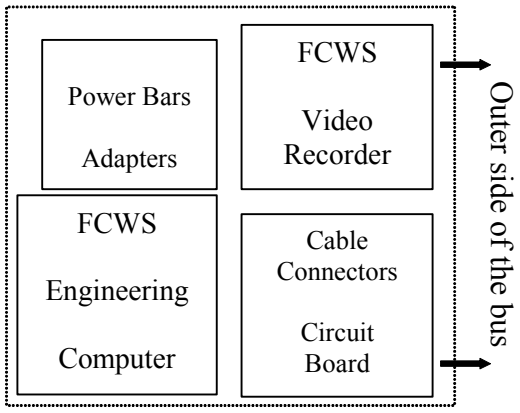


Figure 20. FCWS Computer enclosure on the bus (top view)

1.13. Digital Video Recorder PC-104 Platforms

1.13.1 SCWS Digital Video Recorder PC-104 platform

The digital video recorder / data repository computer also serves as an Internet gateway via a cellular telephone modem. This provides remote system monitoring; something we have found quite useful when managing such a complex system in the field.

PC/104 Stack	
PCMCIA Interface	PCMCIA Cellular Modem Adapter
MPEG-1 Hardware Encoder	External Video Cameras
CPU 2 Serial Ports 100 Mbps Ethernet Removable Disk Drive	GPS for Location Tagged Data
Power Supply	

Table 8. Configuration of SCWS Digital Video Recorder

Initially, the removable hard drive was a 250 GB desktop model, the largest drive available. It was hoped that although the environmental vibration and shock would well exceed the manufacturer's specifications, the drive would still function most of the time. This has proven to not be the case. For this reason, the two 80 GB notebook drives

provide enough data storage for about 2 weeks of use, as opposed to the 3+ weeks with the larger drive.

1.13.2 FCWS Digital Video Recorder PC-104 platforms

The cameras capture the front road scene, the left and right front corner road scene, and the passenger compartment of the bus. The video streams from the four cameras are combined into one video stream by a quad image combiner to extend the hard drive storage capacity. The video-in port of video recording computer is connected to the video-out port of the quad combiner by a $75\,\Omega$ video cable.

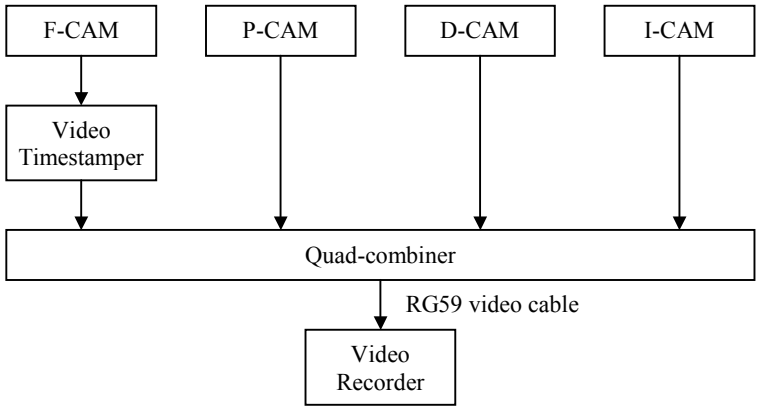


Figure 21. FCWS Video recorder-camera interface

The video recording system is a standalone PC/104 system with a video board. It reads commands from the engineering computer and records the MPEG video clips to a removable hard drive. The video is recorded at 1Mbps, which is about 450MBytes per hour. The specifications of the board are as follows:

General specifications	
Capture rate	30 frames/sec (NTSC, RS-170, CCIR) 25 frames/sec (PAL)
A/D resolution	8-bits for luminance 8-bits for chrominance
Output resolution	768 x 576 (PAL) 640 x 480 (NTSC, RS-170)
Video inputs	4 multiplexed input channels total: 2 S-video or 4 composite.
Video output	PAL or NTSC from a BNC connector
Output data	MPEG2 streaming data at rates of 100 kbits to 10 Mbits/second
Bus structure	PC/104
Board size	3.80” x 3.55”
Input power	5 volts at 280 mA
Number of cards per system	2
Supporting operating systems	Windows, Linux, QNX6

Table 9. Video board specifications

The video board supports variable bit rates (number of bits of the stored video data per second).

1.13.3 SCWS timing synchronization

Internal to the SCWS, NTP (Network Time Protocol) is used to synchronize the clocks on our three computers over the Ethernet network. The right computer is considered the master clock, independent even of the more accurate GPS clock, which is however slow to converge and unreliable as the bus moves. Upon system boot, the left computer and the

data storage computer resynchronize their clocks to the master clock to correct for temperature induced clock drift, which is especially noticeable when the computers have been exposed to extreme temperatures. Thereafter, the NTP daemon on the slave computer uses the network to statistically sample the master clock so that it can determine the error on the local clock. It corrects this error by effectively speeding up or slowing down the local clock to close the difference. The clock is always monotonically increasing, and without steps in time.

1.13.4 FCWS timing synchronization

The following serial ports on the engineering computer are used for synchronization between the FCWS engineering computer and the video recorder:

Port 1: Sensor-Video Computer Communications (115200 Baud)

Port 8: (RS-232) Video timestamper (9600 baud)

The video files and the sensor file need to be synchronized to describe the same scenario. The video recorder reads commands from the engineering computer and records the MPEG video clips to a removable hard drive. The commands from the engineering computer are “begin recording (with a time stamp),” and “stop recording”. Every time the video recorder gets a “begin record” command it closes the old video file, opens up a new file (named by the time stamp) and starts recording.

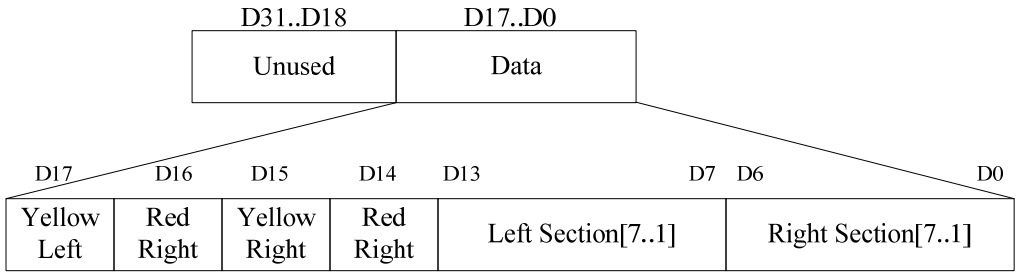


Figure 22 FCWS Warning signal definition

1.13.5 FCWS / SCWS data synchronization

Data between the FCWS and the SCWS is a serial port on the SCWS left computer and the FCWS at 115 Kbaud. Data that is exchanged by the computers is sent with no time tag, but is saved by the receiver with the receiver's time tag

1.13.6 FCWS / SCWS data protocol

Each message consists of a Header, ID, Length, Data, Checksum values. The Header is a four character sequence (HEADER0 … HEADER3). The ID is a message identification byte. Messages from the FCWS computer to the SCWS computer will have odd ID numbers. Messages from the SCWS computer to the FCWS computer will have even ID numbers. This is a two byte value. The length is the total number of data bytes. The length does not include itself, the header, ID, or checksum. This is a 2-byte sequence. Data is an array denoting the length of bytes. The checksum is the last byte per message. The checksum is a two's complement of the sum of all the prior bytes in the message, including the header, ID, length, and data. The two's complement is used so that if all of the bytes of the message (including checksum) are summed by the receiver, the result is zero for a valid message. Specific values and parameters are shown below:

HEADER0	0x99
HEADER1	0x44
HEADER2	0x22
HEADER3	0x66
PATH_TO_CMU_ID	1
CMU_TO_PATH_ID	2
Bytes for status flags	
FRONT_DOOR_OPEN	0x01
REAR_DOOR_OPEN	0x02
RIGHT_TURN_SIGNAL_ON	0x04
LEFT_TURN_SIGNAL_ON	0x08
HAZARD_LIGHTS_ON	0x10
POWER_DOWN	0x20
OVERRIDE_ON	0x40
IN_REVERSE	0x80

UNKNOWN_POSITION_CM	-10000
UNKNOWN_POSITION_M	(UNKNOWN_POSITION_CM/10.0)
Bit numbers for warning message from SCWS to FCWS	
RIGHT_FRONT_LOW_ALERT	0
RIGHT_FRONT_LOW_WARN	1
RIGHT_FRONT_MEDIUM_ALERT	2
RIGHT_FRONT_MEDIUM_WARN	3
RIGHT_FRONT_HIGH_ALERT	4
RIGHT_FRONT_HIGH_WARN	5
RIGHT_REAR_LOW_ALERT	6
RIGHT_REAR_LOW_WARN	7
RIGHT_REAR_MEDIUM_ALERT	8
RIGHT_REAR_MEDIUM_WARN	9
RIGHT_REAR_HIGH_ALERT	10
RIGHT_REAR_HIGH_WARN	11
LEFT_FRONT_LOW_ALERT	12
LEFT_FRONT_LOW_WARN	13
LEFT_FRONT_MEDIUM_ALERT	14
LEFT_FRONT_MEDIUM_WARN	15
LEFT_FRONT_HIGH_ALERT	16
LEFT_FRONT_HIGH_WARN	17
LEFT_REAR_LOW_ALERT	18
LEFT_REAR_LOW_WARN	19
LEFT_REAR_MEDIUM_ALERT	20
LEFT_REAR_MEDIUM_WARN	21
LEFT_REAR_HIGH_ALERT	22
LEFT_REAR_HIGH_WARN	23
RIGHT_NOTIFY	24
LEFT_NOTIFY	25
RIGHT_UNDER_WHEEL	26
LEFT_UNDER_WHEEL	27
LOW_SETTING	28
MEDIUM_SETTING	29
HIGH_SETTING	30

Table 10. Parameter Values

1.13.6.1 Data sent from the FCWS to the SCWS

timestamp_secs	Number of seconds since 1/1/1970
timestamp_usecs	Additional microseconds
Warning_msgs	Warning field
Forward object of interest, z=-10000, x=-10000 means no object	
front_obj_x	Longitudinal position of object (= x in SCWS)
front_obj_y	Lateral position of object (= -y in SCWS)
front_obj_heading	Orientation of object velocity vector (= -heading in SCWS)
front_obj_speed	Left object speed along heading direction
sound_index	Index of sound in sound directory, -1 for none
sound_bearing	Left to right bearing of sound, percentage
curb_loc_2x	Longitudinal curb position (= z in FCWS)
Status	(Note: REAR_DOOR_OPEN, HAZARD_LIGHTS_ON, and OVERRIDE_ON are not produced by the FCWS)

Table 11. Data sent from the FCWS to the SCWS

1.13.6.2 Data sent from the SCWS to the FCWS

timestamp_secs	Number of seconds since 1/1/1970
timestamp_usecs	Additional microseconds
warning_msgs	Warning field
brake_pressure	Brake pressure - currently always 0
Latitude	GPS latitude
Longitude	GPS longitude
Altitude	GPS altitude
Speed	Speed from vehicle state estimation in km/hour
Left object of interest, x=-10000, y=-10000 means no object	
left_obj_x	Longitudinal position of object (= z in FCWS)
left_obj_y	Lateral position of object (= -x in FCWS)
left_obj_heading	Orientation of object velocity vector (= -heading in FCWS)
left_obj_speed	Left object speed along heading direction
Right object of interest, x=-10000, y=-10000 means no object	
right_obj_x	Longitudinal position of object (= z in FCWS)
right_obj_y	Lateral position of object (= -x in FCWS)
right_obj_heading	Orientation of object velocity vector (= -heading in FCWS)
right_obj_speed	Right object speed along heading direction
Tracked and predicted curb locations, ordered in increasing x (longitudinal)	
curb_loc_1x	Longitudinal curb position (= z in FCWS)
curb_loc_1y	Lateral curb position (= -x in FCWS)
curb_loc_2x	Longitudinal curb position (= z in FCWS)
curb_loc_2y	Lateral curb position (= -x in FCWS)
curb_loc_3x	Longitudinal curb position (= z in FCWS)
curb_loc_3y	Lateral curb position (= -x in FCWS)
curb_loc_4x	Longitudinal curb position (= z in FCWS)
curb_loc_4y	Lateral curb position (= -x in FCWS)
curb_loc_5x	Longitudinal curb position (= z in FCWS)
curb_loc_5y	Lateral curb position (= -x in FCWS)
curr_curb_loc_x	Current Longitudinal curb position (= z in FCWS)
curr_curb_loc_y	Current Lateral curb position (= -x in FCWS)
sound_index	Index of sound in sound directory, -1 for none
sound_bearing	Left to right bearing of sound, percentage
Status	(Note: IN_REVERSE is not produced by the SCWS)

Table 12. Data sent from the SCWS to the FCWS

SYSTEM SOFTWARE

1.14. SCWS Software Architecture Development

The SCWS uses software architectural and communications tools that were originally developed to support the ongoing robotics research of the Navlab project.[4] The architectural tools allow algorithm developers to view the rest of the system through a set of abstract, reconfigurable interfaces. In the initial development and ongoing debugging of an algorithm or in the post development analysis of data, the interfaces can be configured to read data from time tagged files using a common set of data access tools. As the algorithm matures, the interfaces can be reconfigured to use a common set of inter-process communications tools which integrate the individual algorithm into the larger system running in the field.[5]

1.14.1 Inter-process communications

The vast majority of inter-process communications in the SCWS can be considered as analogous to signals in electronics. These are repeated estimations of a consistently changing value, such as the most recent line scanner data or most recent set of tracked objects in the environment. It doesn't truly matter if the recipient misses a signal value: all that matters is the most recent value. What does matter is the minimization of latencies in transporting the signal value from producers to consumers. An appropriate paradigm for propagation of signal type information is global shared memory: A producer sets the memory and a consumer simply reads the most recent value. The Neutral Messaging Library (NML) from the Real-Time Control System (RCS)[6] library produced by NIST demonstrates this control-centric method for integrating robotic systems. We have chosen a simpler implementation than NML for global shared memory which uses a "single-writer, multiple-reader" model. When processes are communicating on the same

[4] Thorpe, Charles E. Vision and Navigation: The Carnegie Mellon Navlab. Kluwer Academic Publishers, 1990.

[5] Gowdy, Jay. Emergent Architectures: A Case Study for Outdoor Mobile Robots. Thesis for PhD at the Robotics Institute, Carnegie Mellon, CMU-RI-TR-00-27. November 2000.

[6] Gazi, Moore, Passino, Shackleford, Proctor and Albus. The RCS Handbook: Tools for Real-Time Control Systems Software Development. New York: John Wiley & Sons, 2001.

machine we use actual System V shared memory, whereas when processes are communicating between machines we transparently propagate changing memory values from writer to readers via the UDP socket protocol managed by shared memory managers running on each machine.

One of the reasons we chose to implement a simple shared memory communications scheme rather than adopting NML was that while signals make up the bulk of the communications, signals are not the only paradigm for inter-process communications in a robotic system. Symbols, i.e., atomic pieces of information, changes in state, or requests for information, are very difficult to communicate via a signal-based communications paradigm. For example, unlike signals, if a symbol value is dropped or missed, then information is lost, state changes don't get noticed, and requests are ignored. The guarantee that a symbol has been transported from writer to reader is worth significant additional latency and complexity in the implementation. Symbolic information is typically communicated in robotic systems via TCP/IP message based packages ranging in complexity from the raw use of socket libraries all the way up to complex, object based systems such as the Common Object Request Broker Architecture (CORBA).[7] In order to limit the complexity and size of our software while still providing some abstraction and flexibility, we have chosen a simple TCP/IP based messaging package developed for the Navlab project: The Inter-Process Toolkit (IPT).[8]

A key abstraction built in the SCWS using the messaging toolkit is the concept of a central black board. Individual algorithms mainly query the black board for their configuration parameters, but they can also post information in the black board and watch for changes in values on the black board. Thus, the black board becomes a channel for propagating information through the system that has to be generally available, but for which a certain degree of latency is acceptable. For example, when a driver sets the sensitivity switch to different levels, this causes a change to be posted to the warning

[7] The Object Management Group. The Common Object Request Broker: Architecture and Specification. Massachusetts: 1996.

[8] Gowdy, Jay. {IPT}: An Object Oriented Toolkit for Interprocess Communication. Technical Report for the Robotics Institute, Carnegie Mellon University, CMU-RI-TR-96-07. March 1996.

levels stored in the central blackboard. These changes are then propagated to the warning algorithm automatically and transparently. In addition, since much of the system's high level information is being funneled through the blackboard, we have chosen to make the black board manager the system process manager. It initiates, parameterizes, and monitors the system processes. Interestingly, this paradigm of a central black board was one of the earliest used in robotics [9], but it has been often rejected because if the black board is the only means for propagating information through the system, it becomes an intolerable bottleneck for the kind of low-latency, high-bandwidth signal-type information that forms the backbone of information flow for a real robotic system.

Thus we see the core of our communications philosophy: Instead of having one tool or one approach, which must be bent and stretched to handle all possible uses, we select a suite of simple tools, each one narrowly focused on a particular style of communications necessary for the efficient and successful operation of the system.

1.14.2 Vehicle state propagation

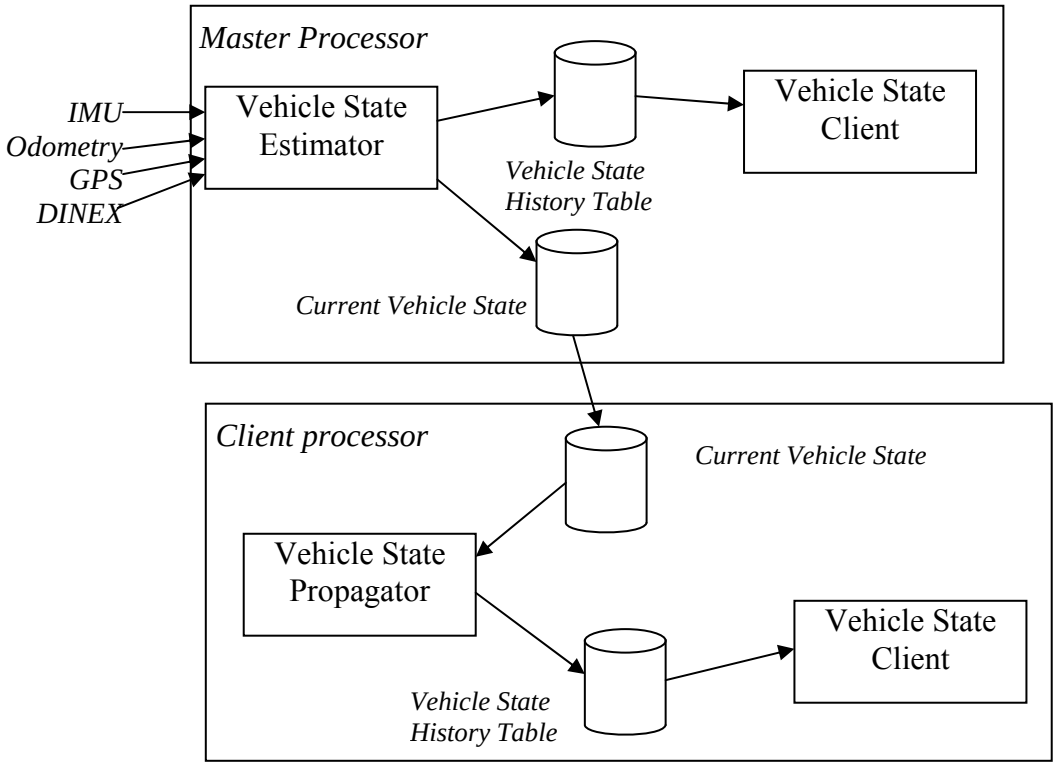
A fundamental question for most mobile robots is, "where am I?" For almost every module in the SCWS this question needs to be answered before going on to "what am I seeing?" Thus, a fundamental part of the SCWS architecture is the ubiquitous availability of vehicle state, i.e., the estimate of where the vehicle is, where it is pointing, and where it is going.

In past Navlab systems, the question of "where am I?" was assumed to mean "where am I right now?". Thus, perception algorithms would ask "where am I right now?", get the answer from the pose estimation system, and then apply that to the latest sensor information. This works fine when a robot is moving slowly, at a few meters per second, but when a robot is moving fast, at 10, 20, or even 30m/s, small discrepancies in time between the latest sensor pose estimation and the latest sensor information can lead to significant errors in placing that sensor information in the world, and thus to significant errors in operation.

[9] Hayes-Roth, B. A blackboard architecture for control. Artificial Intelligence, Volume 26. 1985: 251-321.

The goal of the current Navlab pose estimation system used in the SCWS is to allow perception algorithms on any machine in the system ask "where was I at time T?", where T is the time stamp of some relevant sensor event.

The pose propagation architecture we use is shown below. On one machine there is a pose estimation system connected to all the various sensors which is repeatedly answering the question "where am I?" Each pose estimate is put into a ring buffer in shared memory that any process on that machine can access. The user pose estimation routines take a time tag, and attempt to interpolate (or extrapolate a small amount) in this pose history buffer to come up with the best estimate of the vehicle pose at the requested time. When a new pose estimate is created, in addition to being entered in the local pose history table, it is sent via the shared memory managers to every machine in the system. On each of these client machines there is a process waiting for incoming pose estimates and using them to build a pose history buffer which can be used by other processes running on that machine to precisely match up pose estimations with sensor time tags.



Of course, a potential weakness of this system is that the clocks on all the machines must be precisely synchronized. Although we experimented with using hardware solutions using the IRIG-B protocol to allow us to have time estimates synchronized to within microseconds across machines, we found that the freely available package NTP [10] could synchronize our clocks to within a millisecond across the machines, even in the face of the harsh, changing environmental conditions encountered by a system running over long periods of time on a transit bus. Millisecond synchrony is more than sufficient for successful integration of vehicle pose and sensor information even at high vehicle speeds.

1.14.3 Data flow

Apart from the ubiquitous connections to the blackboard and vehicle state propagation system, the data flow for the vast majority of communications within the SCWS is fairly simple. The system has a left side processor which contains most of the processes for producing warnings on the left side of the bus, a right side processor for producing

[10] Mills, David L. Internet time synchronization: The Network Time Protocol. Published as: The Network Working Group Request for Comments: 1129. October 1989: 1-29.

warnings on the right side of the bus, and a central processor responsible for managing the system, saving data from the left and ride side processors, and saving a video record of the bus operation.

1.14.3.1 Left side data flow

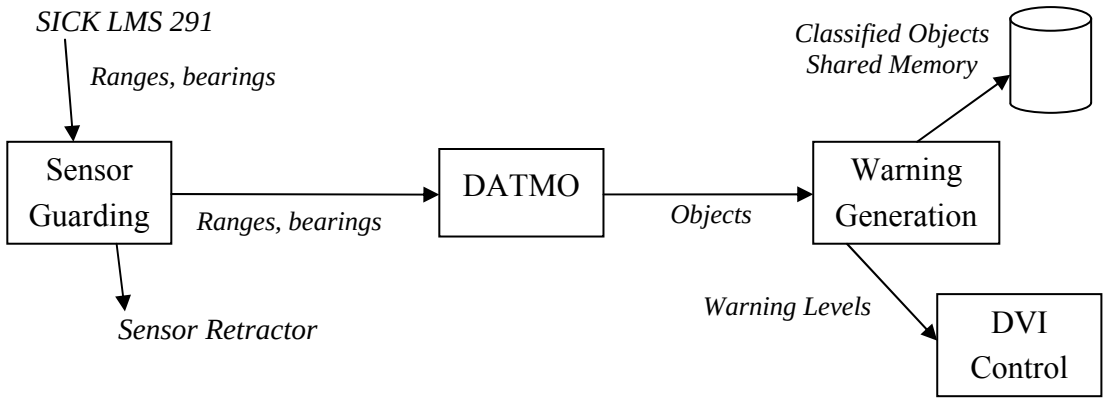
On the left side of the bus, warnings are generated based only on the laser range data.

- The data from the SICK laser range finder is read in by a reflexive "guarding" module. This module monitors the returns from the laser range finder and the velocity of the bus to do a quick determination if the sensor will hit anything. If the algorithm sees an imminent collision, it sends the signal which retracts the sensor and flags the range data as "bad". The algorithm continues to monitor the environment as best as it can from its retracted position, and when it determines there is enough room to extend safely, it does so. This algorithm should be considered analogous to a "flinching" reaction in a human which keeps the sensor (and those the sensor may hit) safe. No matter what happens, the guarding module publishes the laser data to the rest of the system via shared memory. The guarding module also examines the quality of the data coming from the laser. If it detects too many permanent blockages, usually due to mud or dried road salt, it will retract the sensor until the sensor is cleaned.

- The detection and tracking of moving objects (DATMO) algorithm reads in laser data and vehicle state data via shared memory and produces a list of moving and stationary objects around the vehicle.

- The warning algorithm takes the list of moving and stationary objects around the vehicle and combines the vehicle state (specifically the bus velocity and turning speed) to predict collisions. It produces an annotated list of objects with warning classifications and overall warning level.

- The left Driver Vehicle Interface (DVI) control module watches the overall warning level and controls the appropriate lights on the left side to warn the driver about objects around the vehicle. The left DVI control module also monitors the sensitivity and override switches on the DICB and changes values in the blackboard in response.



1.14.3.2 Right side data flow

On the right side of the bus, the system uses a curb detection and prediction system to augment the laser range data in generating warnings.

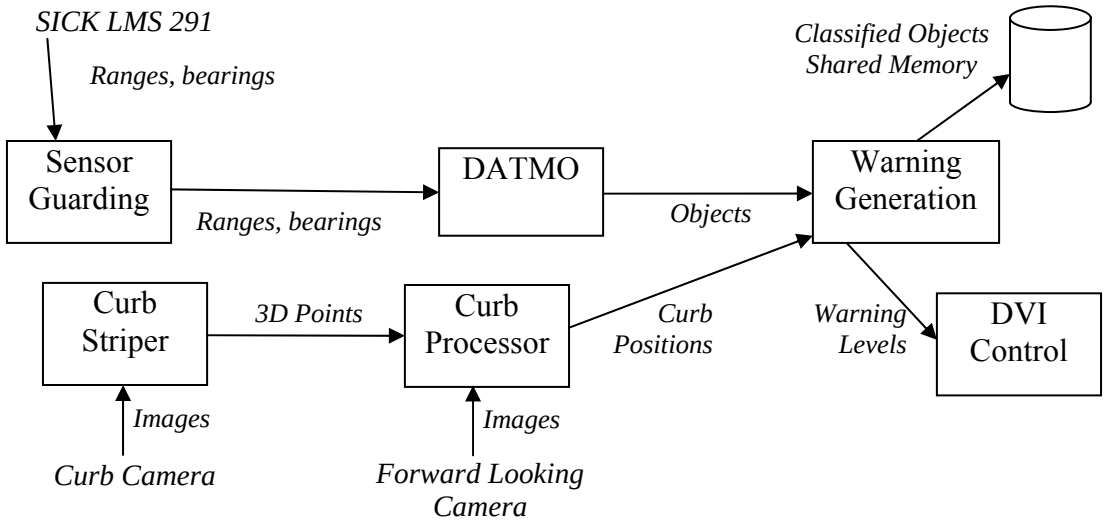
- The curb striper algorithm digitizes a laser strip painted on the curb and uses its knowledge of the intrinsic and extrinsic camera parameters to produce a set of detected 3D points to shared memory

- The curb processing algorithm combines the output of the curb striper with the vehicle state data to produce an estimate of where the curb was over the last few seconds. It then digitizes an image from the right rear forward looking camera and uses the curb estimate to initiate a visual search for the curb ahead of the bus. The resulting curb information is published via shared memory.

- The warning algorithm is configured to read the curb information and uses it to modify its warning level production.

- As with the left DVI control module, the right DVI control module monitors the right

warning levels and sets the lights appropriately. The right DVI control module does not monitor the switches. That is done by the left DVI control module alone.



In addition, the right side processor has the vehicle state estimation module that is connected to the various sensors and data source and produces the actual vehicle state for the rest of the system.

1.14.3.3 Central processor data flow

The central processor is responsible for many of the data collection and system management aspects of the system.

- It runs the central blackboard and process management modules.

- The four external side bus cameras feed into a quad-combiner which then feeds into an MPEG encoder card which the central processor reads. The MPEG stream is time tagged and saved to disk

- It runs modules attached to all the other various shared memory outputs on the system, such as vehicle state estimation, laser range data, classified objects, etc., and saves them to disk.

1.14.4 Integration with the FCWS

The connection to the Forward Collision Warning System (FCWS) is through a serial link. There is a gateway module running on the left processor which gathers together, packages, and writes the following information over that serial link:

- Warning levels
- Nearest object position and speed for each side
- Curb position, if any
- GPS position
- Vehicle speed
- Auxiliary bus state information, such as door open/closed status or lights status.

The gateway module also monitors the output from the FCWS and saves the following information:

- Front warning levels
- Nearest front object position and speed.
- The FCWS's estimation of the auxiliary bus state information.

The bus state information is duplicated because the actual hardware sensors may not be connected to the same system on different platforms. For example, on the Pittsburgh bus the SCWS system has direct access to the bus state information sensors and has a separate module to read and publish these values, but on the San Mateo bus, the bus state information is read from the FCWS and propagated to the rest of the system by the gateway module.

1.15. FCWS Software Introduction

This chapter focuses on the data acquisition program of the FCWS on integrated Samtrans bus 601. This includes most of the interfaces that serve as the bridge between the low-layer hardware/software drivers and the upper-layer application programs such as warning algorithms. The communication of FCWS and ICWS is specified in the ICD document.

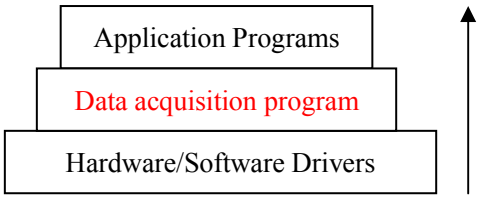


Figure 23 FCWS Data acquisition program

The purpose of the data acquisition program is to save data from sensors and synchronize the engineering computer with a video recorder. Basically, the data acquisition program is comprised of an initialization process and a loop body; the program has a short period of time to save all files and then to abort when the power is turned off. The loop body is composed of the following actions:

1. Copy the sensor data from the database to the local memory.
2. Save sensor data from the local memory to a set of disk files.
3. Check power-off flag (if power-off flag is set, run power-off subroutine)
4. Check time consumed for file collection.
5. (If exceeds 15 minutes, open a new set of files)
6. Generate synchronization signals.
7. Wait for the 75ms flag.

The LIDAR data is saved every 75 ms, which is the lowest update rate. About every 15 minutes old files will be closed and a new set of files will be opened. A timestamp is also included with each entry.

1.15.1 FCWS Software structure

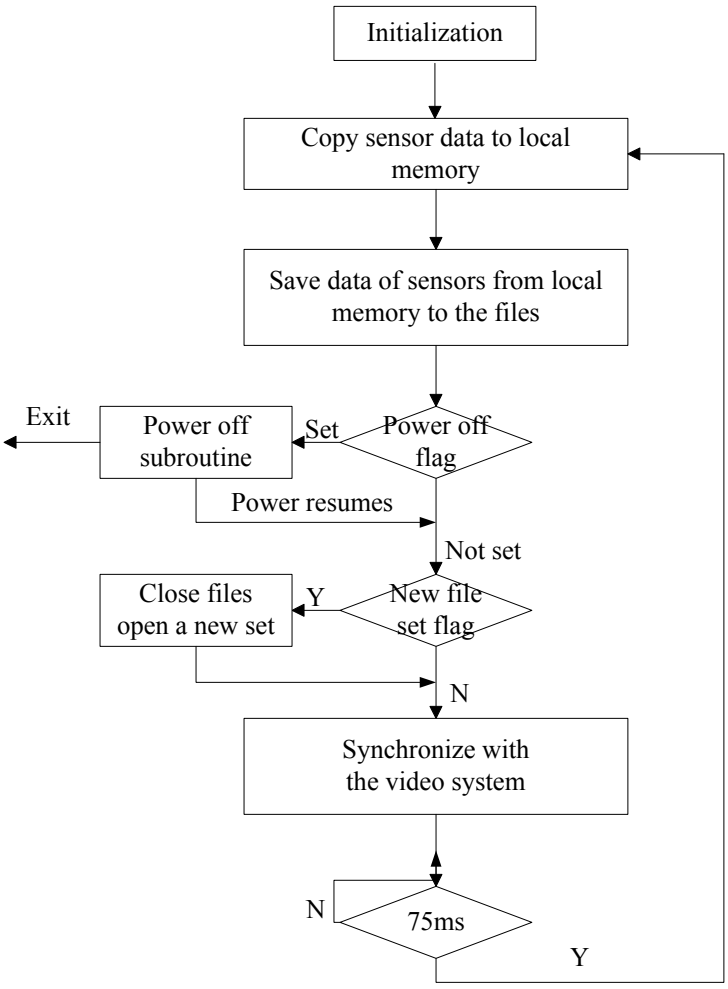


Figure 24 FCWS Software flow chart

1.15.2 FCWS Initialization

1.15.2.1 Define variables

1.15.2.1.1 File pointers (Global variables)

File pointer	Sensor
*f_RADARA	P-RADAR
*f_RADARB	D-RADAR
*f_LIDARO	F-LIDAR
*f_LIDARM	P-LIDAR
*f_LIDARN	D-LIDAR

Table 13. FCWS File pointers – sensors

1.15.2.1.2 System signals

Signal	Description
SIGINT	Interruption
SIGQUIT	Quit
SIGTERM	Terminate
ERROR	System error

Table 14. FCWS System signals

After initialization (signals added), whenever the program receives these signals, it will close files, log out of the database and exit. For example, Ctrl+C from the keyboard will generate a SIGTERM signal and this program will receive the signal then close files, log out of the database and exit.

1.15.2.1.3 Database variables

The following database variables are used for database read operation; each variable (a structure) contains some variables and an unsigned char. This char will be the pointer of the sensor data in the local memory after the execution of clt_read () function (database read operation).

Variable	Sensor
Db_data_radarA	P-Radar
Db_data_radarB	D-Radar
Db_data_LidarOA	F-Lidar(section A)
Db_data_LidarOB	F-Lidar(section B)
Db_data_lidarMA	P-Lidar(section A)
Db_data_lidarMB	P-Lidar(section B)
Db_data_lidarNA	D-Lidar(section A)
Db_data_lidarNB	D-Lidar(section B)
Db_data_long_input	Host-bus sensors
Db_data_gps_gga	GPS
Db_data_gps_vtg	GPS
Db_data_jeec2	J-bus
Db_data_dduA	DDU-display of P-Radar
Db_data_dduB	DDU-display of D-Radar

Table 15. FCWS Database variables – sensors

1.15.2.1.4 Sensor data pointers

The pointers listed below point to the sensor data in the local memory. For instance, pRADAR gets its value from the database variable: db_data_radarA or db_data_radarB that contains the pointer pointing to the RADAR data. The data structures of RADAR, LIDAR, host-bus sensors are shown in FCWS hardware documentation. These pointers are then used to save sensor data from local memory to a hard disk.

Pointers	Sensor
*pradar	RADAR
*pddu_display	DDU-display
*plidarA	Lidar(section A)
*plidarB	Lidar(section B)
*plong_input	Host-bus sensors
*pgps_gga	GPS (position)
*pgps_vtg	GPS(speed)
*plong_jeec	For J-bus

Table 16. FCWS Sensor data pointers

1.15.2.1.5 Time variables

1.15.2.1.5.1 Start_time, Curr_time

These two variables are used to check the time consumed for file collection, if Curr_time-Start_time>15 minutes, the old files will be closed and a new set opened.

1.15.2.1.5.2 Hour, minute, second, millisec

These four variables are used to generate the time of day an entry is recorded.

1.15.2.2 Process user switches

A user should specify the time for file collection. In this program, 15 minutes are allotted for file collection. Command format: wrfiles3 –m 15

1.15.2.3 Open a serial port for the titler

This port is used to send current time (hour, minutes, second) to the titler for adding a timestamp.

1.15.2.4 Log in to the database

In order to read data from the database, we need to get a node ID and then log in to the database.

1.15.2.5 Get the current time

The current time is used to calculate time consumed for file collection.

1.15.2.6 Open files

File name	Sensor
AMMDDSSS.dat	P-Radar
BMMDDSSS.dat	D-Radar
OMMDDSSS.dat	F-Lidar
MMMDDSSS.dat	P-Lidar
NMMDDSSS.dat	D-Lidar
EMMDDSSS.dat	Host-bus sensors and others

Table 17. FCWS File name format

In the above names, MM is replaced by a 2-digit month code, DD is replaced by a 2-digit day code, and SSS is replaced by a 3-digit serial code. Serial codes for a given day start at 000 and proceed to 999. Detailed information of the file format is in the program comments.

1.15.3 FCWS Loop body

1.15.3.1 Database operations

The program copies the specified sensor data from the database to local memory consecutively before performing any disk operations. (Please note, we do not read data of a specified sensor, save it to a disk file, and then read data of another sensor.) As a result, the timestamp of all disk files can be consistent. The reason is that memory operation are much faster than disk operations, given the same quantity of data transmissions.

1.15.3.2 Disk file operations

Disk file functions save: the data of RADAR sensor and DDU display, two sections of LIDAR sensor data (section A and section B combined) and saves data of host-bus sensors, GPS, and J-bus data. All these save functions perform memory read and disk write operations and add the same timestamp to their files.

1.15.3.2.1 RADAR file format (P-RADAR, D-RADAR)

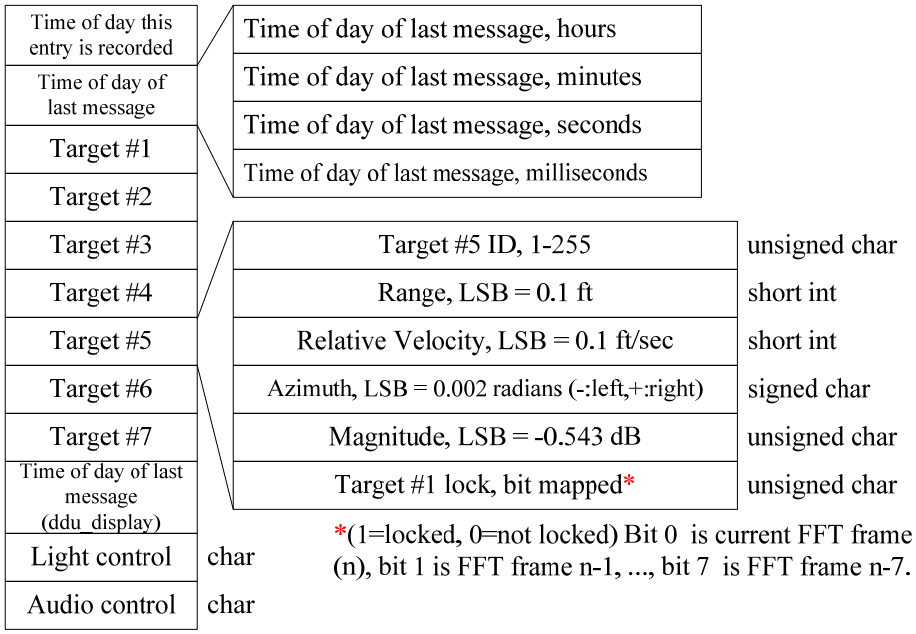


Figure 25 FCWS RADAR file format

1.15.3.2.2 LIDAR file format (F-LIDAR: First generation)

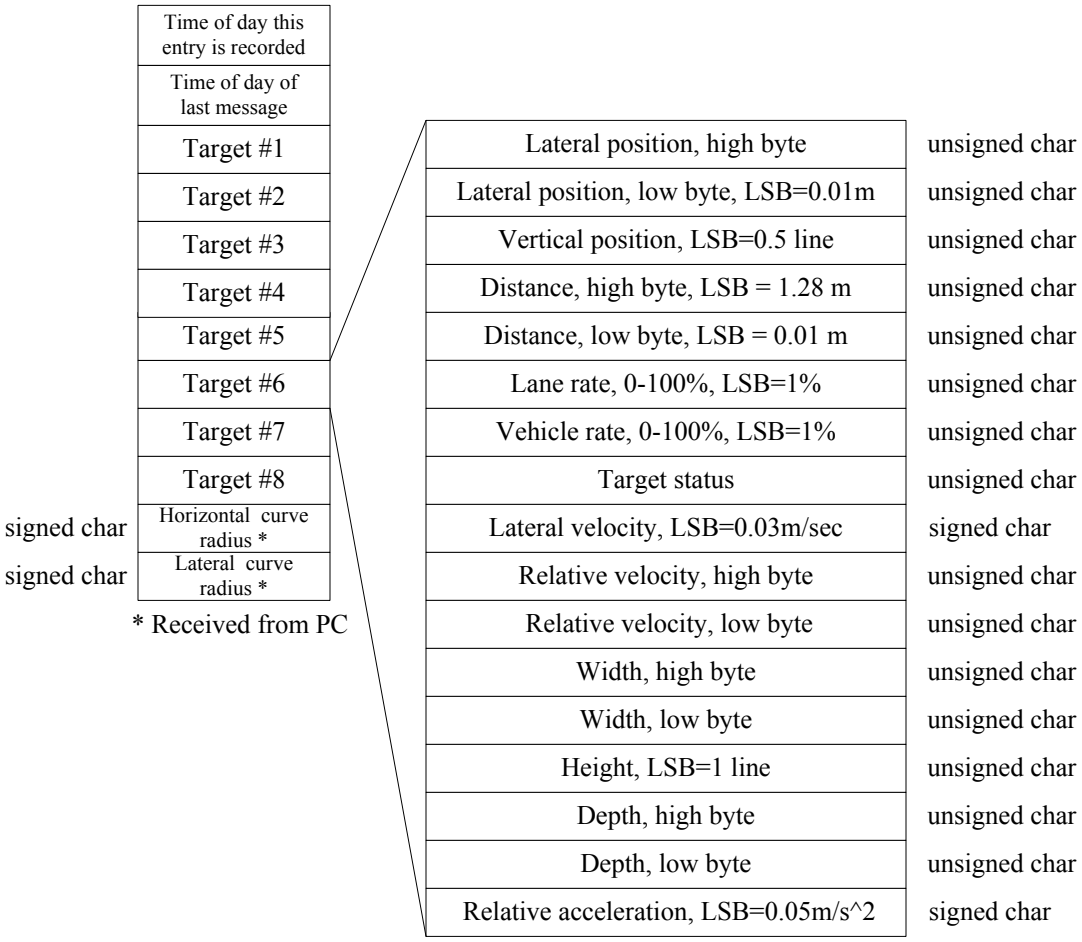


Figure 26 FCWS lidar file format

1.15.3.2.3 Host-bus sensor file format

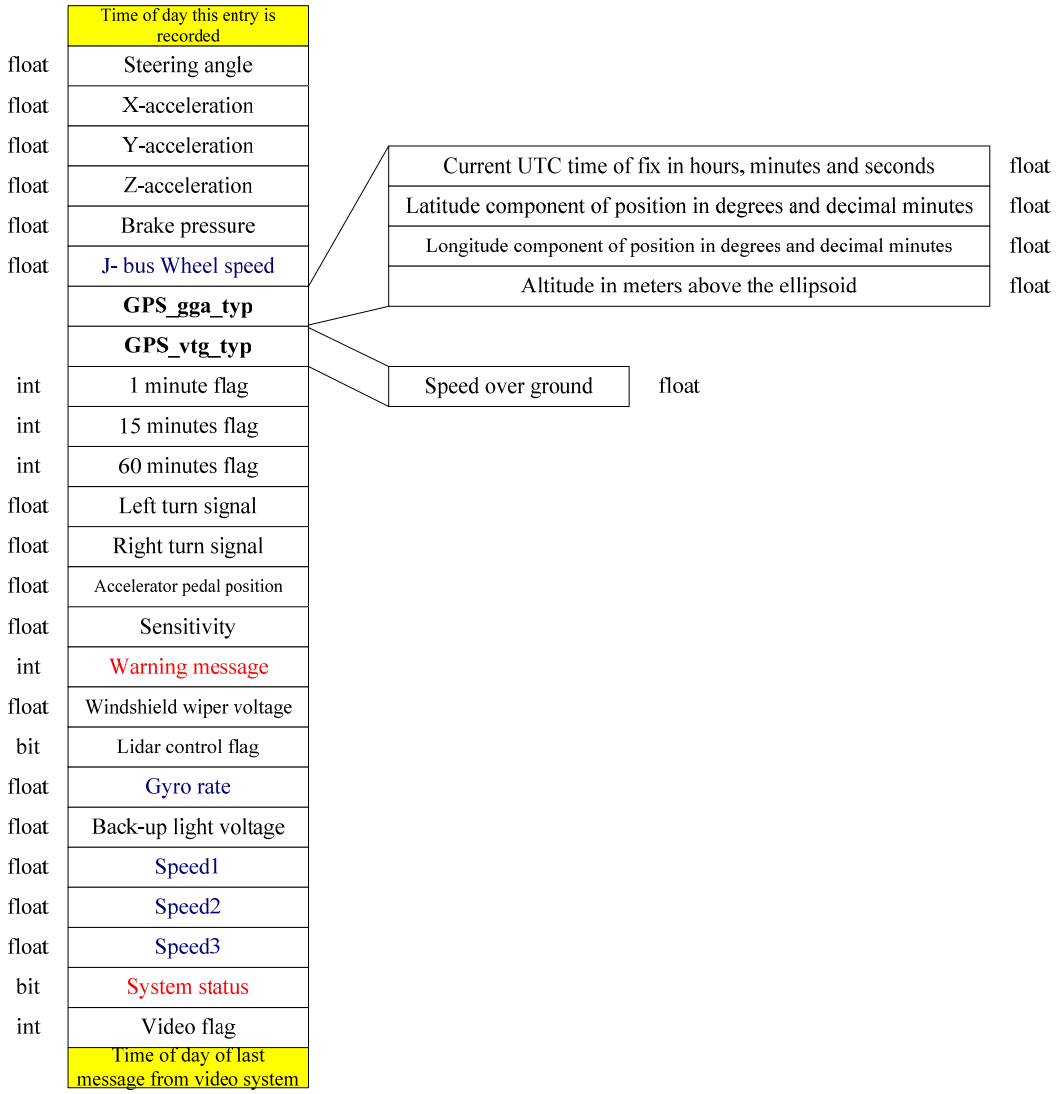


Figure 27 FCWS Host-bus sensor file format

1.15.3.3 Check power off flag

If the power off flag is set, there is less than 1 minute to close files and clear video alarm signals. If power resumes within 1 minute, a new set of files will be opened, a new timestamp will be sent to the video recorder and all synchronization signals will be cleared before the program check time consumed for file collection.

1.15.3.4 Check time to open a new set of files

If the current time - the start time >15 minutes, the old files will be closed and a new set of files will be opened. Whenever a new set is opened the current timestamp will be sent to the video recorder through the serial port already opened.

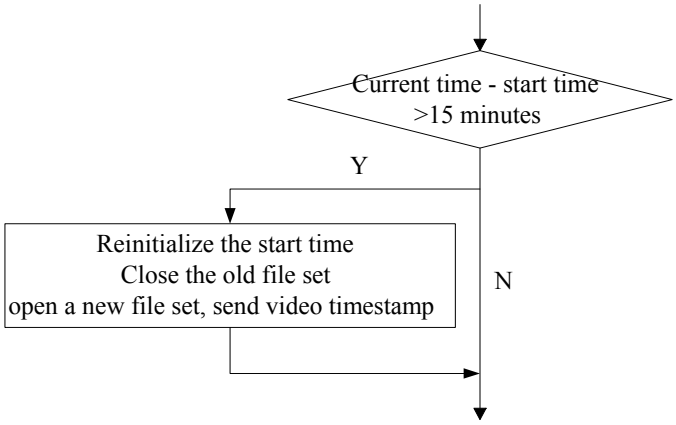


Figure 28 Check time to open a new set of files

1.15.4 FCWS Synchronization

The video files recorded and the sensor file recorded need to be synchronized to describe the same scenario. The engineering computer will send the master time to the video recorder after ignition to synchronize the two systems in real time. The video recorder will adjust its clock accordingly. The engineering computer will send instructions to the video recorder to open or close a file for video recording when it opens or closes a new set of engineering files. The video recorder also records the start time and end time of each video file for synchronization verification.

1.15.5 FCWS Program exit

The program will abort when the power is turned off for four minutes. The program will exit when there are:

(1) Signals (added in initialization) received

(2) Invalid user switch or bad number of minutes for file collection.

(3) Failure to initialize the timer (75ms).

(4) Error in opening serial port for video timestamp.

(5) Database initialization error, database reading error, database update error.

ALGORITHM DEVELOPMENT

1.16. Object Tracking Using Scanning Laser Rangefinders

CMU has developed software for the tracking of vehicles and pedestrians using scanning laser rangefinders mounted on a moving vehicle. Although the system combines various algorithms and empirical decision rules to achieve acceptable performance, the basic mechanism is tracking of line features, so we call this approach *linear feature tracking.*

There are three major parts to this presentation:

- Introduction of the sensor characteristics, comparison with other tracking problems, and discussion of some specific problematic situations.
- Presentation of the structure and algorithms used by the tracker.
- Discussion of the performance and limitations of the current system.

1.16.1 Input / Output example

To get some idea of what the tracker does, consider the tracker input and output. Figure 29 is a portion of an input frame from the laser rangefinder:

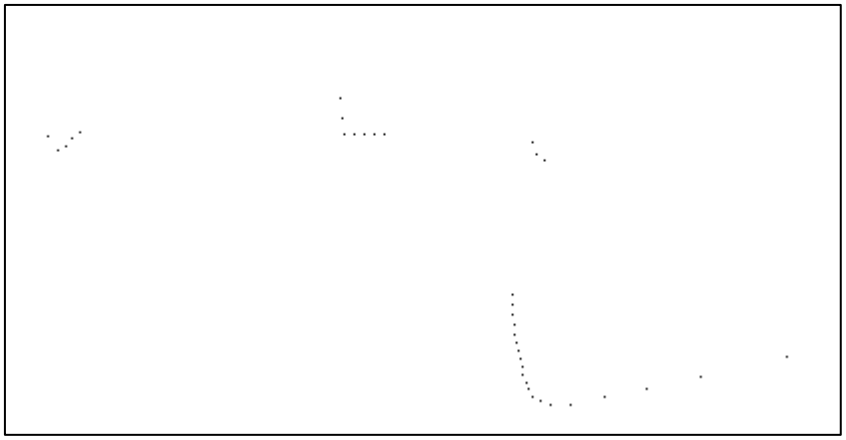


Figure 29: Tracker input (one frame)

Figure 30 is a visualization of the tracker output. The numbers are track identifiers, with additional information displayed for moving tracks. Track 38 (brown) is a car moving at 5.7 meters/sec and turning at 21 degrees/sec. The light blue arc drawn from track 38 is

the projected path over the next two seconds. The other tracks are non-moving clutter objects such as trash cans and light poles. The actual scanner data points are single pixel dots. The straight lines have been fitted to these points. An X is displayed at the end of the line if we are confident that the line end represents a corner.

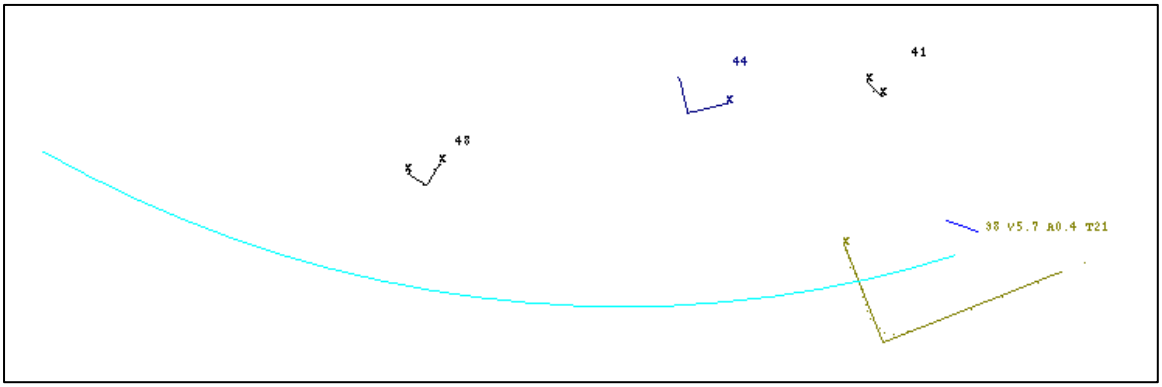


Figure 30: Tracker output example

1.16.2 Sensor characteristics

A laser rangefinder (or LIDAR) is an active optical position measurement sensor. Using the popular time-of-flight measurement principle, a laser pulse is sent out by the sensor, reflects off of an object in the environment, then the elapsed time before arrival of the return pulse is converted into distance. In a scanning laser rangefinder, mechanical motion of a scanning mirror directs sequential measurement pulses in different directions, permitting the building of an approximation of a 2D model of the environment (3D with two scan axes.) We will use the term *scanner* as a shorter form of scanning laser rangefinder.

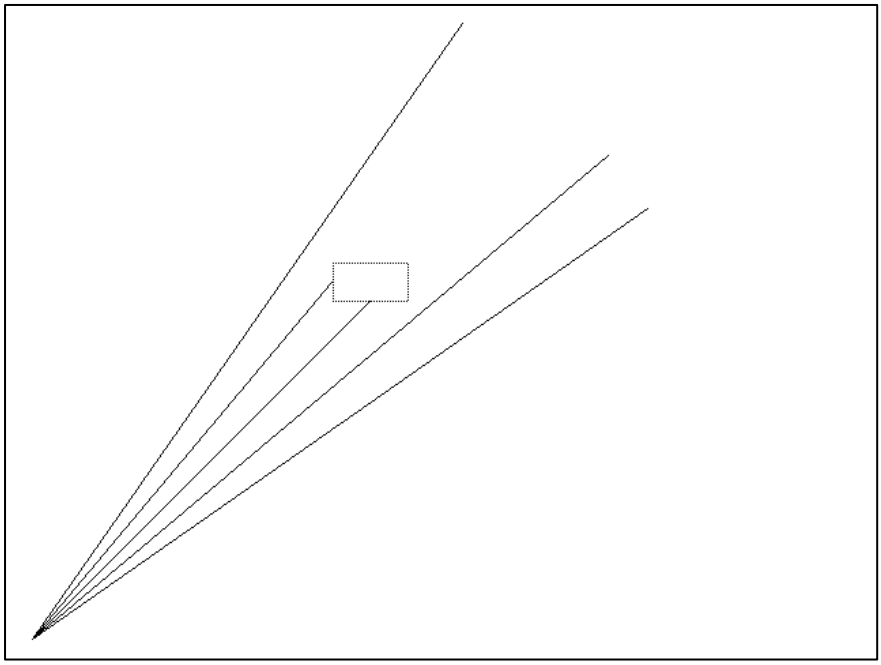


Figure 31: Scanner angular resolution

The scanning mirror moves continuously, but measurements are made at discrete angle increments (see Figure 31.) Though this is not actually how the scanner operates, the effects of angular quantization are easier to understand if you visualize the scanner as sending out a fixed pattern of light beams which sweep across the environment as the scanner moves (sort of like cat whiskers.)

When viewed in the natural polar coordinates, the rotational (azimuth) and radial (range) measurement errors are due to completely different processes, and have different range dependence:

- The azimuth error is primarily due to the angular quantization, though this is related to the underlying physical consideration of laser spot size. For a given beam incidence angle on the target, the Cartesian position uncertainty is proportional to the range.
- The range measurement error comes from the per-pulse range measurement process, and in a time-of-flight system is largely due to the timer resolution. This results in a range accuracy that is independent of distance.

Linear feature tracking was developed for a collision warning system for transit buses. This system uses the SICK LMS 200, which is a scanning laser rangefinder with a single scan axis. The scanner is oriented to scan in a horizontal plane, and all processing is done using 2D geometry in this scan plane. Performance specifications are 1cm range resolution, 50 meter range, 1 degree basic azimuth resolution and 75 scans/second update rate. The output of the scanner is simply a vector of 181 range values. If the measurement process fails due to no detectable return, this is flagged by a distinct large range value.

Note that with these range and angle resolutions, the position uncertainty is dominated by azimuth quantization throughout the entire useful operating range. At a practical extreme range of 20 meters, a one degree arc is 34cm, whereas the range resolution is still 1cm.

1.16.3 The tracking problem

Given this sensor, we would like to identify moving objects, determine the position and velocity, and also estimate higher order dynamics such as the acceleration and turn rate. The tracker must also be computationally efficient enough so that it can process 75 scans a second in an embedded system with other competing processes.

A *track* is an object identity annotated with estimates of dynamics derived by observing the time-change. The function of the tracker is to generate these tracks from a time-series of measurements. The purpose of maintaining the object identity is twofold:

- We need to establish object correspondences from one measurement to the next so that we can estimate dynamics.
- The object identity is in itself useful as it allows us to detect when objects appear and disappear.

In general, tracking can be described as a three-step process which is repeated each time a new measurement is made:

1. *Predict* the new position of each existing track based on the last estimate of

position and motion.

2. *Associate measurement data* with existing tracks. If there is no good match, consider making a new track.
3. *Estimate* new position and motion based on the difference between the predicted position and the measured one.

1.16.3.1 Comparison of tracking with laser scanner vs. other sensors

The problem of tracking moving objects using a scanning laser rangefinder is in some ways intermediate in characteristics between long range RADAR tracking (e.g. of aircraft) and computer vision tracking.

What advantages for object tracking does a laser scanner have over computer vision? Two difficult problems in vision based tracking are:

- *Position:* determination of the position of objects using vision can only be done using unreliable techniques such as stereo vision or assuming a particular object size. Position determination is trivial using ranging sensors like RADAR and laser scanners, as long as there is adequate angular resolution,
- *Segmentation*: when two objects appear superimposed by our perspective, how do we tell where one ends and the next begins? Range measurement makes segmentation much easier because foreground objects are clearly separated from the background.

An important problem that laser scanners have in common with computer vision is *point correspondence:* given two measurements of the same object, which specific measurements correspond to the same point on the object.

In long range RADAR, the point correspondence problem typically doesn't exist -- the object size is at or below the angular resolution, so the object resembles a single point. In contrast, when a laser scanner is used in an urban driving situation, we need to be able to track objects whose size is 10 to 100 times our angular resolution. Not only do the tracked vehicles not resemble points, after taking into consideration the effect of azimuth resolution, they often effectively extend all the way to the horizon in one direction.

When the size of objects can't be neglected, this creates ambiguity in determining the position of the object (what point to use). Since a tracker estimates dynamics such as the velocity by observing the change in position over time, this position uncertainty can create serious errors in the track dynamics.

As in computer vision, the extended nature of objects does also have some benefits. Because we have multiple points on each object, we can make use of this additional information to classify objects (bush, car, etc.)

1.16.3.2 Shape change

It is a crucial aspect of the tracking problem considered here that the laser rangefinder is itself in motion. If the scanner is not moving, the problem of detecting moving objects is trivial: just look for any change in the sensor reading. Once the scanner is moving, we expect fixed objects to appear to move in the coordinates of the scanner, and can correct for this with a coordinate transformation.

It is assumed that the motion of the scanner is directly measured, in our case by a combination of odometry and an inertial turn rate sensor. Since tracking is done over relatively short ranges and short periods of time, the required accuracy of the estimate of scanner motion is not great, and relatively inexpensive sensors can be used.

The more intractable difficulty related to scanning from a moving vehicle is that, even after object positions are corrected by a coordinate transform, the appearance still changes when we move due to the changing scanner perspective. The scanner only sees the part of the object surface currently facing the scanner. As the scanner moves around a fixed object, we see different contours of the object surface.

The shape change doesn't cause any serious difficulty for determining that scan data corresponds to the same object from one scan to the next because the change is small.

What is difficult is determining that these small changes are due to changes in perspective, and not actual motion of the tracked object.

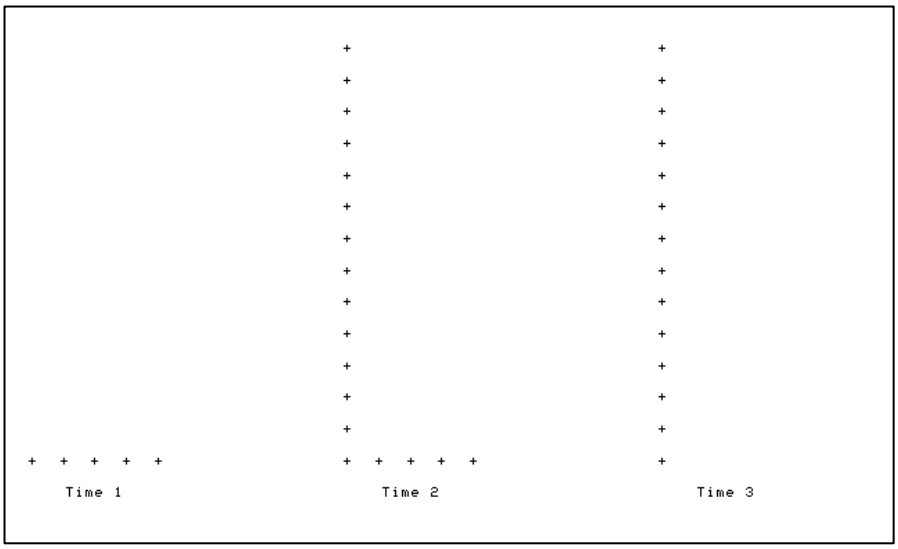


Figure 32: Shape change

To get a sense of the shape change problem, consider a naive algorithm which considers the object position to be the mean position of the object's measured points (see Figure 32.) Suppose that we are driving past a parked car. At time 1, we see only the end of the car. At time 2, we see both the side and end. By time 3, we only see the side. During this process, the center of mass of the point distribution shifts to the left, giving the parked car a velocity moving into our path, causing a false collision prediction. The point distribution also moves in our direction of motion creating false velocity in that direction.

1.16.3.3 Occlusion

Another problem happens when a small object moves in front of a larger background object (see Figure 33.) In this case, what is in effect the shadow of the foreground object creates a false moving boundary on the background object (as well as splitting the background object in two.) Due to the changing perspective, moving shadows also appear when both objects are fixed but the scanner is moving.

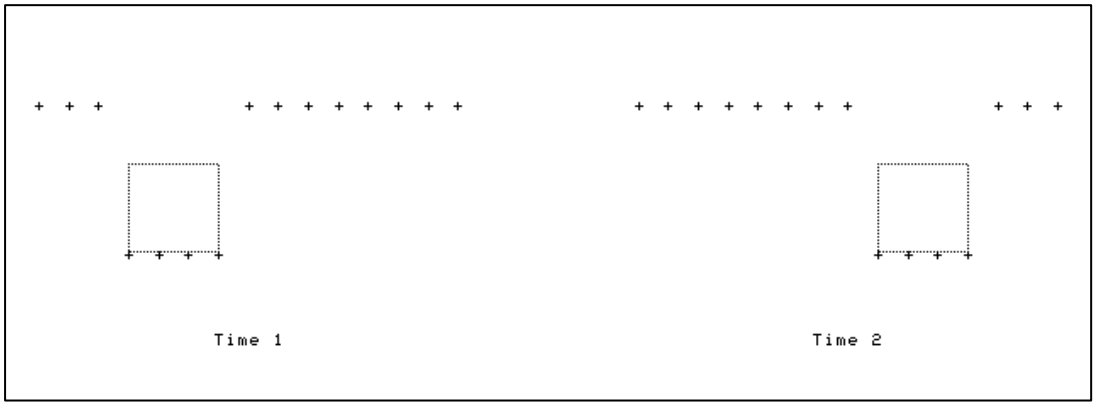


Figure 33: Occlusion

1.16.3.4 2D scan of a 3D world

Two major problems come from using a single axis scanner:

- When the scanner pitches or rolls, we see a different contour of each object, and if the surface is not nearly vertical, we may see a large amount of motion.
- When the ground is not flat, the scanner beam may hit the ground, resulting in seeing the ground itself as an obstacle. Due to pitch and roll, these ground-strike returns may also appear to be rapidly moving.

Use of a scanner with several beams that scan in parallel can greatly help with this problem because we can detect when the beam is striking an object that is significantly sloped, and either disregard it or attempt to compensate in some way.

1.16.3.5 Vegetation

With some objects, the outline seen by the scanner appears to fluctuate in a random way as the scanner moves. Vegetation has this problem. Figure 34 shows the superimposed points from 20 scans combined with markers for the points from one single scan. Clearly there is a great deal of noisy fluctuation of the range measurements. Also, the underlying outline which we can see in the superimposed scans is complex enough to defy simple geometric models.

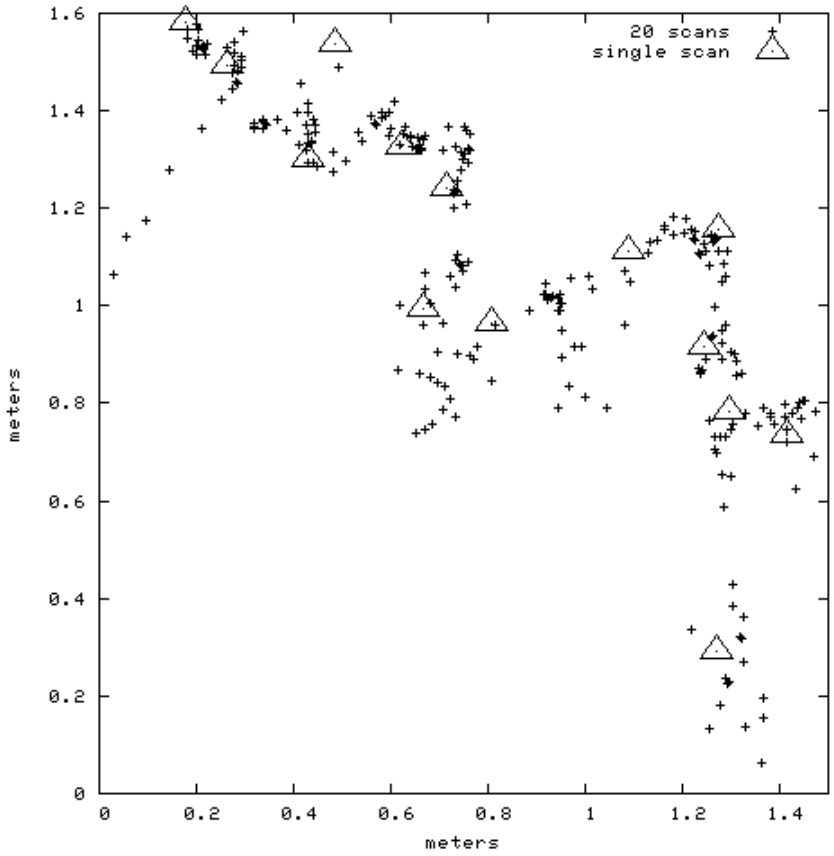


Figure 34: Vegetation

1.16.3.6 Weak returns

Some objects have very poor reflectivity at the infrared frequency where the SICK scanner operates. Figure 35 shows an example of a car that is almost invisible to the scanner. During the 10 seconds that we drive by, we are able to build up a reasonably complete idea of the car (small dots), apparently largely from specular glints. However, on any given scan, very little of the car is visible. In this particular single scan, we are mainly seeing inside the wheel wells (oblong areas area inside outline box.) Evidently the dirt inside the wheel well is a better reflector than the paint.

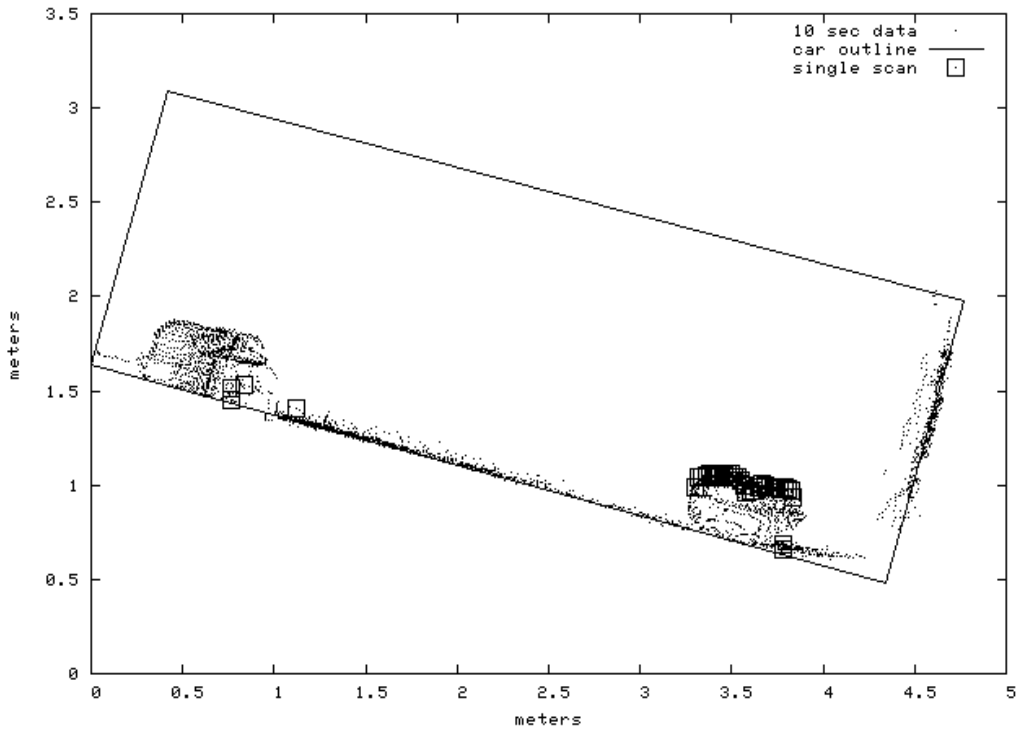


Figure 35: Weak returns

1.16.3.7 Clutter

Another cause of unclear object outlines is clutter: when objects are close together. In this case, it isn't clear whether to segment the data as one or two objects. If the segmentation flips between one and two objects, this causes apparent shape change. Clutter can also cause spurious disappearance of tracks, for example when a pedestrian moves close to a wall, and appears to merge with the wall.

1.16.4 Tracker structure and algorithms

These are the major parts of the tracker:

- Segmentation: group scanner points according to which object they are part of.
- Feature extraction: fit line and corner features.
- Prior noise model: assign feature error covariances using a measurement error model.
- Data association: find the existing track corresponding to each new segment, creating a new track if there is none.

- Dynamic model and Kalman filter: determine velocity, acceleration and turn rate from the raw position measurements.
- Track evaluation: assess the validity of the dynamic estimate and see if the track appears to be moving. Check how well the estimate "predicts" the measured past positions when time is reversed.

There are 60 numeric parameters used by the tracker. For concreteness and conciseness, we will refer to the specific numeric values that have been empirically tuned for our particular scanner and application, rather than to parameter names. Generally the parameter values are not all that sensitive, but for best performance with a different scanner or application, different values would be used.

Also, since the source code is available and well commented, we will avoid in-depth discussion of implementation details better read from the source. In particular, although efficiency is one of the important characteristics of the tracker, we won't do much in-depth discussion of performance-related issues.

One performance consideration is worth discussing because it affects the structure of the algorithm, especially in the segmentation and feature extraction steps. We have exploited two major geometric constraints that come from the use of a single scanner:

- Given an assumption that all corners are 90 degrees, at any time it is possible to see at most two sides and three corners of an object. Data structures are designed for this fixed number of linear features, rather than an arbitrary number. This also simplifies the feature correspondence problem in data association.
- In various places we exploit the assumption that the inherent azimuth ordering in the scanner output is also an ordering of consecutive points on the object surface.

Both of these assumptions break down if there is more than one scanner. We have demonstrated one way to use multiple scanners: convert all the scan points into a point cloud in Cartesian coordinates, and then convert each point back to polar coordinates of a

single "virtual scanner." Although not a great solution, it does show that the limitation to a single scanner can be relaxed.

1.16.4.1 Segmentation

Segmentation takes the list of 181 range and azimuth points returned by the scanner and partitions it into sublists of contiguous points. Two neighboring points are contiguous if the separation is less than 0.8 meters. After segmentation, the points are converted into a non-moving Cartesian coordinate system by transforming out the effects of the known motion of the scanner.

During segmentation we also classify each point as *occluded* or normal. A point is occluded if an adjacent point in the scan is in a different segment and in front of this point, or if it is the first or last point in the scan. This flag has two uses:

- When an occluded point appears at the boundary of an object, we consider this to be a false boundary (the feature is *vague.*)
- We only count non-occluded points when determining if there enough points to create a new track or if the point density is high enough for a segment to be compact.

In segmentation, missing range returns are treated as points at maximum distance, and are not assigned to any segment. If there is a large enough dropout in the middle of an object, this splits the object into two segments.

1.16.4.2 Linear feature extraction

For each segment, we do a least-squares fit to a line and to a right-angle corner. Since the stability of feature locations is crucial for accurate velocity measurement, there are two refinements to the basic least-squares fit:

- Points are weighted proportional to their separation along the line. Since some parts of the object may be much closer than others, the point density along the object contour can vary a great deal. This weighting reduces problems with rounded corners that have high point density causing the line fit to rotate away from more distant points that actually contain more information about the overall rectangular

shape approximation.

- The 20% of points with worst unweighted fit are discarded, and then we refit. Although this reduces sensitivity to outliers from any source, the scanner has little intrinsic noise, so the effect is mainly on real object features that violate the rectangular model, notably rounded corners and wheel wells.

Because conceptually both the point spacing (distance along the line) and fit error (distance normal to the line) depend on the line (which is what we are trying to find in the first place) we use an iterative approximation. Each line fit requires three least-squares fits:

- An equal-weight least-squares fit of all the points in the segment, used to determine the point spacing for weighting.
- A trial weighted fit, used to determine the outlier points.
- The final weighted fit.

The position of each line end-point is determined by taking the ending point in the input points and finding the closest point lying on the fitted line.

1.16.4.2.1 Corner fitting

Corner fitting is done after fitting as a line. This is a somewhat degenerate case of a polyline simplification algorithm. We split the point list in two at the knuckle point: the point farthest from the line fit. The geometrically longer side is then fit as a line. Since we constrain the corner to a right angle, the long side fit determines the direction of the short side. All we need to do is determine the location of the short side, which is done by taking the mean position along the long side of the short-side points. The location of the corner itself is the intersection of the two sides.

When doing the corner fit, we test for the corner being well-defined (approximately right angle) by doing an unconstrained linear fit on the short side, and testing the angle between the two sides. The angle must be at least 50 degrees away from parallel to be considered a good fit.

The corner must also be convex, meaning that the point of the corner aims toward the scanner (hence away from the interior of the opaque object.) We impose this restriction because in practice concave corners only appear on large fixed objects like walls, not on moving objects.

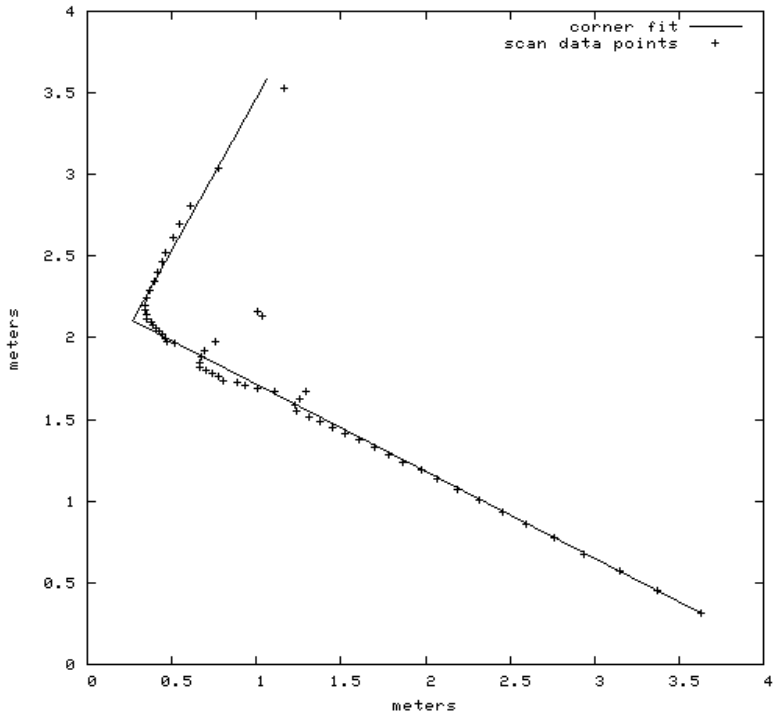


Figure 36: Corner fitting

Figure 36 is an example of corner fitting in the presence of corner rounding, outliers and variable point spacing. The fit matches the overall outline accurately fairly accurately.

1.16.4.2.2 Shape classification

After fitting and line and corner, each segment is given a shape classification:

corner Corner fit mean-squared error less than line fit and less than 10 cm.

line Line fit mean-squared error less than 10cm.

complex Fall-back shape class for objects with poor linear fit.

There are two Boolean shape attributes which are semi-independent from the shape-class:

compact Diagonal of bounding box < 0.7 meters and point density > 5 points/meter. The compact criterion is chosen so that pedestrians will be compact (as will lamp-posts, fire hydrants, etc.) Because compact objects are small, we can estimate their velocity reasonably accurately without having a good linear fit.

disoriented Rotation angle not well defined. True if the line or long side of corner has less than 6 points, the RMS fit is worse than 4 cm, or the line and corner fit disagree by more than 7 degrees and the chosen classification's RMS error is less than 4 times better than the alternative. Segments that are complex or < 0.7 meters diagonal are always disoriented.

The disoriented attribute is used to determine whether to use the change in orientation for turn rate estimation. Also, if a segment is disoriented and not compact, then it has a poor linear fit, and we are more skeptical of the motion estimate.

Due to the restrictions of single-scanner perspective and 90 degree convex corners, there is a small fixed set of possible features that a segment can have:

min, max The two diagonal corners of the bounding box (in world coordinates.) Defined for all shape classes.

first, last End points in a line segment, ends of the two sides in a corner segment.

corner Corner point in a corner segment.

center Rough estimate of object center derived from other features. Defined for all shape classes.

Figure 37 illustrates a segment with all features present.

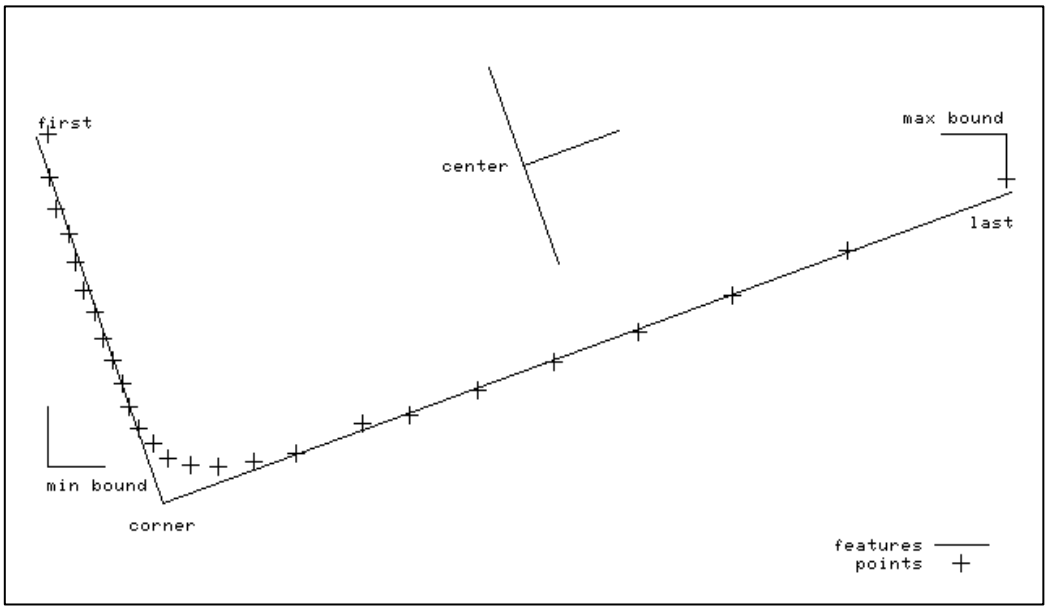


Figure 37: Features

1.16.4.3 Feature noise model

We observed earlier that due to the distinct angular and range resolution of the scanner, the shape of the position uncertainty in Cartesian coordinates is strongly asymmetrical.

The Kalman filter used in the tracker uses a statistical model of error in the measurement. If the actual measurement and modeling errors are white and zero-mean, then the Kalman filter is optimal. In practice, our errors differ greatly from this ideal, so there is no theoretical optimality. However inaccurate, in order to use the Kalman filter, we must attempt to capture the measurement error characteristics as a two-dimensional position covariance.

There are two parts of the noise model: the *prior* noise and the *adaptive* noise. The prior noise is determined from the segment points for one single scan, and is based mainly on the geometric properties of the scanner. The adaptive noise is estimated as part of the filter update, and will be discussed later.

The prior noise for linear features is computed from separate longitudinal and lateral variances that are then rotated into the world coordinates. The lateral variance is 1 cm or the mean-square line fit error, whichever is greater.

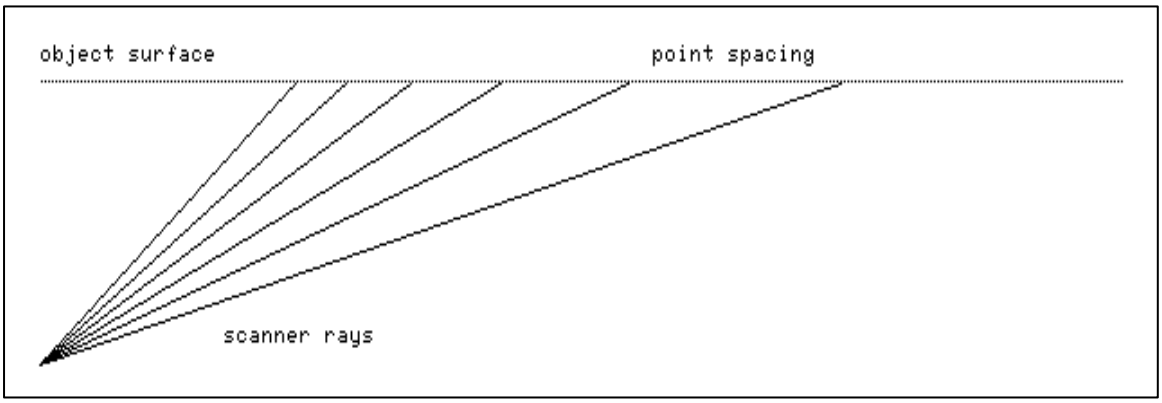


Figure 38: Incidence angle and point spacing

Geometrically, the longitudinal measurement uncertainty goes to infinity as the scanner ray approaches parallel to the line (see Figure 38.) Due to the segmentation algorithm, the observable inter-point spacing is limited to the segment threshold (0.8 meters.) The prior longitudinal variance is (0.3 * *spacing*) ^{2}, where *spacing* is the maximum inter-point spacing of the last 7 points at the end of the line. Using the actual inter-point spacing incorporates the geometric angular resolution effect, and also additionally discriminates against objects with missing returns (due to poor reflectivity.)

If a segment is disoriented, we are unsure of the orientation, so also unsure of the orientation of the measurement error uncertainty. In this case we bound the ratio of the longitudinal and lateral variance to 5 by increasing the lateral variance.

1.16.4.3.1 Vague line ends

A line-end feature can be marked as *vague,* which means that its longitudinal position is so poorly determined that the endpoint position should be regarded as arbitrary by the

tracker. A vague line end is still usefully localized laterally, and one end of a line can be vague when the other end isn't.

Due to shape change and occlusion, it turns out to be crucial to identify situations where the position of line ends is unreliable. This need results in a rather complex decision rule. A line end is vague if:

- The scanner point defining the endpoint position was occluded,
- The prior longitudinal error estimate is exceeds 15 cm, or
- The next adjacent point not assigned to this segment is at least 1.2 meters behind the line. Any intervening no-return points are skipped.

Rule 3 has these functions:

- It is a more precise occluded test in the case of a line with known orientation.
- It also marks ends as vague when the adjacent point in the next segment falls near the same line, hence may well be part of the same object that happened to be segmented separately due to missing returns.

1.16.4.4 Data association

Data association is the process of determining which current measurements correspond to existing tracks. In the context of the long-range tracking literature, our approach is basically nearest-neighbor. We pick one single segment in the measurement data which most closely resembles the prediction and use that for the new estimate. However, since each segment has many 2D points, the concept of "distance" is rather complex.

Our first approach to data association used the combined log-likelihood of the feature positions as a distance measure, then chose the most likely segment. However, we found this approach performed poorly, in that it both created bad tracks by associating separate objects and also caused good tracks to break up if the object shape changed too much. To the human eyeball the errors seemed rather silly.

1.16.4.4.1 Overlap based association

We now use an approach based primarily on spatial overlap between the track and the measurement data. The concept of overlap-based association is simple: given rectangles representing the outlines of the track and the measurement, the two are associated if the outlines overlap.

The actual overlap algorithm is somewhat different, and handles non-rectangular objects much better. For two objects to overlap, we require that at least one actual measurement point fall inside the track outline, and symmetrically, that when the track's last measured points are updated for predicted motion, that at least one of these predicted points falls inside the segment's outline. Since the points are on the edge of the object, we expand the object outline by 0.8 meters when checking the point overlap. If there is no linear fit (**complex** shape class), then we use the bounding box in fixed coordinates as the object outline.

This algorithm works well even when a moving track passes though a concave part of a large fixed object (wall, etc.) In this case, none of the wall's points fall inside the track outline, even though the bounding box for the wall may entirely enclose the track.

1.16.4.4.2 Track splitting and merging

In addition to its robustness, another advantage of overlap-based association is that it leads to a straightforward way of detecting when tracks split or merge. In these cases, overlap does not establish a one-to-one correspondence between tracks and segments. If a track splits, then there will be two segments overlapping the track. If two tracks merge, then there will be two tracks overlapping one segment.

Actual splitting or merging can happen when pedestrians get into or out of vehicles. In practice, segmentation errors are the most common cause of splitting and merging. In the presence of clutter, an object may appear to split from or merge with another nearby object. Missing returns can also create holes in the middle of objects, causing them to split in two.

Whatever the cause, it is important to detect when tracks merge because it is a common reason for track disappearance. See section *Track Creation and Deletion* where we discuss the "died without offspring" test.

When two tracks join, the ID of the merged track is that of the older track.

1.16.4.4.3 Maximum closeness:

In addition to split/merge situations, our fairly permissive overlap test can also generate complex overlap relationships when tracks are simply close together. Since in general the overlap test may create a many-to-many association between tracks and segments, we need a procedure for deciding which particular association to make.

This discrimination is done based on *maximum closeness,* where closeness of a track and segment is defined as the sum of the reciprocal of the individual feature distances. This differs from the RMS distance in two important ways:

- The result is dominated by the best agreeing features, not the worst, so it is not confused by the gross shape change which happens when tracks split or merge, and
- The result is not normalized by the number of features in correspondence. The more features, the higher the closeness. In a split/merge situation, we want to discard the track or segment with less information.

Tracks are associated from oldest to newest, which gives older tracks preference. A track may associate with a segment that is not mutually closest (there may be another newer track that is closer.)

1.16.4.4.4 Feature correspondence

Once we have decided which track and segment correspond, we are faced with the problem of *feature correspondence*: which features in the track and segment represent the same point? Due to our extremely restricted feature model, this is much simpler than it might be.

The only case where features may not correspond directly is when associating lines and corners. This happens when we move around an object, sometimes being able to see two sides, sometimes only one. Correct handling of this case is important because often visibility of one side of an object will be marginal due to shallow gaze angle, and the segmentation will keep flipping between line and corner.

Often the corner will correspond to one of the endpoints of the line, but sometimes the line ends match better onto the ends of the sides of the corner. We associate the line with whichever side of the corner is a better directional match, as long as this has less mismatch than the direct ends-to-ends match.

1.16.4.4.5 Track creation and death

If we fail to associate a measurement segment with any existing track, then we consider creating a new track. A new track is created if the segment has at least 3 non-occluded points and is not a line with both ends vague.

If we fail to associate an existing track with any measurement segment, then we may delete the track. Tracks are deleted if they have not been associated for 10 cycles or the total number of previous associations, whichever is less.

When a track dies, we report this event, whether the track merged with another object, and also whether the track "died without offspring". This is determined by examining the split-from chain of all currently live tracks, seeing if the dead track was a parent. This test is done to support detection of objects that may still be present but have passed out of the scanner field of view (pedestrians that have fallen down.) If some track that split off is still alive, then the split was probably spurious (due to a segmentation problem), so there is no real disappearance.

1.16.4.5 Track state and dynamic model

The basic dynamic model is constant acceleration and constant turn rate. There are nine state variables: XY incremental motion, XY velocity, XY acceleration, position theta,

heading theta and angular velocity. The first six states estimate translational motion and the last three estimate heading and turn rate. All of the translational states are expressed in the fixed world coordinate system. The track incremental motion is initially zero, and is an estimate of the net XY motion since track creation.

Using a constrained vehicle dynamic model such as the bicycle model, the velocity and acceleration are single-dimensional when expressed in the track's moving coordinates. We implement a similar effect by rotating the velocity and acceleration by the predicted incremental rotation on each update cycle. However, we maintain a 2D velocity and acceleration, so our model allows for velocity and acceleration normal to the vehicle heading (in addition to the apparent acceleration due to turning.) While this can happen in a skidding vehicle, this is not really relevant in our application. The interpretation of the 2D acceleration is somewhat non-obvious, as it is the residual acceleration after the effect of turning has been removed.

There are actually separate linear Kalman filters for linear estimation (6 state) and rotational estimation (3 state.) In the future we plan to investigate a combined 7 state nonlinear filter which would simultaneously estimate the linear and rotational motion without unnecessary degrees of freedom.

The dynamic model for compact objects (such as pedestrians) is modified by forcing acceleration to zero. Pedestrians don't spend much time in a constant acceleration regime, so predicting parabolic trajectories doesn't make sense. We do however predict pedestrian “turn rate” as in the bicycle model. Though pedestrians don't really turn on constant curvature paths, they don't normally turn abruptly either, so this seems to have some value.

1.16.4.5.1 Rotational estimation

Additional complexity for rotational estimation comes from the fact that we can sometimes (see *Shape Classification*) directly measure vehicle heading (orientation theta), and thus fairly directly measure the turn rate. For other tracks, the only way to

discern the turn rate is to observe the change in the direction of velocity (heading theta) over time.

Due to the kinematics of the bicycle model, the instantaneous velocity of a turning vehicle is not normal to the front or back surface of the vehicle. Fortunately, we are really mainly interested in estimating the angular velocity, not the heading, and the time rate of change of the velocity vector is the same as for the orientation of any part of the vehicle outline.

However, we run into problems when we are forced to switch between these two orientation estimates (when the **disoriented** flag changes.) This can create spurious jumps in position that would be interpreted as angular velocity. By maintaining separate estimates for these two different headings we avoid this confusion.

1.16.4.5.2 Tracking features

Because we have up to three features for each for each track, we have a data fusion problem. How do we combine the motion estimates from the separate features into one track motion estimate? Fortunately, the Kalman filter provides a natural framework for data fusion. Each feature can be considered an independent measurement of the position. Because each feature has its own error model, the Kalman filter weights the contribution of each feature appropriately.

One complication is that the features do not have the same position because they are different points on the object. To allow motion assessment of individual features, we need to know the previous position of each feature. This is done by a modified Kalman filter structure. Each feature has an independent position, but shares its motion with the track. The position innovation is determined by subtracting the old feature position and the new measurement. The Kalman gain is the computed using the feature's position noise and the track state covariance. The position part of the track state change is applied to both the feature position and the track incremental motion.

Vague line ends are tracked specially. Although vague line ends are given a large longitudinal position covariance, this does not cause a sufficiently profound suppression of spurious longitudinal motion. There is large rapid motion of vague line ends in common situations such as a side of an object becoming visible when we round the corner. To completely suppress this motion, we zero the longitudinal component of velocity and acceleration state change for vague line ends. We allow the incremental motion to be affected so that huge jumps in longitudinal position can still cause the association to fail (due to excess state change Mahalanobis distance.)

Which features contribute to the motion estimate depends on the shape classification. Of course, features not present can't contribute. Less obviously, the motion of the bounding box corners only contributes to the motion estimate when the shape class is **complex**. We track the position of the bounds regardless, so that we have a valid feature position and noise estimate available if the shape switches to **complex**.

1.16.4.5.3 Track center

We maintain a crude estimate of the track center. This is done via the center pseudo-feature. During shape classification, we compute the center as the midpoint of the object. For center finding on linear objects, we extend vague sides to be at least 2 meters long. This can be regarded as a prior object model that everything is a 2 meter square.

Because this is a very low quality estimate, and is in any case contains no additional motion information not present in the other features, the motion of the center feature is measured using a separate Kalman filter. This gives some smoothing of the center position, while avoiding any interaction with the actual motion estimate.

1.16.4.5.4 Noise adaptation

We make use of measurement noise adaptation to estimate the position covariance for each feature. The primary benefit of measurement noise adaptation is in its effect on the data-fusion aspect of the Kalman filter: noisy features contribute less to the motion estimate. This is valuable because some features are much noisier than others.

The noise is estimated from the statistics of the measurement residue, which is the disagreement between the predicted feature position and the measured one. (With our dynamic model, the residue is the same as the feature position innovation.)

The covariance of the residue is an estimate of the position measurement covariance.
In theory, for computing an adaptive measurement noise, we should reduce the noise residue by the estimate covariance to represent the contribution to the measurement residue from the estimate uncertainty. However, this risks creating small or even negative measurement noise when the estimate covariance is too high. Since we are knowingly inflating process and measurement noise to deal with time-varying behavior and modeling error, subtracting out the state covariance would to cause problems.

If the mean of the residue is not nearly zero, this indicates a non-zero-mean error. In some applications it would be appropriate to subtract out this mean from the measurement; in our case there is no independent measurement of the feature position, so a significant mean error means that we are not tracking properly. If the residue mean exceeds 10 cm, we reset the feature position from the current measurement and clear the residue mean to zero.

The need for this resetting comes from a sort of instability that the noise-adaptive tracker exhibits in the presence of time-varying non-zero-mean disturbances. In short, if a feature position drifts around, and no compensating track velocity is inferred (perhaps due to data fusion correctly rejecting spurious motion), then the measurement covariance for that feature becomes inflated, and this further degrades the tracking performance for that feature. Our response in this situation is to allow the feature to be basically ignored by fusion, but to keep the feature position approximately correct by resetting the position when the residue mean becomes too large.

Residue update has two phases: during track startup, we compute recursive mean and covariance with equal sample weighting. After 0.3 seconds, we switch to a first order response (exponential decay) with 0.3 second time constant. Initially, the sample is

small, so the estimate is unreliable. For the first 15 cycles, we use only the prior noise estimate. After 15 cycles, we use the sum of the residue covariance and (*prior_ noise* * 0.57). The prior noise then serves as a lower bound on the adaptive noise. This lower bound is particularly valuable in cases where the prior noise is high (like vague line ends.)

1.16.4.5.5 Improving the Kalman filter for non-Gaussian errors

The Kalman filter is optimal when the measurement error and process disturbance are Gaussian. One of the characteristics of the Gaussian is its light tails: it is very unlikely that a result will be very far from the mean. Unfortunately, when feature tracking breaks down, it can produce outlier measurements which are effectively impossible in the Gaussian model.

Since a Kalman filter doesn't suppress these glitches it is common to extend the Kalman filter by limiting the magnitude of outlier measurements or discarding them. We make use of this approach in several places:

- If the change in track state due to a measurement is too incredible, then we discard the measurement. This is done when the Mahalanobis distance of the state change with respect to the estimate covariance exceeds 6. Before discarding the measurement, we see if we can get a reasonable association by resetting a subset of the features. We reset any features that contribute a state change with Mahalanobis > 4 by setting their position to the measurement, zeroing their contribution to the innovation.
- The time rate of change (*d/dt)* of velocity, acceleration and angular velocity are limited to physically plausible values: 9.8 meters/sec^2, 5 meters/sec^3, 60 degrees/sec^2. Note that these limits are applied to the incremental state change, not just to the output estimate. For example, the acceleration limit is applied not only to the acceleration estimate, but (more importantly) also to the change in velocity estimate on any given update cycle. This prevents impossible jumps in position from causing big velocity jumps.
- The measurement of heading from feature orientation (orientation theta) is prone to

jump when there is a track merge/split or the shape classification changes. If the innovation exceeds 7 degrees in this situation, then we reset the position to the new measurement.

1.16.4.5.6 Track startup

When a track is newly created, the dynamics are unknown, so we assume that the velocity and acceleration are zero. If a track splits off of an existing track, we initialize the velocity and acceleration to the one for the existing track, but still leave the estimate covariance at the default (large) value.

Commonly tracks will come into range already rapidly moving, so this prior velocity can be significantly in error. It takes some time for the measurement to settle to the correct value, and *dv/dt* limiting prolongs this. During *dv/dt* limiting we hold acceleration fixed, as otherwise the acceleration slews wildly because the Kalman filter feedback loop is open. We also modify the Kalman filter update so that the velocity covariance is effectively fixed during *dv/dt* limiting, preventing spurious covariance convergence.

The physical *dv/dt* limit does not apply during track startup because this velocity change is not a physical acceleration. As a heuristic, we increase the *dv/dt* limit when the velocity covariance is high. We allow the velocity to change by 12 sigmas per second if this is higher than the physical limit.

1.16.4.5.7 Information increment test

When attempting data association of a segment and track, we find the *information increment*, which is a measure of how much the track was influenced by this measurement. If the information increment is low, then the track is not responding to the measurement because the measurement is considered too noisy relative to how certain we think we are of the current state. In this case, the tracker is not actually tracking anything, just speculating based on past data.

A common problem situation is that a track may change to a line with both ends vague. In this case, the track is localized in one direction only, and the longitudinal velocity is

unmeasured, which can lead to unacceptable spurious velocities. Low information increment can also happen if we reset all of the features in a track or when a track is very noisy (via noise adaptation.)

To keep the track from coasting indefinitely, we fail the association when the information increment is below 0.04 (response time constant of 25 cycles.) Tracks that are not associated for 10 cycles are deleted. If there is a one-to-one overlap relationship, then we pretend not to associate for purposes of the deletion test, but actually do associate. This helps to keep good tracks tracking properly in situations where the association is clear but the information is momentarily bad.

Though we call this "information" increment, the computation is really based on the covariance. This mimics the computation of the Kalman gain, which is what actually determines the response speed of the filter. If w_+ is an eigenvalue of the covariance after the measurement, and w_- is the eigenvalue before update, then the info increment is the mean of $w_- / w_+ - 1$ for the two eigenvalues. The eigenvalues are sorted so that we compare eigenvalues of similar magnitude.

The assumption is that the eigenvectors are little changed by any one measurement cycle, so the change in the sorted eigenvalues represents the change in uncertainty on each axis in the rotated (uncorrelated) coordinates. This insures that the track is well localized in two dimensions.

1.16.4.6 Track evaluation

The tracker outputs two flags for each track to aid in the interpretation of the result:

Valid true when the motion estimate is believed to be accurate

moving true if there is compelling evidence that the track is moving

Spurious velocities on fixed objects can easily cause false collision warnings. Since true imminent collisions are very rare, and fixed objects are very common, we must drive

down the reporting of false velocities on fixed objects to a very low level in order to get an acceptable rate of false alarms.

To achieve this we have developed an additional evaluation procedure that operates independently of the Kalman filter. We collect the last 35 segments (raw measurements) associated with each track, then check how well the track path matches up with the measurements if we project it backward in time.

If our dynamic model adequately describes the motion and the estimated parameters are close, then the paths should agree well. If the match is poor, then there is either unmodeled disturbance (rapid change in acceleration or turn rate), or there is a tracking failure due to problems in feature extraction, etc.

We accumulate two different streams of information about the difference between the back-predicted location and measured location of corresponding features:

1. The mean-square Mahalanobis distance according to prior position error. This is used to evaluate moving and valid.
2. The Euclidean distance normalized by the elapsed time from the measurement to now. The mean is a velocity correction and the covariance is an estimate of the velocity covariance. The velocity correction is added to the output velocity. If the covariance is bigger than Kalman filter covariance, then it is output instead.

We also find the sum of the position information for all features in the history that contributed to the distance estimate. This is used to determine whether the history has adequate quality to localize the track in two dimensions. The smaller eigenvalue of the information matrix must be greater than 35 meters^{-1}.

This information eigenvalue is also used to normalize the Mahalanobis distance back into a nominal length, which is then compared to a threshold for the moving/valid test. For a track to be marked valid, the distance must be less than 5 cm, and after that must stay below 15cm to remain valid. Though the units are meters, the physical interpretation is

obscure. The empirically chosen values seem reasonable if regarded as an RMS fit error in meters. The advantage of this distance measure over a simple RMS distance is that it takes into consideration the asymmetric prior error distributions generated by the position measurement model.

We also compare the past measurements to the null hypothesis that we are not moving at all. This is done using the same comparison procedure, but with no projected motion. We then compare the matching error of the two hypotheses. For the track to be moving and valid, the matching error of the moving hypothesis must be 4 times less than that of the fixed hypothesis. This rejects situations where there is equal evidence for moving and non-moving because one feature moves and another doesn't.

In order to minimize the effect of noisy variation, the results of the history analysis are filtered using an order 21 median filter before being tested against the above limits.

Because the history-based validation is fairly computationally expensive (about 250 microseconds per track on 1.2 GHz PC), we have used several optimizations:

1. Only do history test on apparently moving tracks. A track is apparently moving if it has a feature that has been tracked for at least 15 cycles, the speed is greater than 0.75 meters/sec, Mahalanobis distance of the velocity from zero exceeds 6. There is hysteresis in this test so that tracks tend to stay moving once they are initially moving. Also, if a track is a line with two vague ends that was not previously moving, then only the lateral component of the velocity is considered.
2. Only use the oldest 1/3 of the history data, as this contains most of the information about velocity error.
3. Limit the number of tracks validated on any tracker cycle to 4. There are seldom this many moving tracks, so this limit is rarely exceeded. The purpose is to bound the runtime of a single tracker iteration.

1.16.5 Evaluation

Linear feature tracking has been tested primarily as part of the larger collision avoidance system, with the main tuning criterion being minimizing the number of false alarms due to spurious motion. However, there are a number of ways that we can characterize the performance of the tracker. First, we can visually examine the output to get sense of the noise and response time of the output. Figure 39 shows the output for a car that comes into view turning a corner, then drives straight.

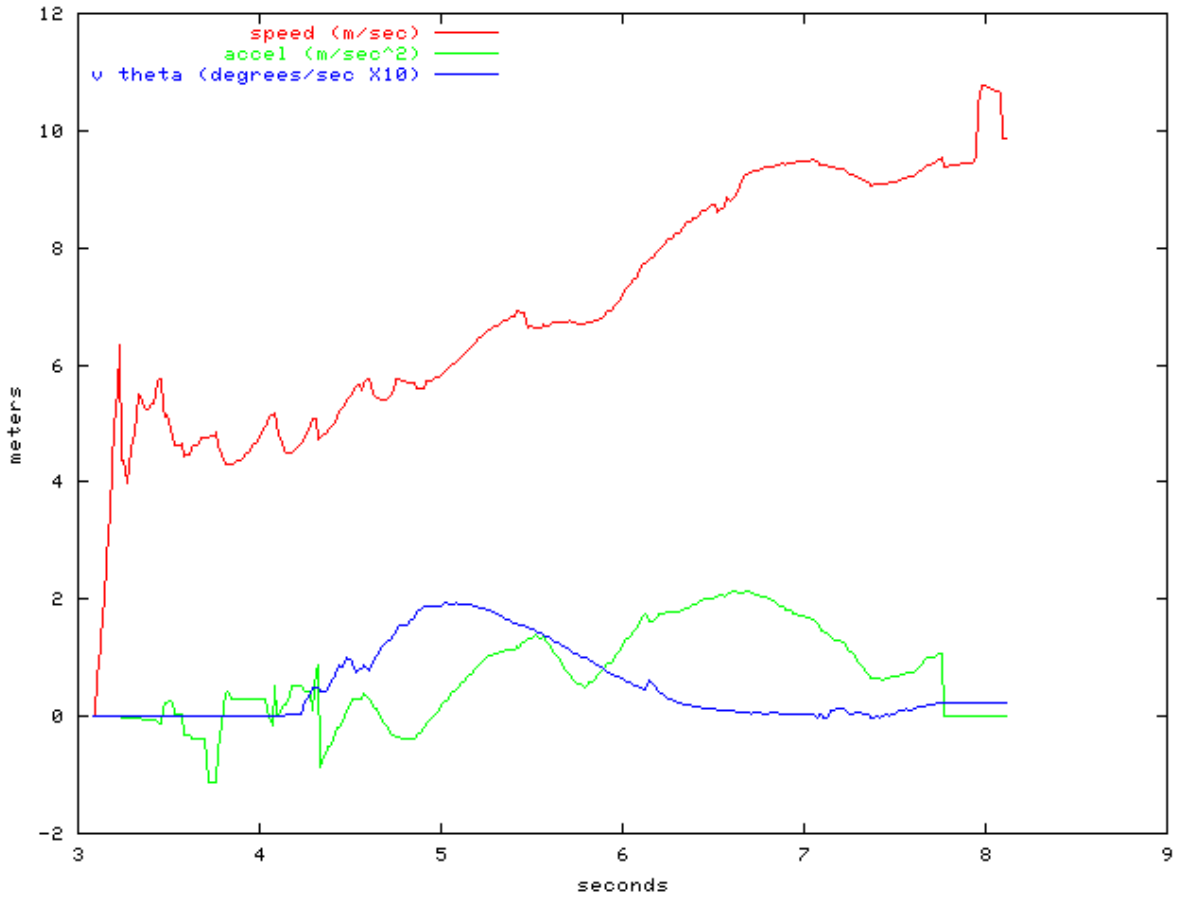


Figure 39: Tracker dynamics

The track is flagged as moving and valid at 4.3 seconds. After that time, the noise fluctuations in the velocity seem to be less than 0.5 meters/sec peak-to-peak. The acceleration and turn rate (v_theta) are fairly smooth, but clearly have a lot of delay.

The vehicle is probably near peak turn rate at the time that the track is detected as moving, but the peak in output turn rate happens almost a second later.

The acceleration estimate is also responding to the low frequency change in velocity, but slowly. At around 5.7 seconds it appears that the acceleration is near zero. The output estimate is dropping fairly rapidly, but doesn't make it to zero during the apparent low acceleration period.

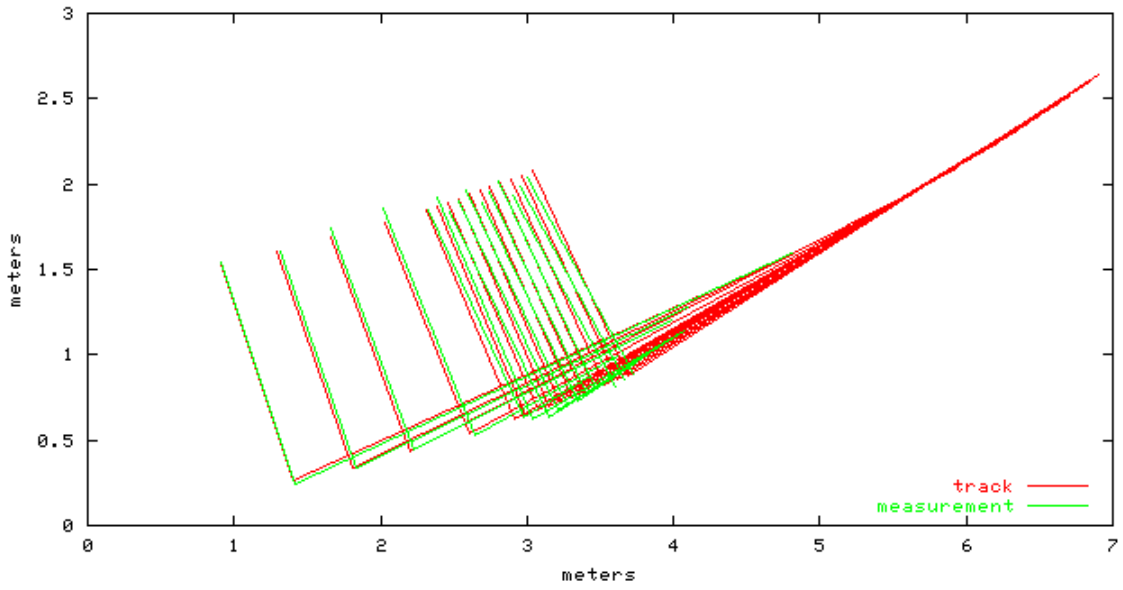


Figure 40: Track history matching

The history-based track evaluation provides another way to investigate tracker performance. By using the current motion estimate to predict the past position, then comparing with the actual measurements, we can get a sense of how well the tracker can predict motion over short periods of time (0.3 seconds.)

Though the idea of comparing the prediction to actual measurements is a good way to verify the tracker performance, this particular data is not very good evidence because of the short time scale (where the effect of acceleration and turn rate error is slight) and because the velocity correction is calculated from this very data so that it minimizes the error. It would be much more convincing to show that we can predict the future.

We can see that there is very good agreement of position, showing the velocity estimate is fairly accurate. Also, we can see that the turning approximately matches up as well. Slight acceleration error is visible in the middle of the sequence (the first order velocity correction forces both ends to line up.)

1.16.6 Summary

We have found that the linear feature tracker combined with history-based track validation is able to generate reasonably accurate velocity estimates for cars and pedestrians in a cluttered urban environment, while giving a low rate of spurious motion indications that can cause false alarms. These false alarms are discussed in detail in the Sections 6.3 titled SCWS Warning Algorithm and 6.4 False Alarms. The estimation of acceleration and turn rate appears to improve prediction of future positions, but the prediction would be significantly better if the response time could be improved.

We also have enough experience in working on this particular problem to be able to state with some confidence that any significantly simpler approach will not be able to achieve comparably low levels of false motion estimates. When the scanner is in motion, the apparent shape of objects changes, and fairly sophisticated measures are required to determine that this is not actual motion.

The computational efficiency is significantly better than some of our previous attempts at solving this problem. No large data structures such as maps are used. The average time to process one scan on a 1.2 GHz PC is 4 milliseconds. The code size is about 7000 lines of C++, and is written at a fairly high level of abstraction, so considerable further performance gains could likely be achieved if necessary.

1.17. FCWS Warning Algorithm

From 2001 to August 2003, three generations of warning algorithms were developed, with each later version being an improvement on the previous version. The current version is the third generation algorithm, which has undergone further improvements based on data analysis and driver feedback since September 2003. The features and

improvements of the third generation of algorithm are summarized in the following table and the five points below. The improvements to the third generation algorithm and its modification in order to process radar data are introduced later in the chapter.

	Object model	Bus model	Driver	Coriolis effect	Finite-size-object effect	Threat measure
1st algorithm (2001)	Free-moving	No consideration	No consideration	No consideration	No consideration	TTC (Time-to-Collision)
2nd algorithm (2002)	Non-holonomic	Non-holonomic	Empirical TTC threshold	Decoupling bus motion from sensor data	No consideration	Speed-dependent TTC
3rd algorithm (2003)	Free-moving (stopped or creeping targets) + non-holonomic (moving targets)	Non-holonomic	Empirical required deceleration threshold	Decoupling bus motion from sensor data	Delayed-filter which can well estimate acceleration from range data	Required deceleration

Table 18. Features and improvements of three generations of FCWS algorithms

The main features of the third generation algorithm are described in the following five points:

1. Model: Moving targets are modeled with non-holonomic constraints, so that heading and yaw-rate can be more precisely estimated. For stopped and creeping targets a free-moving model is used because moving direction can not be detected from short-time displacement. A free-moving model is a 2D kinematic model based on Newton's laws of motion. A non-holonomic constraint means that lateral slide is prohibited.

2. Driver's role: It is taken into account that the bus driver is working in parallel with FCWS, and responsible for fusing warning information with his/her own perception and decision making. Empirical data were analyzed to derive thresholds from the driver's behavioral data so that the FCWS can better match the driver's normal operation.
3. Coriolis effect: The algorithm decouples the bus's motion from sensor observations so that the Coriolis effect can be eliminated. The Coriolis effect introduces an imaginary component of motion due to the rotating of the sensor's coordinate frame of reference.
4. Finite-size-object effect: The finite-size effect introduces ranging error due to the size of vehicle bodies. The "delayed filter" can better estimate target velocity and acceleration by delaying the update of the model to improve the displacement-to-error (signal-to-noise) ratio that is usually impaired by finite-size effect.
5. Threat measure: Required deceleration is used as threat measure. Required deceleration is the minimum deceleration that should be applied to the bus to avoid an imminent collision with a preceding object. TTC (Time-to-Collision), speed-dependent TTC (which is a look-up-table of empirical TTC derived from real data indexed by object speed and bus speed) were also tried as threat measures, but required deceleration is more natural in terms of matching with the driver's operation. It is the delayed filter which makes it possible to utilize the required deceleration as the threat measure.

1.17.1 FCWS Algorithm structure

The structure of the warning algorithm is shown below. The main data structure-track file, the tracking and warning detection algorithm are described in detail at the end of this section.

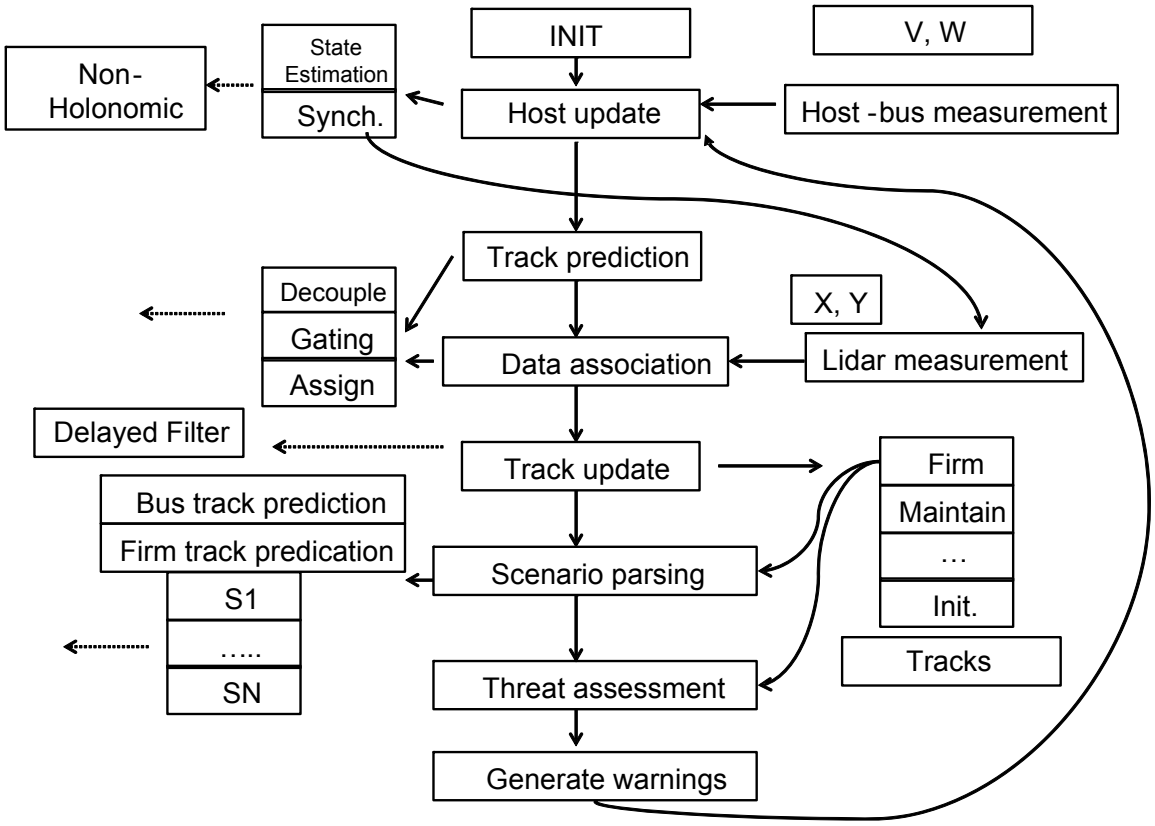


Figure 41 FCWS Algorithm structure

1.17.2 FCWS Data structure

1.17.2.1 Track file structure

A track file is a list of tracks being processed. Each track is a correlation and refinement of the time sequence of observations of an object (a target). An ID indexes each track using the name (usually an integer) for an object under tracking. An object under tracking is described by the object state in the track file. Object state is a combination of kinematic states and track properties of an object.

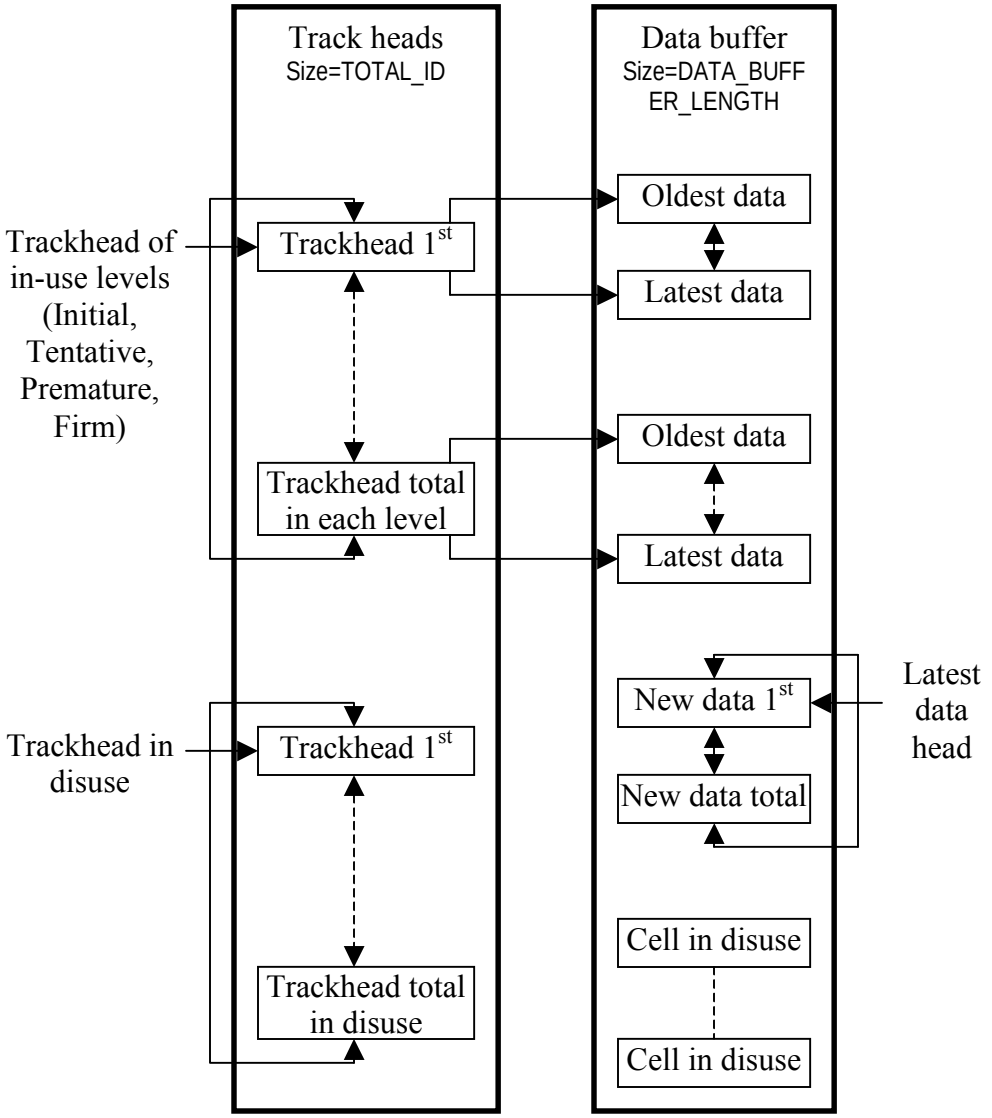


Figure 42 FCWS Track file structure

The designed track file consists of two major memory buffers: track head buffer and data buffer. Both are declared as linear arrays but are organized in linked-lists. Every track head cell belongs to one of the following five levels:

LEVEL_DISUSE: currently not in use;

LEVEL_INITIAL: initial tracks;

LEVEL_TENTATIVE: tentative tracks;

LEVEL_PREMTURE: premature tracks;

LEVEL_FIRM: firm tracks.

Each level of track heads is organized as a double-linked list. Each track head is then linked to a double-linked list of historical data of the track. The whole data buffer is organized as a circular queue (or equivalently a First-In-First-Out (FIFO)). If the head of the queue reaches the tail, the oldest cells are released from double-links of tracks to provide memory for new data. New data collected in the latest snapshot are saved in a double-linked list. The structure of track file is illustrated in the above picture.

1.17.2.1.1 Track head data structure

One key element of a track file is the data structure for storing track information. This is defined in the TRACK_HEAD structure;

```
typedef struct
{
        int                 Prev,ID,Next,Count,Level;
        OBJECT_STATES Pred;
} TRACK_HEAD;
```

where ID is the index of a track, Pred is the predicted state.

1.17.2.1.2 Object *state data structure*

Another key element of a track file is the data structure for storing data of an object in one snapshot. This is defined in the OBJECT_STATES structure:

```
typedef  struct {
        double   t[TOTAL_OBJST_T_MBRS];
        int      pntr[TOTAL_OBJST_PNTR_MBRS];
        int      stat[TOTAL_OBJST_STAT_MBRS];
        int      iobsv[TOTAL_OBJST_IOBSV_MBRS];
        double   dobsv[TOTAL_OBJST_DOBSV_MBRS];
        double   par[TOTAL_OBJST_PAR_MBRS];
} OBJECT_STATES;
```

where t[] are time members (e.g. time of availability of data and estimated time for filtering output considering the delays); pntr[] are pointer members for data structure manipulation (e.g. building up linked lists); iobsv[] and dobsv[] are observation members for raw data storage; par[] are estimated motion states members for refined parameters storage; the identifiers in brackets are constants.

This structure is easily extendable. The usage of members is defined in the program and is subject to change. Currently the constants are:

```
TOTAL_OBJST_T_MBRS = 2;
TOTAL_OBJST_PNTR_MBRS = 16;
TOTAL_OBJST_STAT_MBRS = 2;
TOTAL_OBJST_IOBSV_MBRS = 10;
TOTAL_OBJST_DOBSV_MBRS = 10;
TOTAL_OBJST_PAR_MBRS = 10;
```

1.17.2.1.3 Linked lists of tracks

Tracks are categorized into four levels: initial, tentative, premature and firm (see section 1.17.3.3 and 1.17.6 for details of these levels). Each level of tracks is organized in a double-linked list. The sub-routine "ChangeTrackLevel()" can move a track from one level to another. Upon initialization, all track heads are put in "disuse" category. Sub-routine "FreeTrack()" can move a track from any level to disuse.

1.17.2.1.4 Linked lists of track histories

The historical data of tracks are built in double-linked lists. Each node is an OBJECT_STATES structure. The following figure shows a typical linked list.

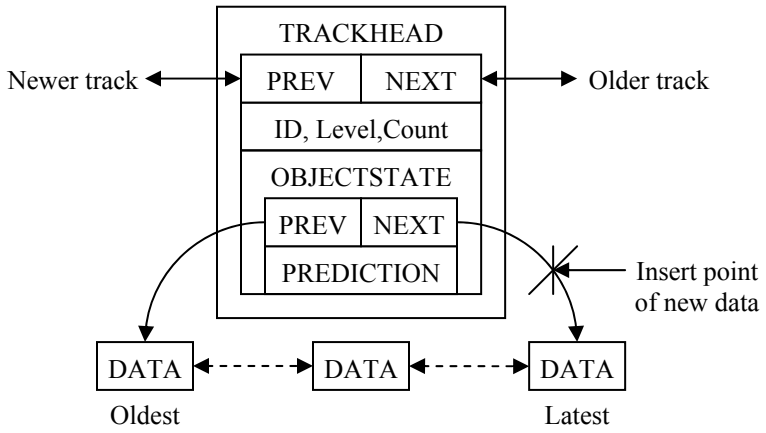


Figure 43 Linked list of tracks and historical data

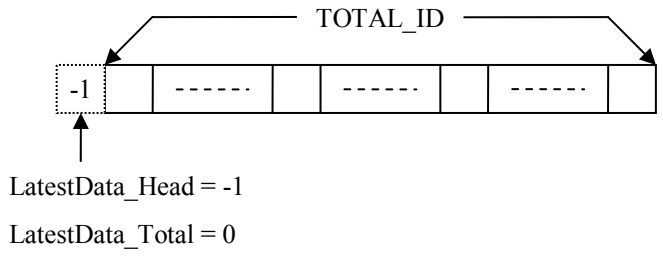
where ID of the track is saved in DATA set of each node.

Sub-routine “AssignData2Track()” and “FreeHistoricalData()” can add a data cell into or remove a data cell from the historical data list.

1.17.2.1.5 Object state FIFO buffer

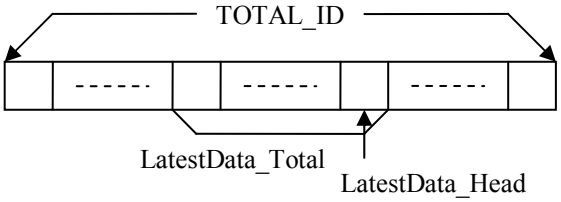
The object state buffer is a FIFO structure. Each entry is an OBJECT_STATES structure. The head pointer of the FIFO is the “LatestData_Head”. There is no tail pointer. Total number of targets detected in the latest snapshot is saved in LatestData_Total.

Upon initialization, each entry of the FIFO is released. LatestData_Head is set to -1. LatestData_Total is set 0.



Object state buffer initialization

When new observations come in, they are going to be pushed into the FIFO. The old entries that are going to be replaced are released first.



Object state buffer update

When an old entry is going to be released, its previous entry’s pointer to the next entry (the one that is going to be released) is set to NULL. When a new entry is assigned to a track, the pointer in the track is updated to point to the latest entry, and the former no.1 entry becomes the 2nd entry.

Sub-routine “NewSensorMeasurement()” can put new sensor measurements into object state buffer.

1.17.2.1.6 Host-vehicle state structure

Host-vehicle data are saved in a separate data structure. It is defined in “HOST_STATE”:

```
typedef struct
{
        OBJECT_STATES fifo[HOST_BUFFER_LENGTH];
        int LatestData_Head;
        int LatestData_Total;
        int Ready;
} HOST_STATE;
```

Constant HOST_BUFFER_LENGTH is currently 10. The data structure is organized as a FIFO very similar to the object state buffer. The variable “Ready” is used to indicate that the buffer is filled with data.

1.17.2.2 Variable allocation

It is important to note that data are saved in OBJECT_STATE structure. Variable allocations of host vehicle and objects are different.

Memory	Variable	Comment
dobsv[0]	$\widetilde{V}$	speed measurement
dobsv[1]	$\widetilde{\omega}$	yaw-rate measurement
par[0]	x	x position in ground frame of reference
par[1]	y	y position in ground frame of reference
par[2]	v	forward speed
par[3]	ω□	yaw-rate
par[4]	A	Heading
par[5]	al	forward acceleration
par[6]	Aω	angle acceleration
par[7]	CosT	cosine of rotation angle
par[8]	SinT	sine of rotation angle

Table 19. FCWS Host vehicle state variable allocation

Memory	Variable	Comment
dobsv[0]	$\tilde{x}$	decoupled x position in ground frame of reference
dobsv[1]	$\tilde{y}$	decoupled y position in ground frame of reference
dobsv[2]	L	Lateral position measurement
dobsv[3]	R	Longitudinal position measurement
par[0]	x	x position in ground frame of reference
par[1]	Y	y position in ground frame of reference
par[2]	vx	x component of velocity
par[3]	vy	y component of velocity
par[4]	V	forward speed
par[5]	A	heading
par[6]	Al	forward acceleration
par[7]	ω	yaw-rate
par[8]	CosT	cosine of heading angle
par[9]	SinT	sine of heading angle

Table 20. FCWS Object state variable allocation

1.17.3 FCWS Tracking algorithm

Data association for tracking is the process to determine the correlation between observation-track pairs, i.e. to assign observations to existing tracks, update them and extend the tracks. To reduce computations, data association is usually done in two steps: gating and assignment. Gating is a coarse association process, which removes unlikely correlations. Assignment is a fine association process, which determines the correlations.

1.17.3.1 Association metrics

An association metric is a measure of distances between observation-track or observation-observation pairs. An association metric must satisfy the following three criteria:

Distinguishability: Given any two entities *a* and *b*, the distance between them must satisfy

$d(a,b) \geq 0$

$d(a,b) = 0 \Leftrightarrow a = b$

Symmetry: Given any two entities *a* and *b*, the distance between them must satisfy

$d(a,b) = d(b,a)$;

Triangle Inequality: Given any three entities *a*, *b* and *c*, the distances between them must satisfy

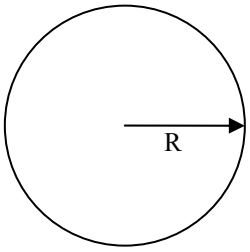
$d(a,b) + d(b,c) \geq d(a,c)$;

The normal distance measure in 2D space (*x*,*y*) is:

$$d(a,b) = |a-b| = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

The corresponding gate is a circle:

$\sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} = R$, *R* is the radius of the gate.



Another distance measure in 2D space is:

$$d(a,b) = \max\{|x_a - x_b|, |y_a - y_b|\}$$

where (x_a, y_a) and (x_b, y_b) are coordinates of entities *a* and *b* in 2D space. The properties of absolute value operation immediately satisfy criteria 1 and 2. To prove that $d(a,b)$ is a valid distance measure, we only need to verify the triangle inequality. Because

$\max\{|x_a - x_b|, |y_a - y_b|\} \geq |x_a - x_b|$

$\max\{|x_b - x_c|, |y_b - y_c|\} \geq |x_b - x_c|$

hence,

$\max\{|x_a - x_b|, |y_a - y_b|\} + \max\{|x_b - x_c|, |y_b - y_c|\} \geq |x_a - x_b| + |x_b - x_c| \geq |x_a - x_c|$;

and similarly

$$\max\{|x_a - x_b|, |y_a - y_b|\} + \max\{|x_b - x_c|, |y_b - y_c|\} \geq |y_a - y_b| + |y_b - y_c| \geq |y_a - y_c|$$

we then have

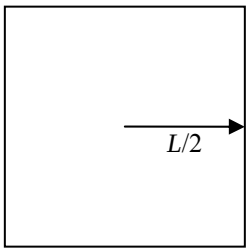
$$\max\{|x_a - x_b|, |y_a - y_b|\} + \max\{|x_b - x_c|, |y_b - y_c|\} \geq \max\{|x_a - x_c|, |y_a - y_c|\},$$

that is

$$d(a,b) + d(b,c) \geq d(a,c).$$

The corresponding gate is a square:

$\max\{|x_a - x_b|, |y_a - y_b|\} = L/2$, L is the side length of the square.

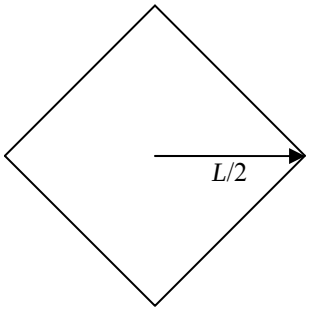


An even simpler distance measure in 2D space is:

$$d(a,b) = |x_a - x_b| + |y_a - y_b|$$

The corresponding gate is a square rotated by $\pi/4$:

$|x_a - x_b| + |y_a - y_b| = L/2$.



The latter two measures are computationally simpler and appropriate for gating. The former is more precise and appropriate for assignment.

1.17.3.2 Data association

1.17.3.2.1 Gating

Gating is the process prior to assignment to remove unlikely correlations between observation-track pairs.

First of all, we calculate the distances between observation-track pairs using simpler distance measures, which form a matrix.

Obser-vations	Tracks			
	1	----------	*k*	---------- K
1	*d*(1,1)	----------		*d*(1,K)
n			*d*(*n*,*k*)	
N	*d*(N,1)	----------		*d*(N,K)

where K is the total number of tracks, N is the total number of observations, $d(n,k)$ is the distance between observation *n* and track *k*.

For an observation-track pair (n,k), use $n \propto k$ to denote the relationship that observation *n* is inside the gate of track *k*; use $n \overline{\propto} k$ to denote the relationship that observation *n* is outside the gate of track *k*.

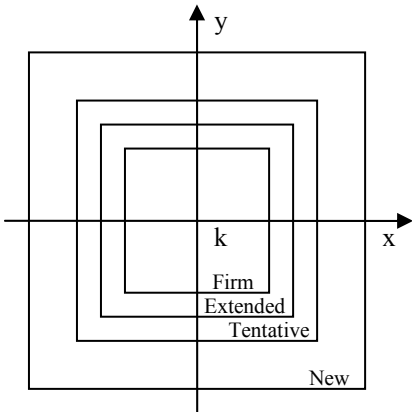
Gating is a binary hypothesis-testing process. The two hypotheses are:

$$\begin{cases} H_0 & n \overline{\propto} k \\ H_1 & n \propto k \end{cases}$$

The gating criteria are:

$$d\left(n,\hat{k}\right) \underset{H_1}{\overset{H_0}{\gtrless}} \sigma_R + \sigma_v \cdot T \,,$$

where T is the time from last update of the track to the moment of observation, $\hat{k}$ denotes the prediction of the track to the moment of observation, σ_R is the range error threshold, σ_v is the speed error threshold. Temporary settings of the thresholds are: $\sigma_R = 3m$, σ_v for firm, premature, tentative and initial tracks are $5m/s$, $8m/s$, $15m/s$ and $30m/s$ respectively. The relationship among these thresholds can be seen below.



1.17.3.2.2 Assignment

One observation may fall in the gates of multiple tracks. Multiple observations may fall in the same gate of a track. Assignment is the process to resolve the ambiguities. Co-existent tracks may be at different tracking levels. The tracking levels, from lower to higher, are initial, tentative, premature and firm. These levels also represent the growing-up stages of tracks.

To simplify the assignment process, we make the following assumptions:

1. *Higher-level priority*: if an observation can be assigned to multiple tracks at different stages, higher level tracks should be given higher priority.
2. *Higher-level-track uniqueness*: if an observation is assigned to a higher-than-initial-level track, neither should it be assigned again to any lower-level tracks, nor should it be assigned to another same-level track;
3. *Initial track non-uniqueness*: if an observation is not assigned to any higher-level track, it may be assigned to multiple initial tracks;
4. *Observation non-uniqueness*: multiple observations may be assigned to the same track.

5. *False alarm ignore*: if an observation cannot be assigned to any existing tracks, it should be set as the start point of an initial track, in other words, it should not be treated as a false alarm anyway.

The assignment criterion is nearest neighbor:

For observation *n* and track $k \in K$, where *K* is the set of tracks of the same level all with *n* falling in their gates, if

$\forall\ k_K \in K$, $\left|n - \hat{k}_K\right| \geq \left|n - \hat{k}\right|$

then we assign *n* to *k*, and track *k* is called the nearest neighbor of observation *n*. Use $n \Rightarrow k$ to denote the assignment of observation *n* to track *k*. The assignment criterion can be expressed as:

if $\left|n - \hat{k}\right| = \min_{n \propto k_K}\left\{\left|n - \hat{k}_K\right|\right\}$, then $n \Rightarrow k$.

The association process begins with firm tracks and proceeds to lower levels step by step.

1.17.3.2.3 Observation to firm track association

For a firm track, the prediction algorithm is described in section 1.17.6.2.5.

For an observation-firm track pair (n,k), if the following conditions are satisfied:

$n \propto k$ (gating),

$\left|n - \hat{k}\right| = \min_{n \propto k_f}\left\{\left|n - k_f\right|\right\}$, k_f are firm tracks.

then $n \Rightarrow k$, and *n* is removed from observation list.

1.17.3.2.4 Observation to premature track association

For a premature track, the prediction equation is:

$$\hat{k} = k_3 + \frac{k_3 - k_1}{t_3 - t_1}(t - t_3).$$

For an observation-premature track pair (n,k), if the following conditions are satisfied:

$n \propto k$ (gating),

$\forall k_f, n \overline{\propto} k_f$, k_f are firm tracks,

$\left|n - \hat{k}\right| = \min_{n \propto k_p} \left\{ \left| n - k_p \right| \right\}$, k_p are premature tracks.

then $n \Rightarrow k$, and n is removed from observation list.

1.17.3.2.5 Observation to tentative track association

For a tentative track, the prediction equation is:

$$\hat{k} = k_2 + \frac{k_2 - k_1}{t_2 - t_1}(t - t_2).$$

For an observation-tentative track pair (n, k), if the following conditions are satisfied:

$n \propto k$ (gating),

$\forall k_f, n \overline{\propto} k_f$, $\forall k_e, n \overline{\propto} k_e$, k_f and k_e are firm and premature tracks respectively,

$\left|n - \hat{k}\right| = \min_{n \propto k_t} \left\{ \left| n - k_t \right| \right\}$, k_t are tentative tracks.

then $n \Rightarrow k$, and n is removed from observation list.

1.17.3.2.6 Observation to initial track association

For an initial track, there is no way to predict.

For an observation-initial track pair (n, k), if the following conditions are satisfied:

$n \propto k$ (gating),

$\forall k_f, n \overline{\propto} k_f$, $\forall k_p, n \overline{\propto} k_p$, $\forall k_t, n \overline{\propto} k_t$, k_f, k_p and k_t are firm, premature and tentative tracks respectively,

then $n \Rightarrow k$. If n is assigned to at least one initial track, it is removed from observation list.

1.17.3.2.7 Unresolved observations

If an observation cannot be assigned to any existing tracks, it starts an initial track.

1.17.3.3 Track update

1.17.3.3.1 Firm track update

After association, a firm track may result in one of the four outcomes:

1. It is kept firm and updated with only one new observation for only one new observation is assigned in (without ambiguity);
2. It is kept firm and updated with the average of multiple new observations as more than one new observations are assigned in (with ambiguity);
3. It is kept firm but not updated due to lack of a new observation (maintained);
4. It is dropped out as being maintained for a certain period (e.g. 3sec) due to lack of new observations (out of date).

1.17.3.3.2 Premature track update

After association, a premature track may result in one of the three outcomes:

1. It is upgraded to firm (successful initiation) and updated with only one new observation for only one new observation is assigned in (without ambiguity);
2. It is upgraded to firm and updated with the average of multiple new observations as more than one observation are assigned in (with ambiguity);
3. It is downgraded to tentative by removing the oldest point due to lack of a new observation so that it can be put in the tentative category to be tested in association again.

1.17.3.3.3 Tentative track update

After association, a tentative track may result in one of the three outcomes:

1. It is upgraded to premature and updated with only one new observation for only one new observation is assigned in (without ambiguity);
2. It is upgraded to premature and updated with the average of multiple new observations as more than one observations are assigned in (with ambiguity);
3. It is downgraded to new by removing the oldest point due to lack of a new observation so that it can be put in the new-track category to be tested in association again.

1.17.3.3.4 Initial track update

After association, an initial track may result in one of the three outcomes:

1. It is upgraded to tentative and updated with only one new observation for only one new observation is assigned in (without ambiguity);
2. It is split into multiple tentative tracks and updated with each of multiple new observations as more than one observations are assigned in (with ambiguity);
3. It is treated as a false alarm and removed from the track list due to lack of a new observation.

It should be noted that one observation may be assigned to multiple initial tracks.

1.17.3.3.5 Initial track initiation

If an observation cannot be assigned to any existing tracks, it initiates an initial track. A new ID is allocated to the initial track.

1.17.4 FCWS Host vehicle state estimation

Host vehicle state observations are longitudinal wheel speed and yaw-rate. Host vehicle model is a nonholonomic bicycle model.

1.17.4.1 Nonholonomic constraint and kinematic model

Nonholonomic constraint means the wheels cannot move sideways. We choose the center point of the rear axle as the reference point of the vehicle body. The nonholonomic bicycle model is illustrated in the following figure, where θ is front wheel turning angle, L is the wheel-base, v is longitudinal speed, R is the turning radius, C is the curvature.

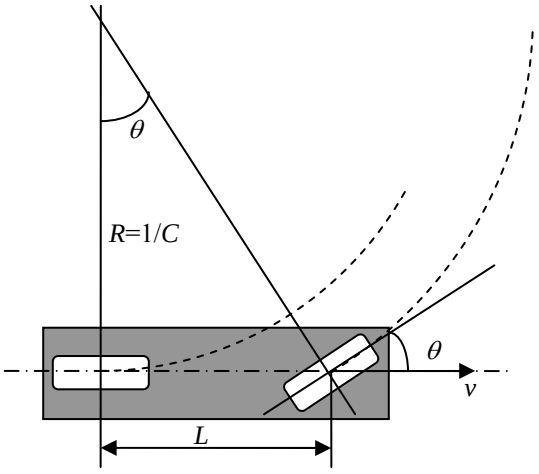


Figure 44 Non-holonomic bicycle model

We have the following equations immediately from the geometry in the sense of nonholonomic constraint:

$$C = \frac{1}{R} = \frac{\tan(\theta)}{L}.$$

And yaw-rate ω would be:

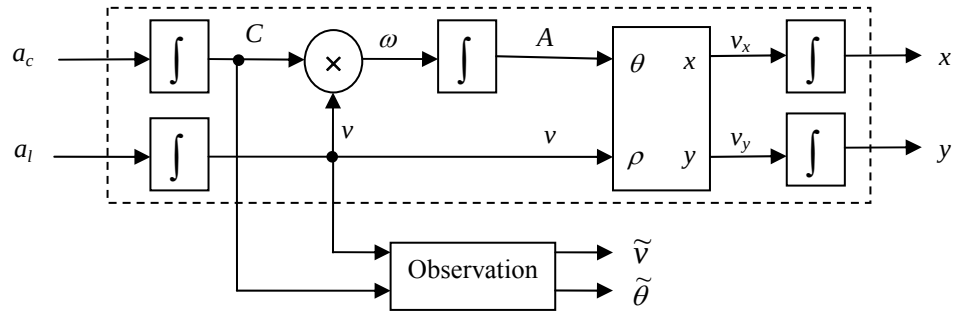
$$\omega = vC.$$

The host vehicle kinematic model with nonholonomic constraint is:

$$\begin{cases} \dot{x} = v \cdot \cos(A) \\ \dot{y} = v \cdot \sin(A) \\ \dot{A} = \omega = v \cdot C \\ \dot{C} = a_c \\ \dot{v} = a_l \end{cases}$$

where (x,y) is position of vehicle's reference point in ground coordinate frame, A is vehicle's heading angle in ground coordinate system, a_l and a_c are driver inputs for adjusting longitudinal speed and yaw rate.

This model can be illustrated below:



The observation model is:

$$\begin{cases} \tilde{v} = v + n_v \\ \tilde{\theta} = \tan^{-1}(LC) + n_\theta \end{cases}$$

where n_v and n_θ are noise components.

The polarity of θ is defined as such that it is positive for left-turn and negative for right-turn. According to this definition, roads curving left have positive curvature, while roads curving right have negative curvature. We use $\tilde{v}$ and $\tilde{C}$ to denote speed and curvature converted from observations hereafter

1.17.4.2 Model initialization

To initialize the model, K (K>1) steps of observations need to be collected. K is adjustable to compensate the object sensor delays so that the host vehicle data can be synchronized with object sensor data.

$$\begin{cases} x(0) = 0 \\ y(0) = 0 \\ v(0) = \tilde{v}_0 \\ C(0) = \tilde{C}_0 \\ A(0) = \pi / 2 \\ a_l(0) = (\tilde{v}_{K-1} - \tilde{v}_0)/(t_{K-1} - t_0) \\ a_c(0) = (\tilde{C}_{K-1} - \tilde{C}_0)/(t_{K-1} - t_0) \end{cases}$$

where $\tilde{v}_0$ is the initial wheel speed measurement, $\tilde{C}_i = \tan(\tilde{\theta}_i)/L$ is the curvature from observation, $\tilde{\theta}_i$ is the front wheel angle measurement.

1.17.4.3 Prediction of observations

$$\begin{cases} \hat{v}(k+K) = v(k) + a_l(k) \cdot (t_{k+K} - t_k) \\ \hat{C}(k+K) = C(k) + a_c(k) \cdot (t_{k+K} - t_k) \end{cases}$$

1.17.4.4 Parameter estimation

$$\begin{cases} v(k+1) = v(k) + a_l(k) \cdot (t_{k+1} - t_k) \\ C(k+1) = C(k) + a_c(k) \cdot (t_{k+1} - t_k) \\ A(k+1) = A(k) + [C(k+1) \cdot v(k+1) + C(k) \cdot v(k)] \cdot (t_{k+1} - t_k)/2 \\ x(k+1) = x(k) + [v(k) \cdot \cos(A(k)) + v(k+1) \cdot \cos(A(k+1))] \cdot (t_{k+1} - t_k)/2 \\ y(k+1) = y(k) + [v(k) \cdot \sin(A(k)) + v(k+1) \cdot \sin(A(k+1))] \cdot (t_{k+1} - t_k)/2 \end{cases}$$

1.17.4.5 Model update

$$\begin{cases} a_l(k+1) = \alpha \cdot a_l(k) + \beta \cdot [\tilde{v}_{k+K} - v(k)]/(t_{k+K} - t_k) \\ a_c(k+1) = \alpha \cdot a_c(k) + \beta \cdot [\tilde{C}_{k+K} - C(k)]/(t_{k+K} - t_k) \end{cases}$$

1.17.5 FCWS Motion decoupling

1.17.5.1 Coriolis effect

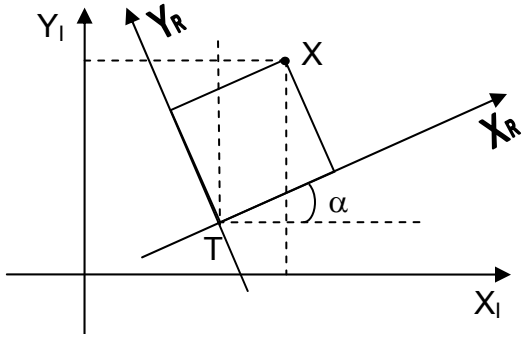
If Newton's laws of motion are used in a rotating system, a Coriolis effect appears. It introduces apparent components in the motion equations.

Let, X_I be the position of a point in an inertial system, T the coordinate of the origin of a rotating system, R the rotation matrix from the rotating system to the inertial system, X_R the observed position of the same point in the rotating system, we have

$X_I = RX_R + T$ or $X_R = R^{-1}(X_I - T)$.

where

$R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$ and $R^{-1} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix}$, see below:



Then we have

$$\frac{d}{dt}X_R = \frac{d}{dt}R^{-1}\left(X_I - T\right) + R^{-1}\frac{d}{dt}\left(X_I - T\right)$$

where

$$\frac{d}{dt}R^{-1} = \begin{bmatrix} -\sin\alpha & \cos\alpha \\ -\cos\alpha & -\sin\alpha \end{bmatrix} \cdot \omega = R^{-1}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \omega$$

ω is the yaw rate of the host vehicle.

Let

$$V_C = \omega\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\left(X_I - T\right),\ V_I = \frac{d}{dt}X_I,\ V_T = \frac{d}{dt}T, \text{ and } V_R = \frac{d}{dt}X_R,$$

then

$V_R = R^{-1}\left(V_C + V_I - V_T\right)$ or $V_I - V_T = RV_R - V_C$.

When $\omega = 0$, $V_C = 0$, the relative speed observed in the inertial frame is equal to the speed observed in the rotating frame rotated by the rotation matrix. When $\omega \neq 0$, $V_C \neq 0$, after the speed observed in the rotating frame is rotated by the rotation matrix, it is not equal to the relative speed observed in the inertial frame. There is an extra component V_C in the rotated non-inertial observation. This is the component caused by Coriolis effect.

1.17.5.2 Decoupling algorithm

The problem could be solved by means of augmented state-space modeling which involves both the states of the target and the state of the host vehicle (sensor platform). However the augmented model is computationally complex. To simplify computation, we estimate the rotation matrix and position of the host vehicle separately, then the results are used as known to estimate the states of the target. Estimation of host vehicle states is

described in section1.17.4. From the states of host vehicle, the rotation matrix and the position of host vehicle are known as:

$$R(k)=\begin{bmatrix}\cos A(k) & -\sin A(k)\\ \sin A(k) & \cos A(k)\end{bmatrix}$$

$T(k)=\begin{bmatrix}x(k)\\ y(k)\end{bmatrix}$, V_T is the observation.

$X_I(k)=R(k)X_R(k)+T(k)$, $X_R(k)$ is the sensor observation.

We can now use X_I as observation for target state estimation. In this decoupling algorithm, we have used the initial position and orientation of the host vehicle as the origin and orientation of the reference inertial frame.

1.17.6 FCWS Target state estimation

1.17.6.1 Kinematic model

The kinematic model for a free-moving object in 2D space is:

$$\begin{cases}\dot{x}=v_x\\ \dot{y}=v_y\\ \dot{v}_x=a_x\\ \dot{v}_y=a_y\end{cases}$$

The kinematic model for a vehicle-like target with nonholonomic constraint (see section 1.17.4.1) is:

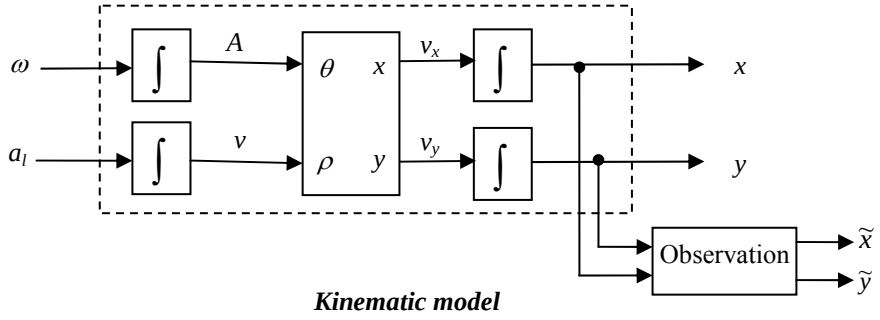
$$\begin{cases}\dot{x}=v\cdot\cos(A)\\ \dot{y}=v\cdot\sin(A)\\ \dot{A}=\omega\\ \dot{v}=a_l\end{cases}$$

The relationships between the two models are:

$$\begin{cases}v_x=v\cdot\cos(A)\\ v_y=v\cdot\sin(A)\\ a_l=a_x\cos(A)+a_y\sin(A)\\ a_c=-a_x\sin(A)+a_y\cos(A)=v\dot{A}\end{cases}$$

In these models, (*x*,*y*) is object's position, *A* is heading angle and (v_x, v_y) is velocity, all in ground coordinate system; v is longitudinal speed, a_l is longitudinal acceleration and ω is yaw rate.

This model can be illustrated as the following:



Kinematic model

The observation model is:

$$\begin{cases} \tilde{x} = x + n_x \\ \tilde{y} = y + n_y \end{cases}$$

where n_x and n_y are noise.

Implementation in the programs may slightly vary from the equations described below, however the models behind those programs are the same.

1.17.6.2 Initialization

Target kinematic model is initialized during the initialization of the track.

1.17.6.2.1 Initial track

$$\begin{cases} x(0) = \tilde{x}_0 \\ y(0) = \tilde{y}_0 \end{cases}$$

where $\begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} = X_I = RX_R + T$. (R and T are defined in section 1.17.5.2.)

1.17.6.2.2 Tentative track

$$\begin{cases} x(0) = \tilde{x}_0 \\ y(0) = \tilde{y}_0 \\ v_x(0) = (\tilde{x}_1 - \tilde{x}_0)/(t_1 - t_0) \\ v_y(0) = (\tilde{y}_1 - \tilde{y}_0)/(t_1 - t_0) \end{cases}$$

1.17.6.2.3 Premature track

$$\begin{cases} x(0) = \tilde{x}_0 \\ y(0) = \tilde{y}_0 \\ v_x(0) = (\tilde{x}_2 - \tilde{x}_0)/(t_2 - t_0) \\ v_y(0) = (\tilde{y}_2 - \tilde{y}_0)/(t_2 - t_0) \end{cases}$$

1.17.6.2.4 Firm track first steps

$$\begin{cases} x(0) = \tilde{x}_0 \\ y(0) = \tilde{y}_0 \\ v_x(0) = (\tilde{x}_3 - \tilde{x}_0)/(t_3 - t_0) \\ v_y(0) = (\tilde{y}_3 - \tilde{y}_0)/(t_3 - t_0) \end{cases}$$

1.17.6.2.5 Prediction

$$\begin{cases} \hat{x}(k+1) = x(k) - v(k) \cdot dt \cdot \sin(A(k)) \\ \hat{y}(k+1) = y(k) + v(k) \cdot dt \cdot \cos(A(k)) \\ \hat{v}(k+1) = v(k) + a_l(k) \cdot dt \\ \hat{C}(k+1) = C(k) + a_c(k) \cdot dt \\ \hat{A}(k+1) = A(k) + C(k) \cdot v(k) \cdot dt \\ \hat{a}_l(k+1) = a_l(k) \\ \hat{a}_c(k+1) = a_c(k) \end{cases}$$

1.17.6.3 Update

$$\begin{cases} v_x = [x(k+1)-x(k-T)]/[t(k+1)-t(k-T)] \\ v_y = [y(k+1)-y(k-T)]/[t(k+1)-t(k-T)] \\ v(k+1) = \sqrt{v_x^2+v_y^2} \\ A(k+1) = \arctan(v_y / v_x) - \pi/2 \\ a_l(k+1) = [v(k+1)-v(k-T)]/[t(k+1)-t(k-T)] \\ C(k+1) = \lfloor A(k+1)-A(k-T) \rfloor_{2\pi} /[t(k+1)-t(k-T)]/[v(k+1)/2+v(k-T)/2] \\ a_c(k+1) = [C(k+1)-C(k-T)]/[t(k+1)-t(k-T)] \\ x(k+1) = x(k)-[v(k)/2+v(k+1)/2]\cdot dt\cdot\sin(A(k)/2+A(k+1)/2) \\ y(k+1) = y(k)+[v(k)/2+v(k+1)/2]\cdot dt\cdot\cos(A(k)/2+A(k+1)/2) \end{cases}$$

If $k < 3T$, then

$$a_c(k+1) = \hat{a}_c(k+1);$$

if $k < 2T$, then

$$\begin{cases} a_c(k+1) = \hat{a}_c(k+1) \\ C(k+1) = \hat{C}(k+1) \\ a_l(k+1) = \hat{a}_l(k+1) \end{cases};$$

if $k < T$, then

$$\begin{cases} a_c(k+1) = \hat{a}_c(k+1) \\ C(k+1) = \hat{C}(k+1) \\ a_l(k+1) = \hat{a}_l(k+1) \\ v(k+1) = \hat{v}(k+1) \\ A(k+1) = \hat{A}(k+1) \end{cases};$$

if target is stationary, then

$$\begin{cases} a_c(k+1) = 0 \\ C(k+1) = C(k) \\ a_l(k+1) = 0 \\ A(k+1) = A(k) \end{cases}.$$

1.17.7 FCWS Threat assessment

1.17.7.1 Threat measure

The threat measure in the final version of algorithm is "required deceleration". Let a_L, v_L, a_F, v_F be deceleration (positive means decelerating) and speed of leading object and following vehicle respectively, the required deceleration can be calculated as follows.

1. When $v_F\left(\frac{v_L}{a_L}\right) \le 2\left(R + \frac{v_L^2}{2a_L}\right)$, to avoid colliding with the leading object, it is required:

$$a_F \ge \frac{v_F^2/2}{R + v_L^2/2a_L};$$

2. When $v_F\left(\frac{v_L}{a_L}\right) > 2\left(R + \frac{v_L^2}{2a_L}\right)$, to avoid colliding with the leading object, it is required:

$$a_F \ge a_L + \frac{(v_F - v_L)^2}{2R}.$$

1.17.7.2 Warning detection

1.17.7.2.1 Thresholds

The following table shows warning levels decided by thresholds and sensitivity levels. Warning level 7 is the highest level. Warning level 0 means no warning. Sensitivity is the input from the sensitivity switch that the driver can adjust.

Thresholds (m/s^2)	4.0	3.8	3.6	3.4	3.2	3.0	2.8	2.6	2.4	2.2	2.0	1.8	<1.8
Sensitivity-6	7	7	7	7	7	7	6	5	4	3	2	1	0
Sensitivity-5	7	7	7	7	7	6	5	4	3	2	1	0	0
Sensitivity-4	7	7	7	7	6	5	4	3	2	1	0	0	0
Sensitivity-3	7	7	7	6	5	4	3	2	1	0	0	0	0
Sensitivity-2	7	7	6	5	4	3	2	1	0	0	0	0	0
Sensitivity-1	7	6	5	4	3	2	1	0	0	0	0	0	0

Table 21. FCWS Sensitivity, threshold and Warning level

1.17.7.2.2 Moving objects

If the following conditions are satisfied:

1. object is in lane: |dx|<1.4m
2. object is in the same direction as bus
3. object is moving
4. object is relatively approaching: vr<0
5. bus is not turning violently: |host yaw-rate|<0.1rad/sec
6. object is decelerating: acceleration<0

In-same-lane moving object is detected. Required deceleration is calculated and compared with thresholds.

1.17.7.2.3 Stationary/stopped objects

If the following conditions are satisfied:

1. object is stopped or stationary
2. object is in lane: |dx|<1.4m
3. object is within 3.5s TTC
4. bus is not turning violently

In-same-lane stationary object is detected. Required deceleration is calculated and weighed with probability factor:

1. For stationary object, factor is 0.3
2. For stopped object, factor is 0.35.

1.17.8 Warning signal generation

Once a warning is detected, the signal sent to driver will be extended. The warning pulse patterns are defined in “WarningSignalPattern[][]”:

```
int WarningSignalPattern[WARNING_SIGNAL_LEVELS+1][WARNING_SIGNAL_LENGTH]=
{
        {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, //level 0 pattern
        {1,1,1,1,1,1,1,1,1,1,1,1,1,1,1}, //level 1 pattern
        {2,2,2,2,2,2,2,2,1,1,1,1,1,1,1}, //level 2 pattern
        {3,3,3,3,3,3,2,2,2,1,1,1,1,1,1}, //level 3 pattern
        {4,4,4,4,4,3,3,2,2,1,1,1,1,1,1}, //level 4 pattern
        {5,5,5,4,4,4,3,3,2,2,1,1,1,1,1}, //level 5 pattern
        {6,6,6,5,5,4,4,3,3,2,2,1,1,1,1}, //level 6 pattern
        {7,7,7,6,6,5,5,4,4,3,2,1,1,1,1}  //level 7 pattern
};
```

When a warning dwells longer than one snapshot, multiple warning pulses overlap. In this case, the highest pulse level (not original warning level) at current moment is displayed. For example, in four successive detection cycles, warning levels are: 7,4,6,4, then warnings displayed are:

Time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1st pulse, level 7	7	7	7	6	6	5	5	4	4	3	2	1			
2nd pulse, level 4		4	4	4	4	4	3	3	2	2	1	1	1		
3rd pulse, level 6			6	6	6	5	5	4	4	3	3	2	2	1	
4th pulse, level 4				4	4	4	4	4	3	3	2	2	1	1	1
Warnings displayed	7	7	7	6	6	5	5	4	4	3	3	2	2	1	1

Table 22. FCWS Warning display

1.17.9 FCWS Further improvement

1.17.9.1 Side recognition

In the third generation warning algorithm, if the warning is triggered from obstacles detected by the Frontal LIDAR, both passenger side and driver side DVI bars will be lit up, as is depicted in Figure 50. Please note target 145, it is the guardrail that is the warning trigger, not the vehicle on the left.

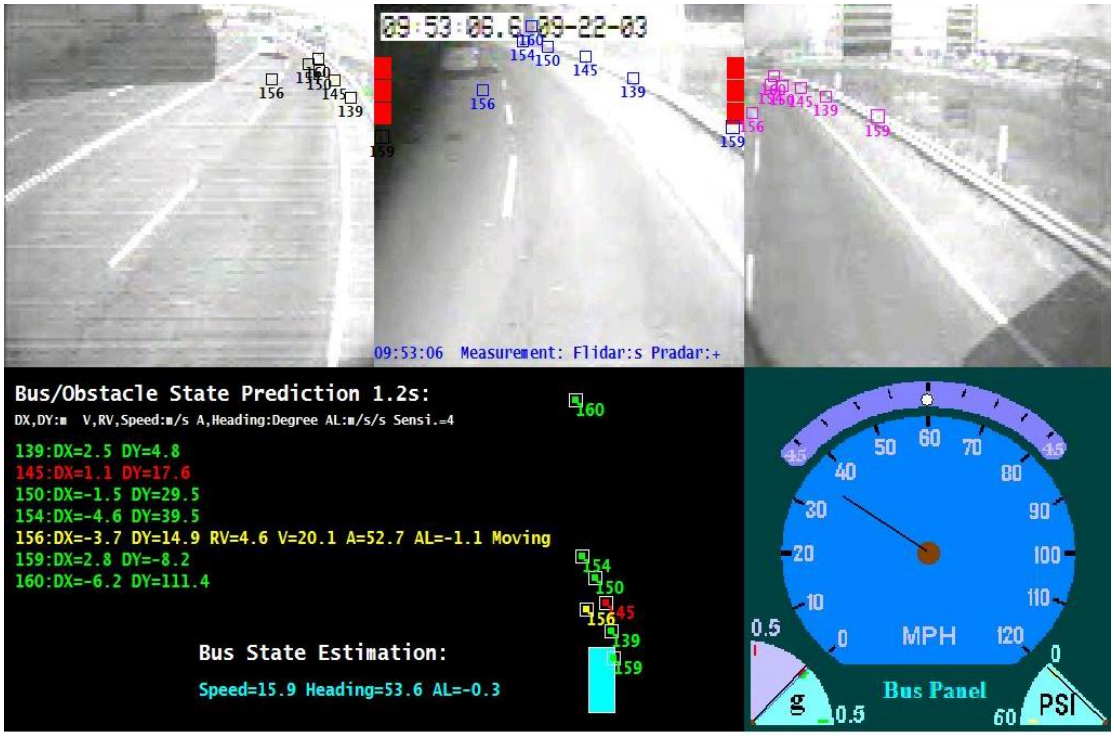


Figure 45 FCWS Warning scenario snap shot (without side recognition)

This warning was often considered a nuisance warning by drivers as they did not think that the hazard was in the path of the bus. To give another example, in a lane changing scenario, a parked car on the passenger side picked up by the frontal LIDAR may lead to a warning lit up of both the passenger side and the driver side DVI. These instances affect the credibility of the system from a driver point of view.

To address the problem, we set up a limit XM, if the lateral position of the obstacle (Dx) is greater than XM, only the passenger side DVI bar is lit up and the warning level is reduced to one if the obstacle is stationary. As shown in Figures 50 and 51, the driver might feel comfortable to the scenario below and easily figured that the guardrail is the warning trigger instead of the car on the left.

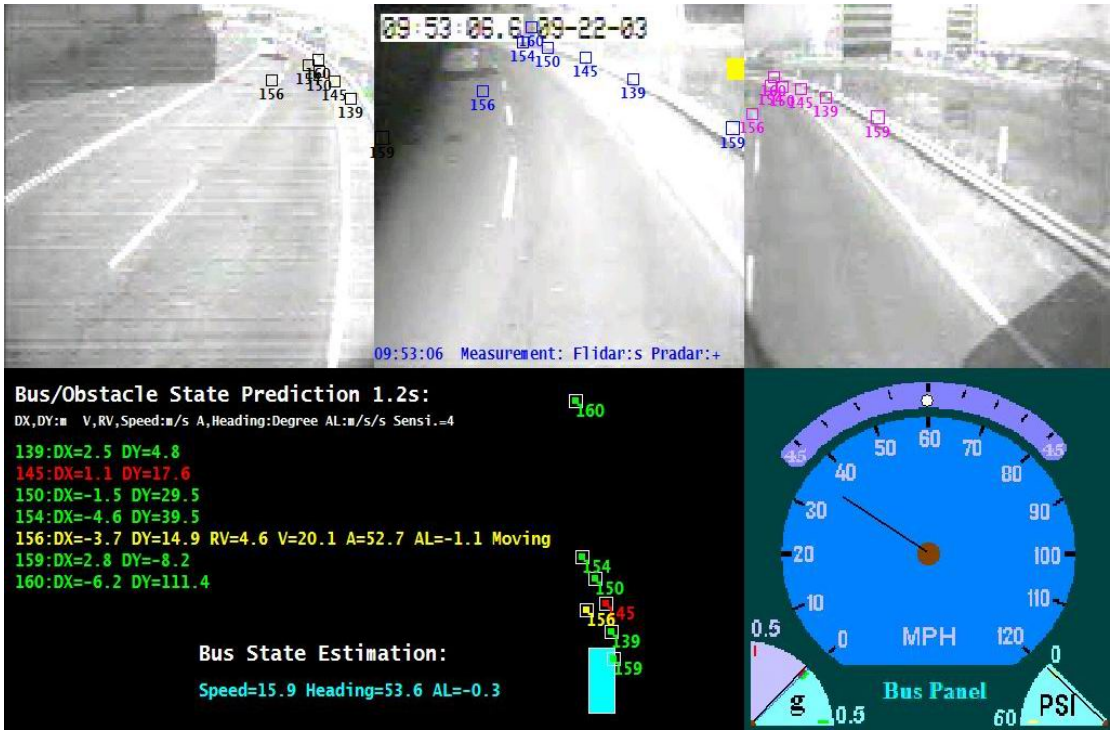


Figure 46 FCWS Warning scenario snap shot (with side recognition)

The strategy here is to turn the nuisance warnings to a friendly reminder as shown in Figure 52. Without any hardware cost, it only involves algorithm change and improves the perception of the system from the drivers point of view.

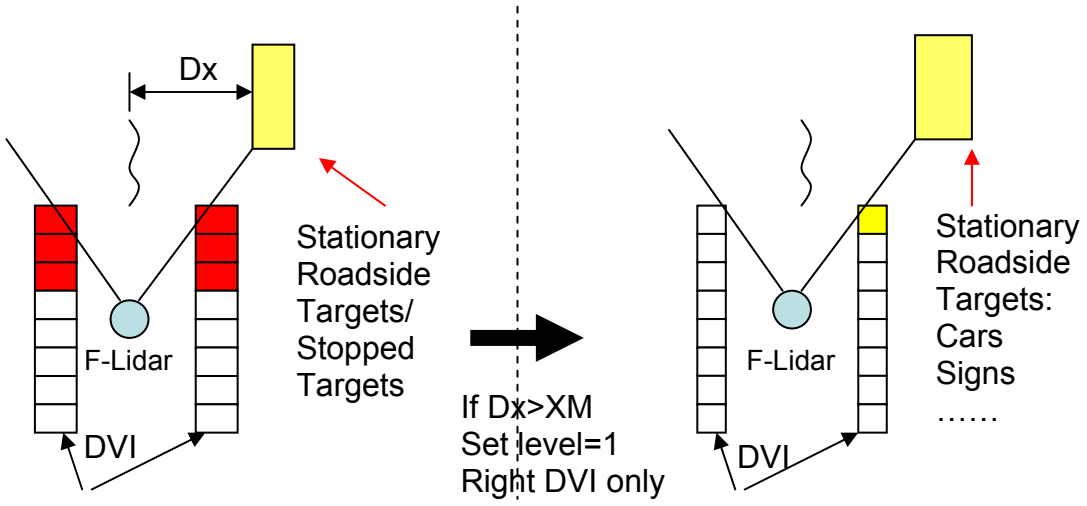


Figure 47 FCWS Strategy of side recognition

1.17.9.2 Scenario parsing and target recognition

1.17.9.2.1 Following distance constraint

One dangerous scenario, which was not accounted for in previous versions of the algorithm, is a tailgate scenario. In this scenario if the bus gets too close to the subject vehicle, for example, two vehicles maintain constant speed at 40 miles per hour, the calculated required deceleration would be almost zero which indicates no hazard and the TTC would not fall within the dangerous zone either. However since the behavior of the driver of the subject vehicle is not completely predictable, there is a chance that the leading vehicle could suddenly decelerate and the bus driver could not have enough time to avoid a collision even if the correct warning is given. Therefore, a following distance constraint was added. As long as the following distance falls within the dangerous zone which is calculated based on the bus speed, the sensitivity level and relative speed of the subject vehicle, a warning will be issued to inform the driver of the potential danger of following too close.

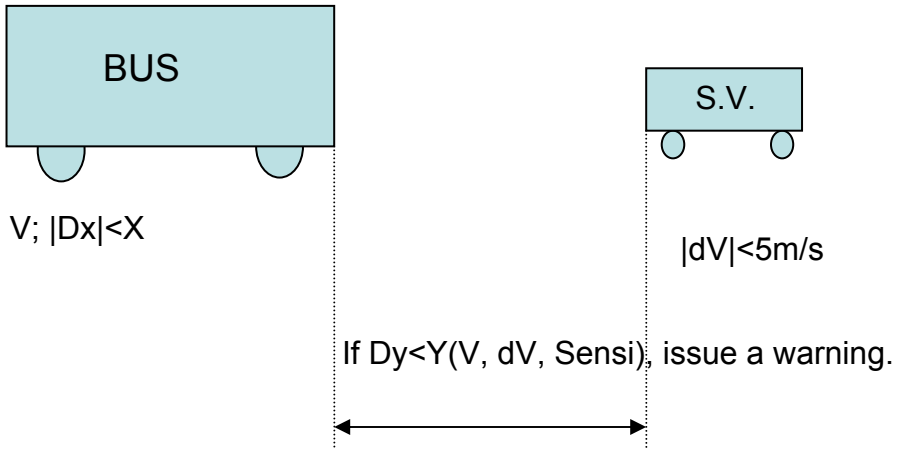


Figure 48 Following distance constraint

The figure below depicts the scenario before and after. The upper part shows the scenario and the "before" situation. The relative velocity is only -0.4m/s and the calculated required deceleration does not fall within the dangerous zone. The DVI does not light up. However, as we can see, the predicted distance between these two vehicles in 1.2 seconds

is dangerously close. After we added the following distance constraint, as shown in the lower part of the figure which is the “after” situation, the system is now going to trigger a warning as shown in red.

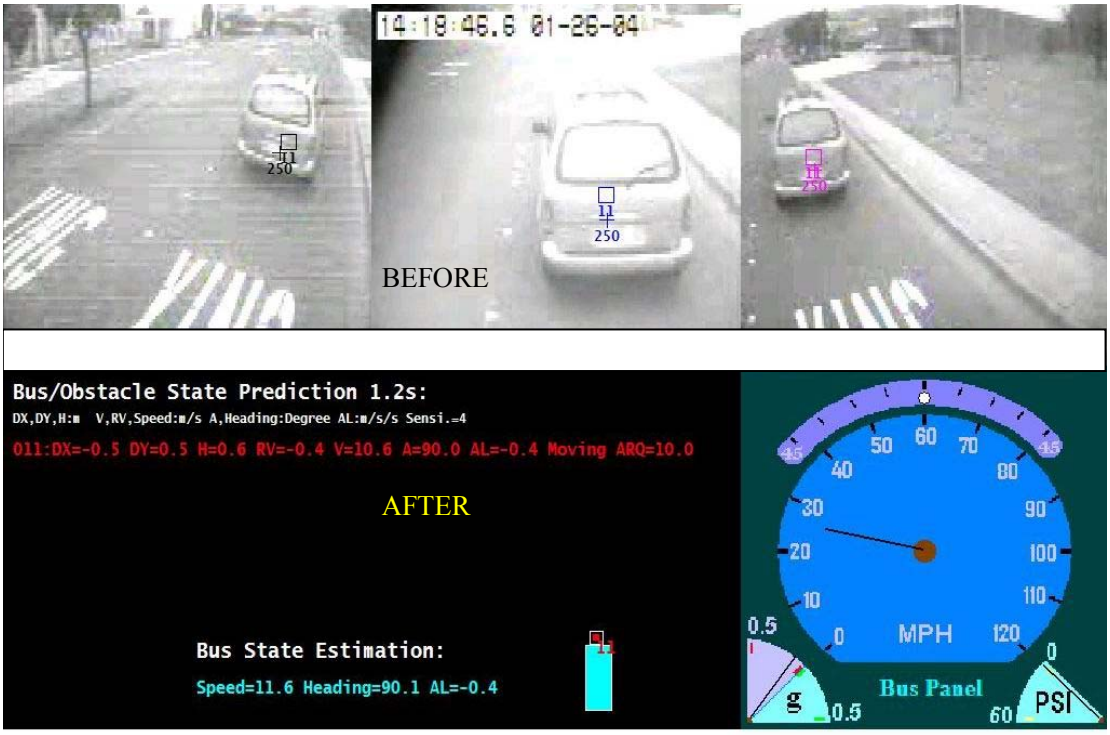


Figure 49 Following distance constraint (before and after)

1.17.9.2.2 Creeping warnings and target recognition

Another dangerous scenario which was suggested by the drivers and human factor researchers is the creeping warning. When the driver slowly follows a vehicle and then stops the bus, there is a chance that the driver gets distracted and his foot slips away from the brake pad therefore causing the bus to move slowly towards a leading vehicle without the driver’s awareness. Hence it is considered important to issue a warning under this circumstance. In order to do this, besides the range limit constraint been enforced when the bus speed is slow, the target recognition subroutine is added to tell if the target is a moving vehicle/object or not. As the tracking algorithm records the historical data of every track, a bit is set to tell the target information based on the pattern of its movement, the probability factor is increase to 0.9 when the target is recognized as a stopped vehicle.

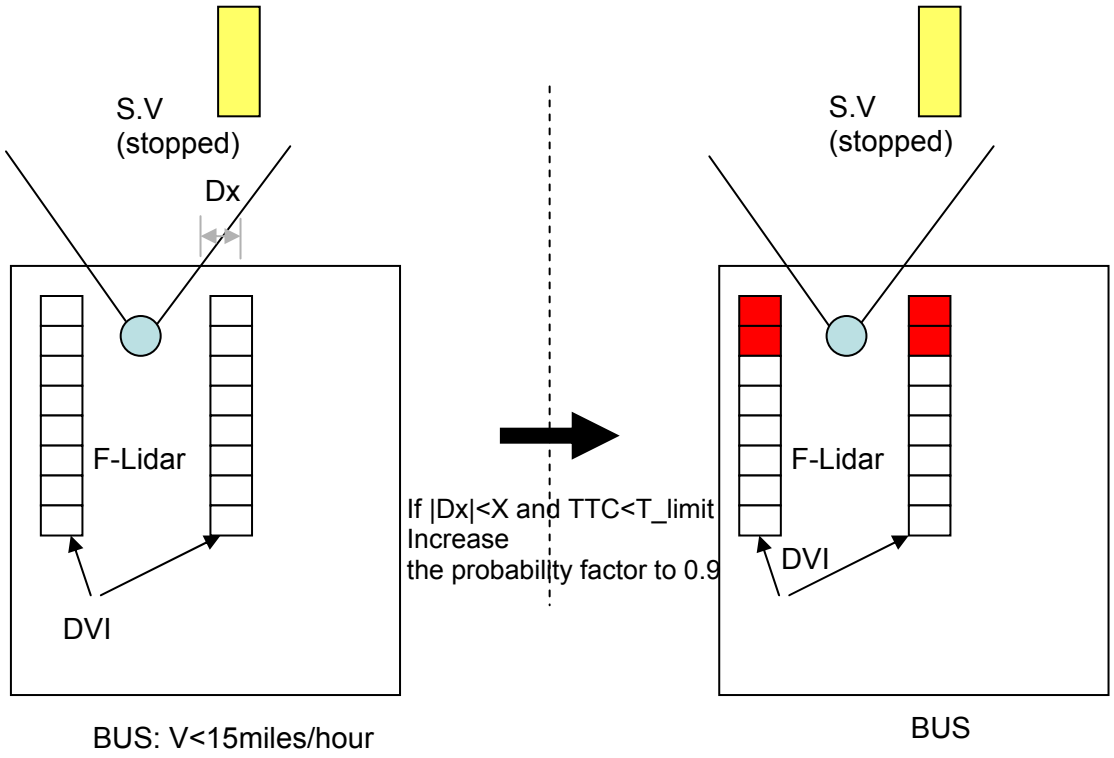


Figure 50 The creeping warning

1.17.9.3 Using RADAR data

The warning algorithm is designed for LIDAR applications but could be used for RADAR data processing as well. To cope with different weather conditions, for example, snow, rain or fog, which LIDARs may have difficulty dealing with, two microwave RADARs are installed on the bus. When the windshield wiper is turned on we assume that the weather is getting bad, the system will automatically switch to RADAR sensors. An interface subroutine is developed for RADAR data conversion. The program will take RADAR data input and convert it to LIDAR data format. That is, converting from the target distance, azimuth angle measurement of RADARs to the lateral and longitudinal position measurement of the target, and then feed the data to the warning algorithm. Although the azimuth angle resolution of the RADAR is not as good as the LIDAR, the system is now capable of working under harsh weather conditions.

1.17.10 FCWS Suggestions

To suggest more improvements of the algorithms, these points should be emphasized:

1. Transition of vehicle models - It was found that nonholonomic model is good for moving targets in terms of estimating yaw-rate and moving direction. However at lower speed, due to short displacement in processing time, it is hard to detect moving direction. In this case free moving model is better. The transition of vehicle models from higher speed to lower speed and vice versa needs to be improved.
2. Scenario parsing - This has been a topic since the beginning of the project. However it is not well resolved yet. It needs to consider the relationship among all objects and subject vehicle and infrastructure. Current algorithm only detects straight road in-lane objects, and cannot avoid false warnings due to lack of lane information and driver status.
3. Driver model - Driver's field operational data were analyzed leading to the empirical threshold settings. However more complex driver model may help to tell whether driver is attentive. Collision warning is supposed to be issued only when driver is inattentive.
4. Road geometry - Knowledge about road geometry and route could be used to eliminate false alarms triggered by road-side objects or out-of-lane objects, which could be obtained via on-vehicle detection or an AVL / map database and GPS.

1.17.11 FCWS Summary

The FCWS has been tested for the over two years time period and has been demonstrated in various occasions including 2003 National IVI event in Washington DC and General Managers' meeting in Santa Monica. The system is able to significantly suppresses false positives (unwanted warnings) but keeps high sensitivity to frontal collision scenarios. The biggest challenge for transit collision warning systems is that buses usually serve in urban/suburban environment where too many objects (guard rails, traffic signs, parked cars, etc.) may trigger false alarms. Additionally, bus drivers are very well trained experienced drivers who are less likely to run into accidents thus are very cautious with collision warnings. It is therefore a difficult problem to detect real imminent crashes and

give drivers timely warnings while suppressing excessive false alarms. The FCWS algorithm developed under this program has addressed this problem to a great extent. It is also worth noting that the target tracking and state estimation algorithms can be used for general applications. In the Intersection Collision Warning Program, the algorithm has been used for LTAP/OD (Left Turn Across Path/Opposite Direction) collision warning scenarios, without major changes.

1.18. SCWS Warning algorithm

The sensors and modules described in the previous sections provide the dynamic quantities of the bus and the observed objects and additional information about the environment. These measurements are combined with preloaded information to analyze the threat level of the situation. In the warning algorithm the system calculates the probability that a collision will occur within the next five seconds. If the probability of collision exceeds a certain threshold, an appropriate warning is displayed to the driver. In the warning algorithm for the SCWS we have two warning levels, "alert" and "imminent warning". An "alert" is displayed to the driver when the situation is somewhat dangerous, an "imminent warning" is given if the situation is dangerous enough to inform the driver in an intrusive way. A detailed description of the algorithm can be found in [11]. A short example is illustrated here.

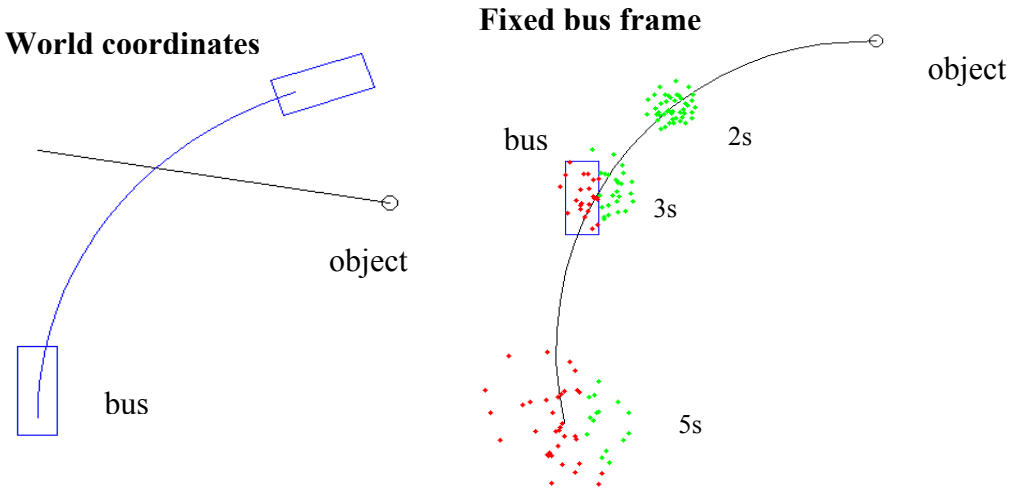


Figure 51 The trajectories of a bus and an object shown in the world coordinate frame (left) and the fixed bus frame (right). In the right figure possible positions of the object are shown for the times 2, 3, and 5 seconds in the future. Green indicates that no collision has happened; red indicates that a collision has happened.

In Figure 51 a bus turns right while an object crosses its path from right to left (World). The sensors measure the speed and turn rate of the bus and the location and velocity of the object. The algorithm calculates possible paths of the object with respect to the bus (Fixed bus). In this calculation the paths are distributed according to the uncertainties of the measured dynamic quantities as well as according to models of driver and object behavior. Next, the system determines for times up to 5 seconds into the future which fraction of these paths lead to a collision. In Figure 51 this is

[11] Mertz, C. "A 2D collision warning framework based on a Monte Carlo approach," Proceedings of ITS America's 14th Annual Meeting and Exposition. April 2004.

shown for the times 2, 3, and 5 seconds. This fraction is the probability of collision and is plotted versus time (Figure 52). This graph is divided into three areas, each a different level of threat severity. The area with the severest level that the probability of collision curve reaches determines the warning issued to the driver.

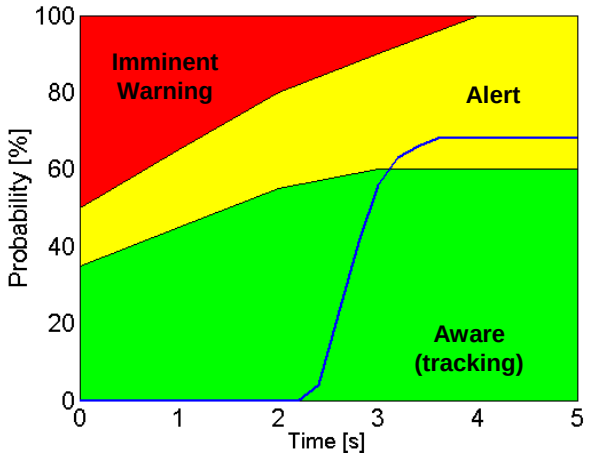


Figure 52 Probability of collision plotted versus time. The three regions correspond to the warning levels aware, alert, and imminent

The algorithm can also deal with environmental information. For example, if the object is a pedestrian and is on the sidewalk, there is an enhanced likelihood that the pedestrian will stay on the sidewalk. This is addressed by giving the paths leaving the sidewalk a lower weight.

1.18.1 Under-bus warning

Another important alarm is the under-bus warning. It is issued when a person falls and has the potential of sliding under the bus. We detect these situations by observing pedestrians who disappear while being close to the bus. The challenge in this algorithm is to distinguish people who disappear through falling and people who only seem to disappear, but in fact either merged with another object or are occluded by other objects. We have not yet completely finalized this algorithm.

1.18.2 Notification that a collision occurred

Sometimes the bus can collide with an object, especially a person, and the driver does not notice it. It is therefore important to notify the driver if a collision has occurred. A

notification will be triggered if the probability of collision is 100% for times up to 1 second.

1.18.3 Frequency of alarms

We analyzed 5 hours of data to see how many alarms we will get. The following table lists the number of alarms according to side (left or right), severity (alert or imminent warning), and sensitivity level of the warning algorithm:

sensitivity	low	medium	high
left alert	62	75	91
left imminent warning	15	21	27
right alert	17	24	40
right imminent warning	2	2	4

Table 23. SCWS Alarm Frequency

These alarms contain true and false positive alarms. The subsection titled False Alarms in section Testing and Data Analyses deals in more detail with false alarms. Taking the numbers for the medium sensitivity, then we will get an alert once every three minutes and an imminent warning once every 13 minutes.

Another interesting measure is how long the alarms will last. Following table lists the average duration of the alarms when the high sensitivity was set:

	cycles	time [s]
Left alert	30.6	0.41
Left imminent warning	24.4	0.32
right alert	26.8	0.36
right imminent warning	7.8	0.1

Table 24. SCWS Alarm Duration

About 80 % of the alerts last less than 0.5 seconds and most are 0.1 and 0.2 seconds long (see figure below).

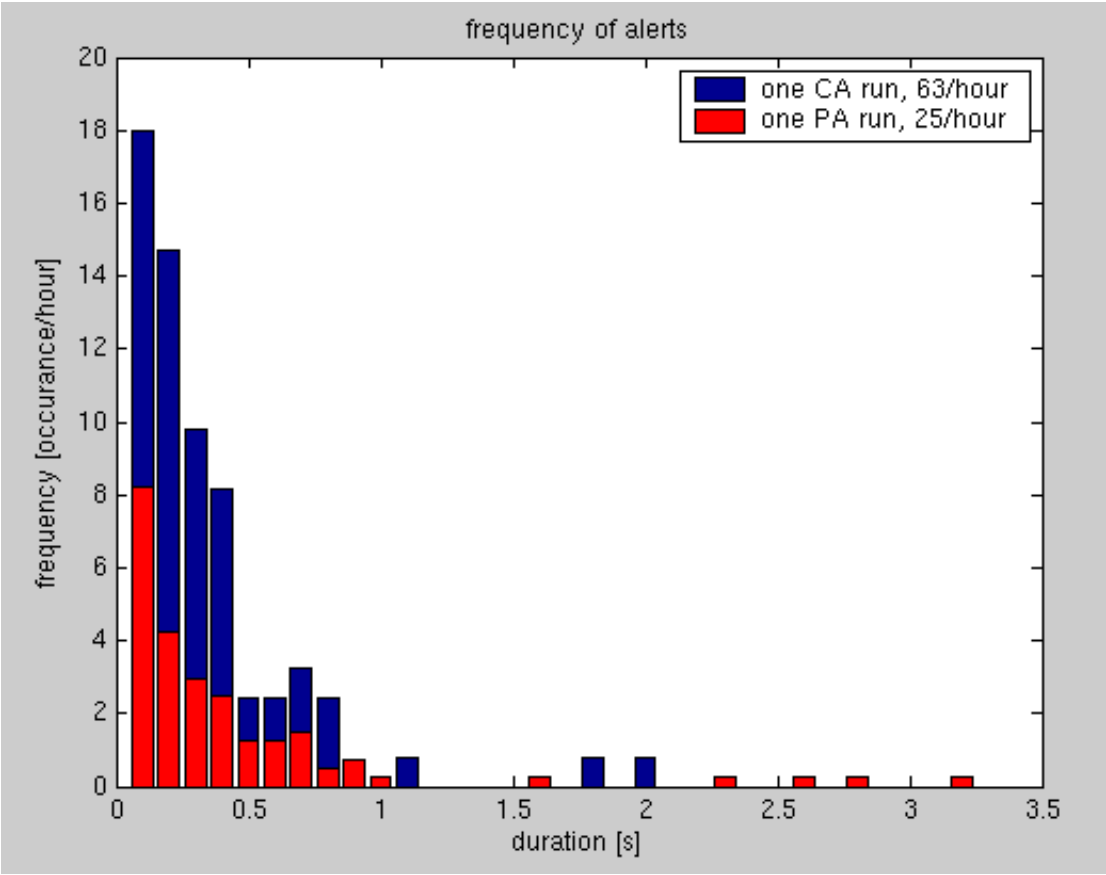


Figure 53 - Duration and frequency of SCWS warnings

At first glance, this seems like warnings occur way too often and certainly more side warnings occur than frontal one. It should be pointed out that the frontal and side CWS systems serve different purposes. The frontal alerts are used as an aid for distracted driver, while the side alerts provide the transit operator with additional information. As shown above, most of the warnings are very short and transit operators have not complained about too many warnings. This was also seen by a researcher riding on the bus who also did not feel that there are too many warnings.

1.19. False Alarms

A few things need to be said about false alarms. False alarms can be caused by system failure, *i.e.* the system did not perform as expected. It is also possible that the system performs as it should, but the driver considers the alarm a nuisance. In this section we will mostly discuss the first kind; we will describe system failures we observed. The second kind is mostly part of testing the system with drivers, but some of these nuisance alarms are due to the inability of the system to recognize certain situations.

Since there are only a few positive alarms, it is relatively easy to test for false positives. To find the rate of false negative alarms on the other hand is very tedious.

In the following sections we will discuss the false alarms mostly qualitatively. Some of the sources of false alarms have since been eliminated, but we have just begun to collect new data.

1.19.1 Sources of false positive alarms

1.19.1.1 Incorrect velocity estimates

In section 1.35.2.2 Error characterization of the full DATMO we found that the error in the velocity of objects can be described by a Gaussian distribution plus some outliers. The Gaussian error will cause some velocities to be a little bit off, but that can increase the probability of collision by enough to trigger a higher warning level. These false alarms are not necessarily a nuisance to the driver since the situations are in fact somewhat dangerous, just not as dangerous as the system calculates it to be. We have found that when the driver can understand the basis for why the warning was triggered, he will not perceive the warning as false, even if it is not as dangerous as the system displays. Quite a different matter is the case of the outliers. Here warnings might be issued for objects which pose no danger at all.

Since most of the false positive alarms in this category are caused by small errors in the velocity and only a very few are caused by the outliers, the transit operator is not overwhelmed by nuisance warnings that they don't understand.

1.19.1.2 Error in location

The distance accuracy of the laser scanner is very good (see previous section on the SICK laser scanner) and is very unlikely to have a false alarm due to an error in location. However, when the system determines if an object is on or off the curb, a small error in the position can have a large effect, if the position is very close to the curb. Also in this case, we need a second position measurement, namely the position of the curb. The position measurement of the curb is more prone to errors than position measurements of objects.

1.19.1.3 Vegetation

Vegetation poses a challenge in many ways. Returns from a bush can be very inconsistent and therefore DATMO might attribute false location and/or velocities to such objects.

It also sometime happens that a small amount of vegetation (e.g. some grass) is very close to the bus and triggers an imminent warning. The bus driver usually will not consider some grass as any threat at all and therefore will consider this warning as a nuisance or malfunction of the system.

1.19.1.4 Ground return

Usually the objects seen by the laser scanner are above the ground, like people, cars, mailboxes, walls, etc. But sometimes the scanner can see the ground itself, either because the ground is sloped or because the scanner is tilted. If the ground is seen in the path of the bus, a warning might be issued. These false positive alarms from ground returns have in the past been the biggest source of false positives. We discovered that the bus itself was titled by a few degrees towards the left side which resulted in many ground returns on the left. The problem has recently been fixed and we hope that this source of false alarms has been greatly reduced.

1.19.1.5 Sensor failure

During the operation of the side collision warning system we had several sensor failures. Cameras got misaligned, the camera of the laser line striper stopped working because water leaked into it, and the laser scanner didn't fully extend. Each of these failures can cause false positive alarms.

One of the cameras and the laser line striper are used to determine the curb position. If they do not work properly, the curb position can not be determined and nuisance alarms can not be reduced.

When the laser scanner does not fully extend, it sees the frame of the bus. This return will be interpreted as an object very close to the bus and a warning will be issued.

Furthermore, the scanner is misaligned and objects appear at incorrect positions which can lead to false alarms.

1.19.2 Statistics of false positive alarms

We looked at all the alarms with the high sensitive setting mentioned in section 1.18.3 Frequency of alarms and tabulated them according to following categories: True positives, velocity might be somewhat off, velocity is an outlier, vegetation, and ground return. The category "velocity might be somewhat off" is a judgment call because we do not have an independent measurement of the velocity. We watch the video or the raw laser scanner data to judge if the velocity given by DATMO is reasonable. It is also not always obvious, if there would have been the same alarm or not if DATMO would have given the correct velocity.

	true positive		velocity off		velocity outlier		vegetation		ground return	
left alert	40	44%	15	16%	5	5%	0	0%	31	34%
left imminent warning	1	4%	7	26%	3	11%	0	0%	16	59%
right alert	25	63%	3	8%	7	18%	4	10%	1	3%
right imminent warning	1	25%	2	50%	1	25%	0	0%	0	0%
total	67	41%	27	17%	16	10%	4	2%	48	30%

This data set was taken before we leveled the bus and it therefore has a great number of false positives caused by ground returns (30% of total). We also analyzed a later run, after the bus was leveled, and we did not see any more problems with the ground return. However, we experienced a failure of the laser scanner in that later run. The sensor did not always extend fully. The data was therefore corrupted and resulted in 120 (!) imminent warnings on the right side within a 5 hour period.

In summary one can say that the majority of the positive alarms are understandable by the transit operator. Many of the false positives are not very seriously false (velocity off), and the driver might not even consider them nuisances. When a large amount of false positives are seen by the operator, the problem can be traced back to sensor failures (e.g. laser scanner not level or not fully extended). The number of serious false positives which will be present even if all the sensors work correctly is small and due primarily to velocity outliers which represent about 10% of warning generated.

1.19.3 Sources of false negative alarms

Many of the reasons for false positive alarms can also cause false negatives. Specifically these are errors in velocity and location. The ratios (false positives)/(correct negatives) and (false negative)/(correct positives) due to these errors should be comparable. But since there are much less correct positives than correct negatives, one should expect much less false negatives than false positives due to these errors.

1.19.3.1 Sensor failure

If the laser scanner or the vehicle state module stops to function, then the system will not be able to issue any warnings. If the laser line striper fails the system will not have the ability to reduce nuisance alarms by considering the relation of objects to the curb.

1.19.4 Reduction of nuisance alarms through curb detection

The system tries to reduce the number of nuisance alarms by taking into account the relation of the object to the curb when the probability of collision is calculated. Details of the method can be found in the paper "A 2D Collision Warning Framework based on a Monte Carlo Approach".[12] We found that there are 30% less alerts when using curb information. For a few scenarios the warning severity increased, where vegetation reached over the curb and therefore its position was considered off the curb. The system worked as expected, but the driver might consider an imminent warning for an overhanging bush as a nuisance alarm.

1.20. System Faults and Recovery

1.20.1 SCWS System faults and recovery

The SCWS system has several layers of fault detection and recovery.

- First, any process which dies is restarted within 5 seconds.

- There are processes which are labeled "vital." If a vital process dies then the whole SCWS system will be gracefully shut down and restarted. Vital processes include data logging processes, as if we lose a data logging process then the data continuity

[12] Mertz, C. A 2D Collision Warning Framework based on a Monte Carlo Approach.

could be compromised.

- The central system manager checks to see if both the left and right processors are still up. If it loses contact with either process it shuts down the SCWS system, waits until contact is reestablished, and then restarts the SCWS system.

- All processes in the system have a "heartbeat" which is propagated via the shared memory system. These heartbeats contain the time of the last run and some simple debugging messages. The central system manager monitors these heartbeats, and if it does not see a heartbeat change for a process in 30 seconds, it shuts down and restarts the SCWS system, as a "hung" process can have serious repercussions on the proper operation of the system.

- The heartbeat information can be displayed in a graphical user interface for debugging as shown in Figure 54, but the same information is also periodically saved to disk for later debugging Just as each line of the GUI indicates the status of a running process to give a system overview at a glance, the log file contains all the necessary information to judge the system's health remotely.

- The system sends e-mail via a cell modem to the researchers when it starts and finishes and researchers can remotely check the heartbeat log to make sure the system is functioning properly.

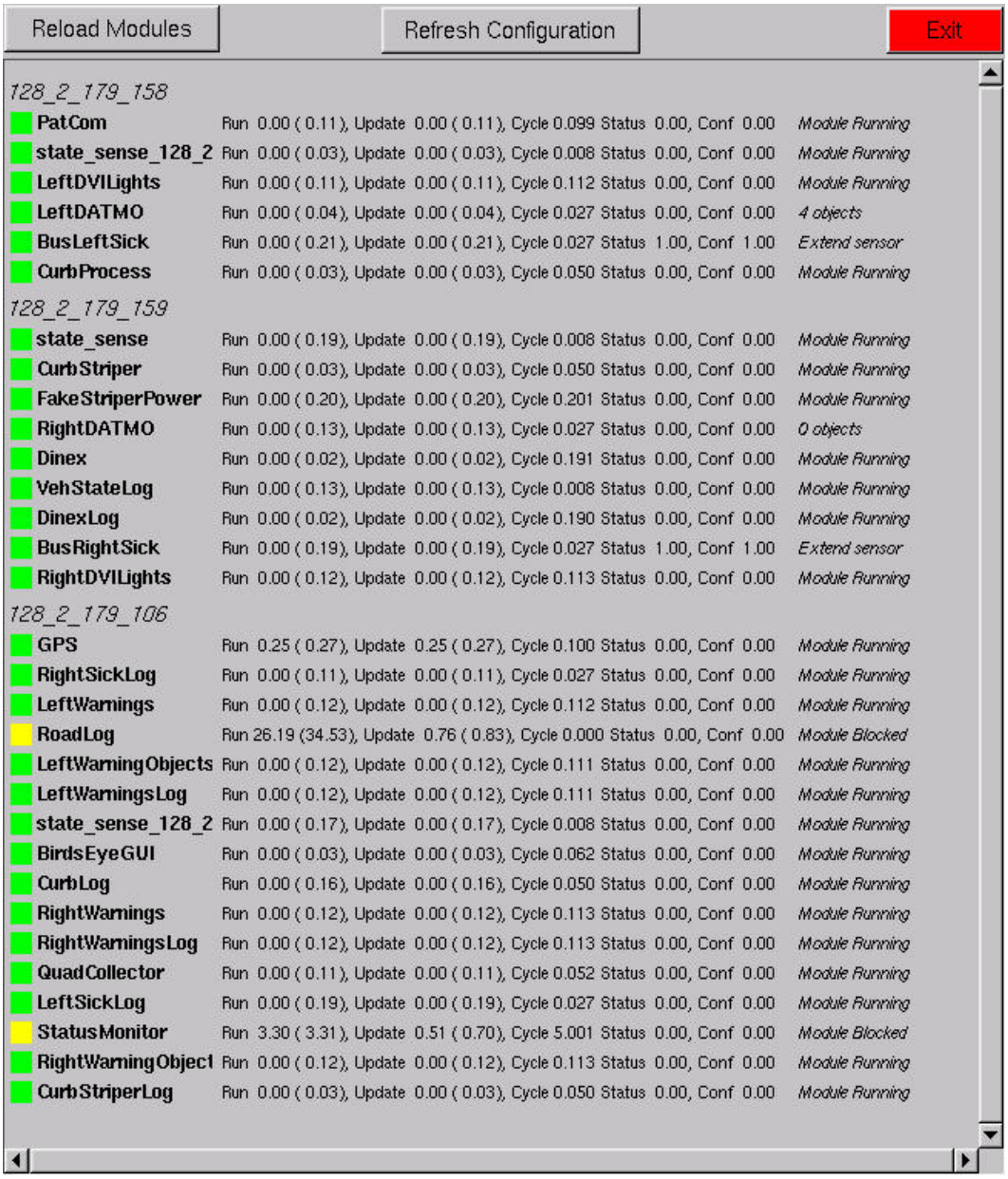


Figure 54 SCWS Status Debugging GUI

1.20.2 FCWS System faults and recovery

A fault is an unexpected change in a system which tends to degrade overall system performance. Early detection of faults in the FCWS can be communicated to the driver in which case the driver may solely rely upon their own judgment when driving and report the malfunction to an engineer as soon as possible.

A fault can be categorized into different classes from different perspectives, for example faults can be categorized into either static faults or dynamic faults, or software faults or hardware faults. For the FCWS from a system input/output perspective, faults are categorized into four main groupings: power faults, sensor data faults, DVI faults and hard disk faults.

Many approaches have been proposed for fault detection, isolation and system recovery. For example, two speed sensors might be installed to measure the vehicle speed, so that if one of them is detected malfunctioning, the data of that sensor will not be used. However, the extra cost incurred must be considered for the hardware redundancy. The approaches proposed here are mainly traditional approaches, which require no or little additional hardware cost, and are model-based approaches, which make use of mathematical models of the system.

When a fault occurs, some actions need be taken based on the severity of the fault. For example, if a power fault is detected and confirmed, a warning message (DVI fast flashing) might be displayed for a couple of seconds, the fault will be recorded in a disk file, and after the warning is given the system will be automatically shut down or switched to a debugging mode until the problem is solved.

1.20.3 FCWS Faults categorization

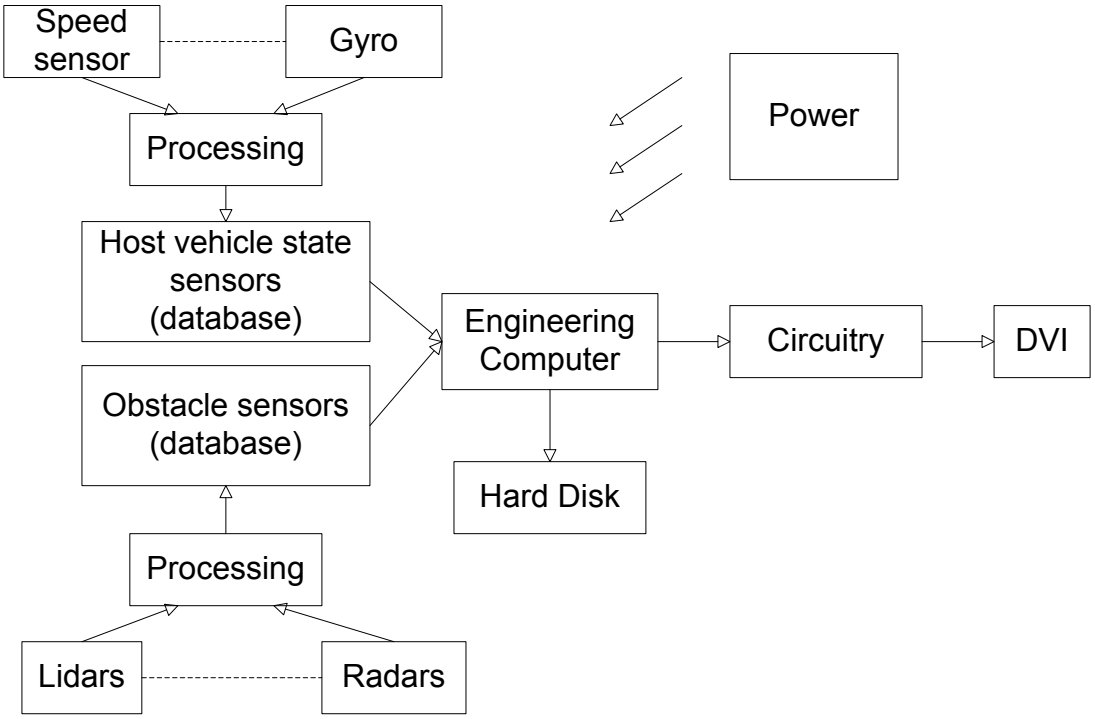


Figure 55 FCWS Fault categorization

To categorize faults of the FCWS, the first thing we need to know is how the system works. Basically, the FCWS reads the sensor data from the database and processes the data to issue warnings (displayed on the DVI), at the same time, sensor data and other information (including fault records) are recorded on the hard disk. A fault may occur in a sensor itself, in the signal driver or filter, or in the software processing. Regardless of where it happens, or what type of fault it is (mechanical or electrical), there will be a fault of the sensor data in the database. Hence from the system input/output prospective, for the FCWS, there are four main fault categories: power faults, sensor data faults, DVI faults and engineering computer faults.

1.20.4 FCWS Fault detection

1.20.4.1 Power faults

An open circuit or a short circuit may occur in power transmission lines. The use of Kalman filter for power system state estimation was introduced in 1986. For the FCWS, it is possible to utilize Kalman filter to detect power faults. Additional A/D channels are needed to monitor the power supply. Thus the state space models for the voltages and currents, the noise statistics could be investigated. Further investigation and research are still needed for power fault detection.

1.20.4.2 Key sensors (for vehicle states and target detection/tracking)

Speed sensors, steering angle sensor/gyro/accelerometers are used for the estimation of vehicle states. LIDARs provide information on target detection/tracking. The measurements of these sensors are essential for vehicle/targets state estimation and prediction. Faults of these key sensors could be circuitry fault, mechanical fault or software fault, which result in corrupted sensor data. The fault could be detected and isolated using the following approaches.

1.20.4.2.1 Traditional fault detection approaches

1.20.4.2.1.1 Installation of multiple sensors (hardware redundancy)

Additional sensors may be installed to compare the measurements of the speed, the steering angle, etc. There are a lot of algorithms based on hardware redundancy, however, the extra cost must be considered.

1.20.4.2.1.2 Limit checking

All measurements could be checked based on a pre-set limit. If the measurement exceeds the limit a fault is indicated. For example, the LIDAR has its own detection range and azimuth coverage. If the LIDAR data exceeds the limit, the malfunction will be recorded. This approach is recommended.

1.20.4.2.1.3 Fault dictionary approach

Each type of fault has its own characteristic. A fault dictionary contains all known "characteristics". We will know if a fault presents by comparing the system behavior with repertoires of faults in the dictionary. For this approach the more we know about the outcome of a fault, the more efficient our fault detection will be. For example, when the string-pot is broken, the steering angle data becomes a constant value even when the bus turns. This pattern could be saved in the dictionary and the fault could be easily detected. This approach is recommended.

1.20.4.2.2 Model based approaches-Basic principle of Kalman residual test

In the FCWS, this method is recommended as it requires no additional hardware, and is easy to implement and capable of detecting and isolating (indicating which sensor data has problems) most sensor data faults. Utilizing the Kalman filtering method, there are two options: Kalman residual test or (χ^2) Chi-square test. After investigations and simulations, the first method is found to be both an effective and efficient way to detect some of the key sensor faults.

If the Kalman filter is correctly modeled, the innovations, which are the differences between what comes out of the sensor and what is expected, would be zero mean white noise and its autocorrelation function would be zeros except at zero delay.

Assuming system model:

$$\begin{cases} X_k = \phi_{k/k-1} X_{k-1} + W_{k-1} \\ Z_k = H_k X_k + V_k \end{cases}$$

where X_k is an n dimensional system state vector, $\phi_{k/k-1}$ is an n*n system matrix, W_k and V_k are independent Gaussian white noise vectors with n dimension and m

dimension respectively, Z_k is an m dimensional measurement vector. H_k is an m*n system measurement matrix.

$r_k = Z_k - H_k \hat{X}_{k/k-1}$ where $\hat{X}_{k/k-1} = \phi_{k/k-1} \hat{X}_{k-1}$.

r_k is m dimensional zero mean white Gaussian noise with $E(r_k) = 0$, $E(r_k r_k^T) = A_k = H_k P_{k/k-1} H_k^T + R_k$ when there are no sensor failures.

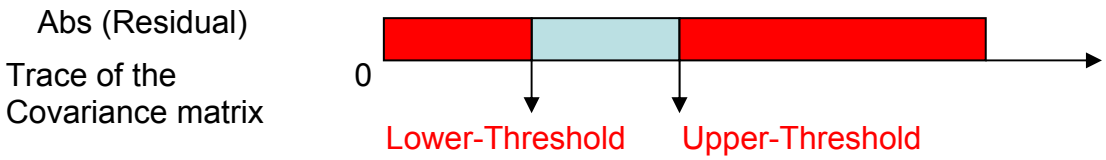
When there are sensor failures, it will not be zero mean white noise.

H_0 : No fault detected: $E(r_k) = 0 \quad E(r_k r_k^T) = A_k$

H_1 : Sensor failure: $E(r_k) = \mu \quad E[(r_k - \mu)(r_k - \mu)^T] = A_k$

r_k is a Gaussian vector. Therefore, to detect sensor failures it is convenient to use the log-likelihood $r_k^T A_k^{-1} r_k$ which is a Chi-square statistic (with m degrees of freedom). The threshold should be determined according to our needs.

An upper threshold and lower threshold for the absolute value of the residual and the trace of the covariance matrix of the Kalman residual are used for fault detection. This is based on the fact that the estimation is not perfect resulting that the Kalman residual can not be too small all the time, but it can not exceed a certain limit either otherwise we would have used other gain factors in the Kalman filter.



The following three figures show the raw speed data, Kalman filter residual, trace of the covariance matrix of the residual and its lower threshold when the speed data is corrupted starting from the 100th sampling (This is a simulation of a fault. Speed: m/s). Once the trace of the covariance matrix of the Kalman residual is below the lower threshold, the event will be recorded on the hard disk.

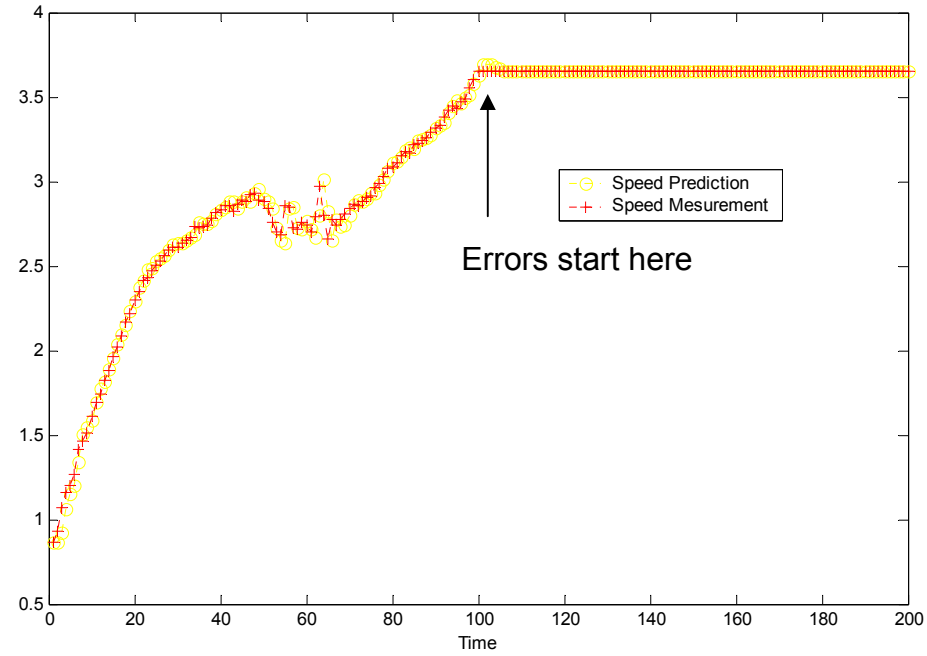


Figure 56 Speed raw data

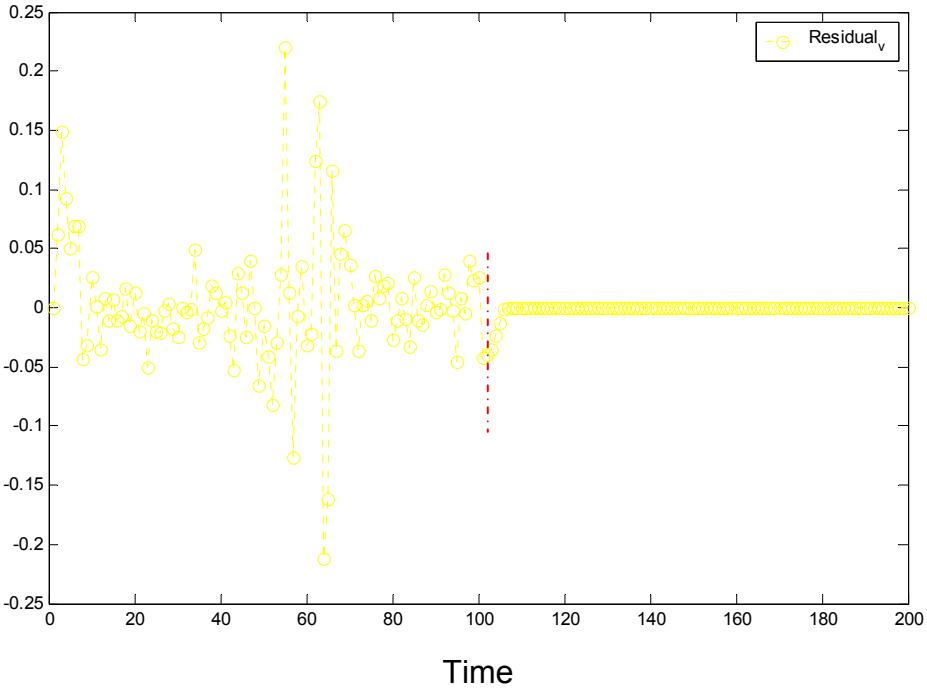


Figure 57 Kalman filter residual

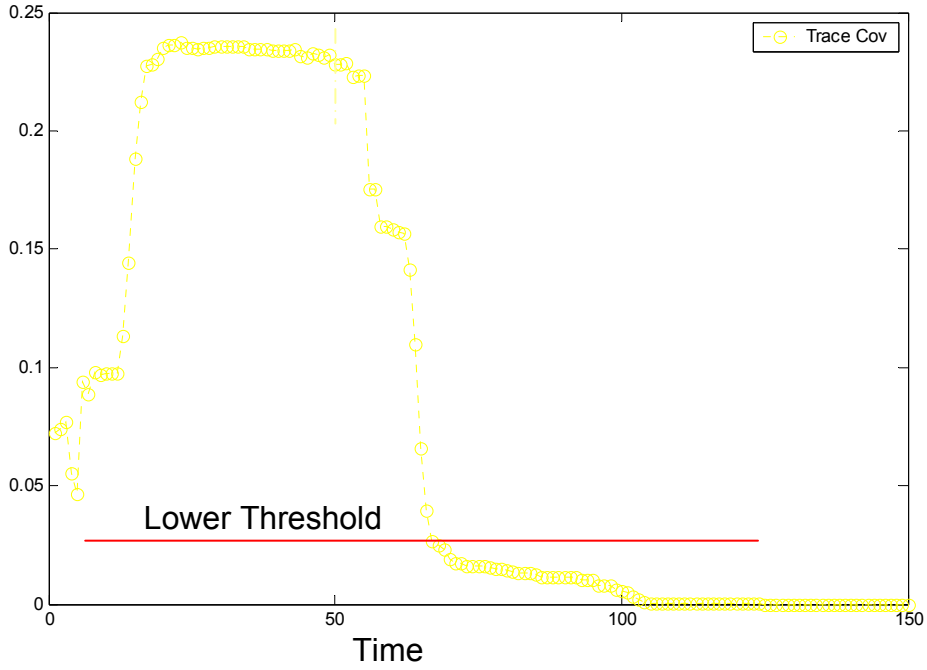


Figure 58 Trace of the covariance matrix

The following three figures show the raw speed data, Kalman filter residual, trace of the covariance matrix of the residual and the upper threshold when the speed data was interfered by an additional -10db white Gaussian noise starting from the 100th sampling. Once the trace of the covariance matrix of the Kalman residual is higher than the upper threshold, the event will be recorded on the hard disk.

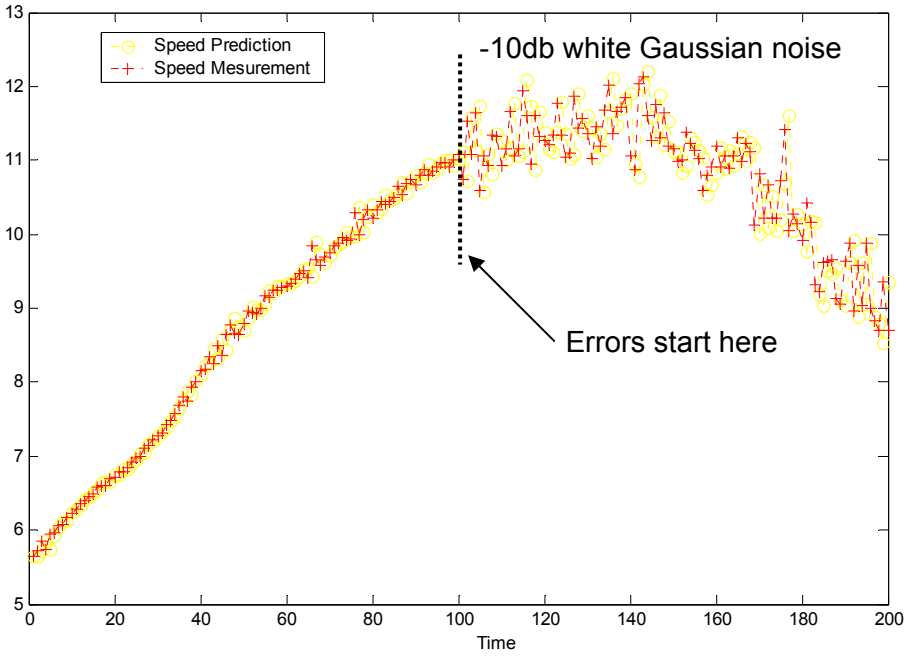


Figure 59 Speed raw data

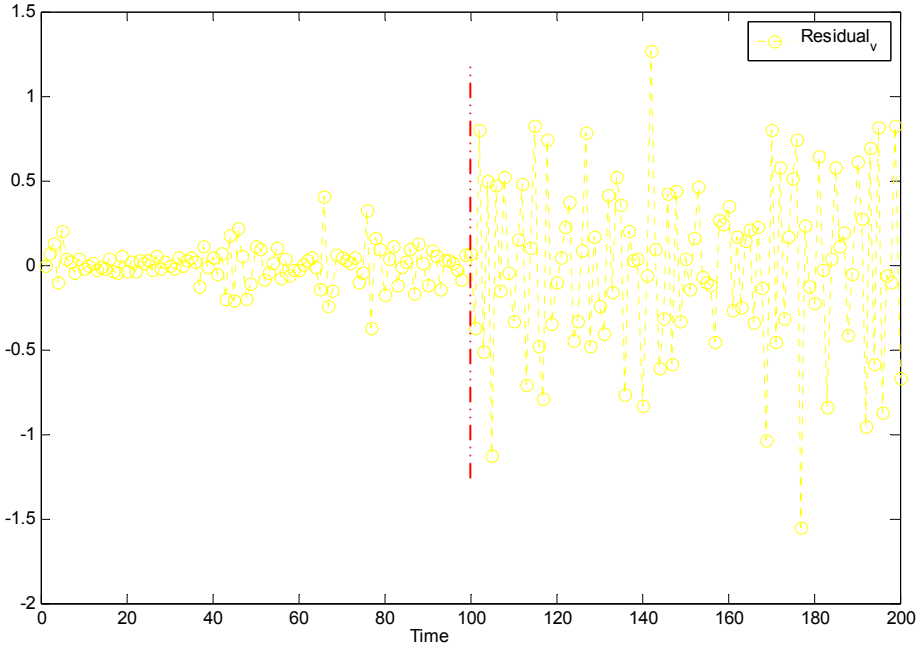


Figure 60 Kalman filter residual

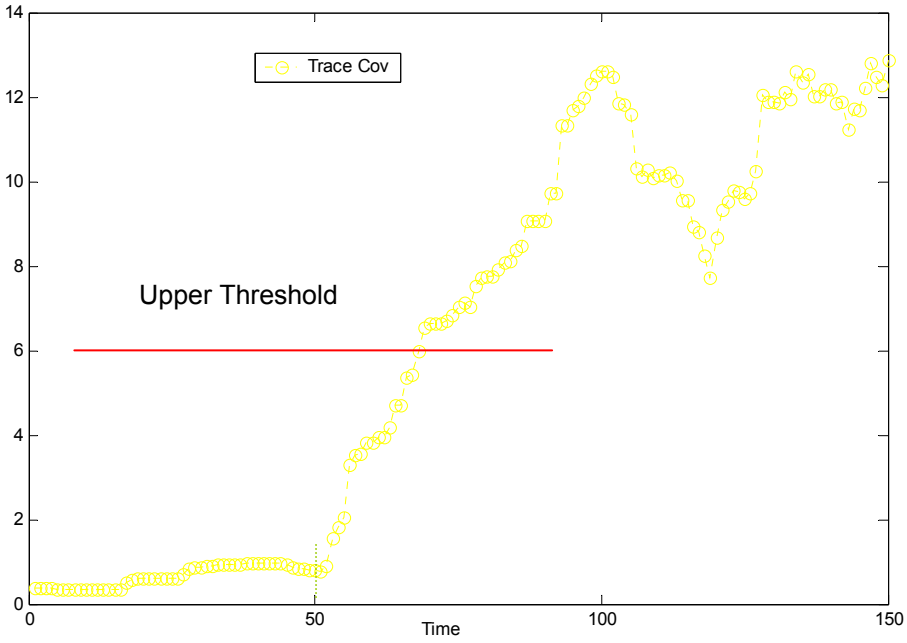


Figure 61 Trace of the covariance matrix

1.20.4.3 DVI faults

Essentially, the DVI is part of the system circuitry. Circuitry faults include stuck faults, bridge faults, short circuits, open circuits, etc.

One simple way of DVI fault detection is to let it flash at certain frequency for a few seconds when the engine is ignited. The driver could easily find any broken LED. In addition if the DVI is broken when the bus is in operation, the driver should identify what is not working and sometime later report it to an engineer.

Despite the off-line fault detection mentioned above, there is an on-line approach as well. At present, DVI could be regarded as “write only memory cells”, which can only be written but can not be read or checked by the engineering computer. If additional A/D channels are available, it will be possible to monitor the DVI in real time and therefore the system will be capable of isolating the broken LED/circuitry. For example, if an open circuit of a LED occurs, when we write a “Low” to the digital output, and then read the input , a +5V voltage instead of a +1V voltage will be read and the open circuit fault will be detected at once.

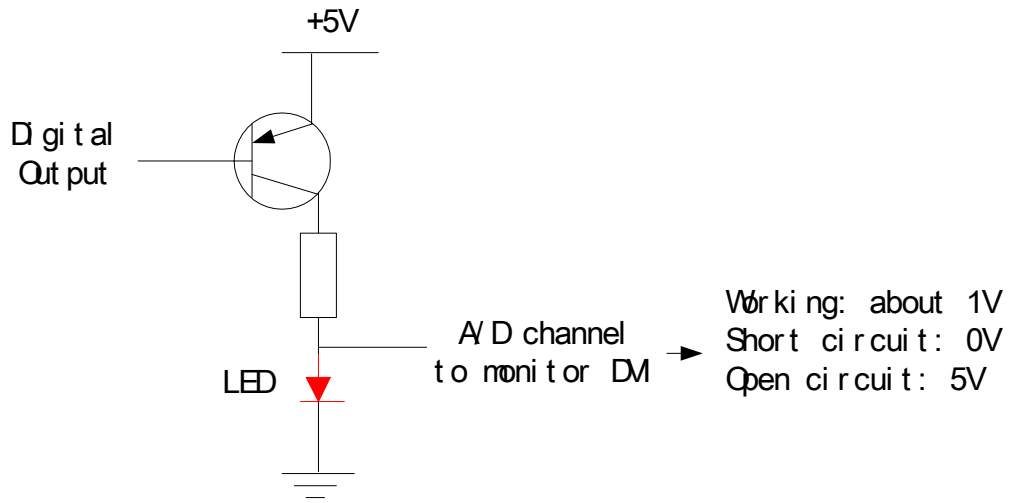


Figure 62 Detect DVI fault

1.20.4.4 Engineering computer / Removable hard disk faults

The fault of the engineering computer can be detected by its self-testing program. An additional program for the removable hard disk could be added. During the system initialization, a specific type of data could be written to the removable hard disk, and then the data will be read and verified to see if the disk is properly locked so that the recording can proceed as expected.

1.20.4.5 Detection scheme

To avoid false alarms, except for power faults, the fault detection should keep monitoring the system for a while (half to 1 minute) before giving an error code which confirms the fault. But all abnormal events will be recorded.

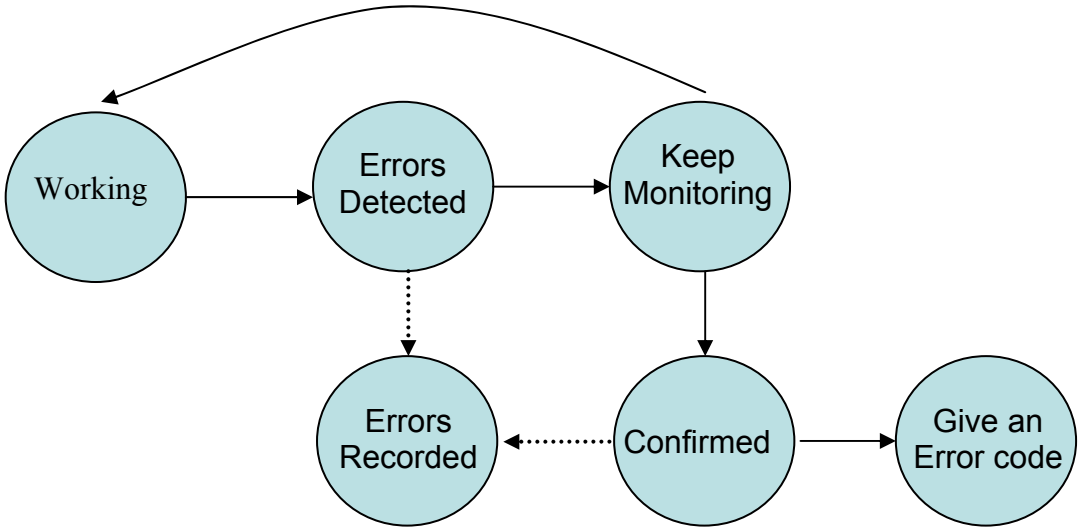


Figure 63 Detection scheme

1.20.5 FCWS Faults reporting and system recovery

1.20.5.1 Power faults

A power fault could be devastating, a fast flashing of the DVI will give an urgent warning telling the driver a severe fault has just occurred and the system will be shut down soon after. The fault should be recorded in a file for later off-line investigation. We might use a power relay or modify the current power relay to build a soft-switch that could automatically turn the system off after the urgent warning is given. Any faults regarding the power will need further investigation by an engineer.

1.20.5.2 Sensor data faults

If a key sensor data fault is detected and confirmed, the DVI will be disabled and an error code will be displayed on the DVI bar, indicating which sensor data may have problems. For example, if there is something wrong with the speed sensor data, the DVI will not display warnings issued by the collision warning algorithm, instead, it will turn on the lowest segment of the DVI bar to inform the driver of the malfunction (Assuming the DVI is working properly). All faults should be recorded in a file for later off-line investigation. Fault detection could locate which sensor data is corrupted, and record the fault property. By analyzing fault record at the receiving end, a good guess could be made about whether it is the mechanical fault or the circuitry fault, but repairing it will still need further investigation by an engineer.

1.20.5.3 DVI faults

If any part of the DVI is out of order, the driver himself could be aware of it quite soon. The rest of the LED's (which still work) may flash slowly for a few minutes to warn the driver that the DVI is out of order. The fault should be recorded in the file for further investigation. Although fault detection could locate the broken LED, it is not necessarily the LED that is broken, for example, the digital output line might be broken. Therefore repairing it will need further investigation by an engineer.

1.20.5.4 Engineering computer / Removable hard disk faults

The engineering computer has its own self testing program. An additional self testing program will be added to check if the hard disk works properly. If there is a fault, a

warning message will be given (an error code will be displayed on the DVI) and the driver may report the problem to an engineer. If it is not locked, then lock it. If the disk is broken, then replace a new one. If the disks are full, the new pc104 system will automatically stop recording new data and record the disk full message.

1.20.6 FCWS Summary

The summary of the system fault and recovery are shown below. Four categories of system fault: power fault, key sensor fault, DVI fault, engineering computer fault are described and its detection algorithm and detection strategy are proposed and system fault reporting and system recovery methods are summarized.

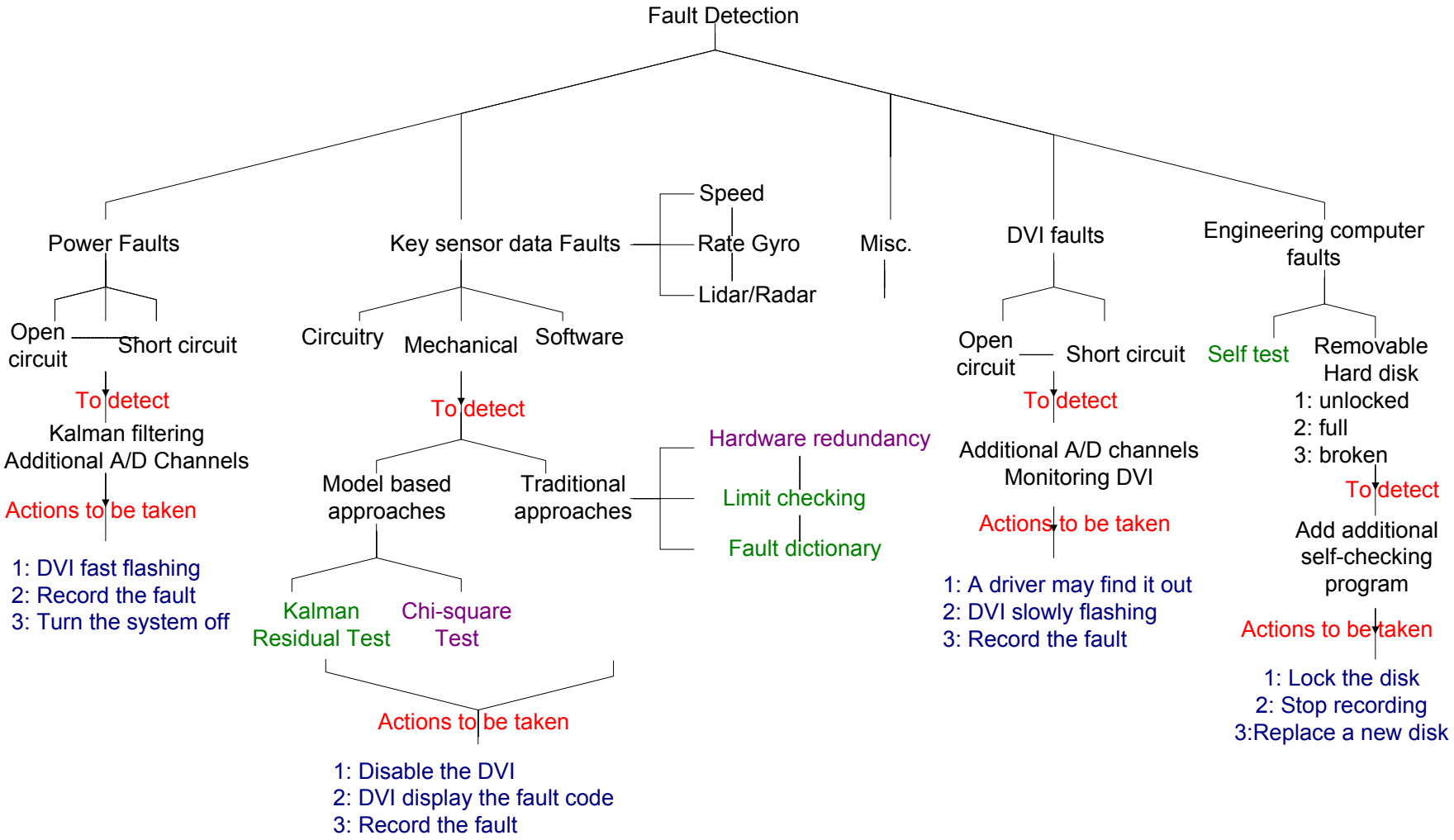


Figure 64 FCWS Fault detection architecture

1.21. FCWS Simulation Playback Tools

To aid in the analysis of the data collected from the buses and to test out alternative warning algorithms and sensitivity levels three tools have been developed. The tools were developed in an iterative fashion with the first tool being developed in May 2000. The three tools are:

1. The Data playback tool (developed in 2000)
2. The Simulator tool (this tool can work in conjunction with an updated data playback tool) (developed in 2003)
3. The Data marking tool (this tool is based on the data playback tool with additional functionality (developed in 2003)

Both the simulator tool and the data marking tools allow the user to post-process data off-line. The tools are used to help us analyze the warning scenarios by recreating detailed state information from any video clips that are of interest. (For example, to figure out if a warning was triggered appropriately). The purpose of the tools can be described as:

1. to run simulations of potential changes/improvements of the algorithm
2. to analyze/set system parameters, such as sensitivity levels

The three tools are described in greater detail below.

1.21.1 The FCWS Data playback tool

The basic data playback tool is a Windows™ based application and is designed so that a user can watch a clip of video (from 4 different views) while simultaneously displaying bus state information such as speed, acceleration, brake pressure, front wheel angle and GPS location. As this tool has been superseded by the Simulator tool and the data marking tool its function and improvements are discussed in the next two sections.

1.21.2 The FCWS Simulator Tool

The simulator tool is programmed in C and based on the FCWS warning algorithm program that is running on the FCWS and ICWS buses. The main difference is that instead of processing data from the database in real time, the simulator processes the data off-line using the interface subroutine, which converts the sensor files to a virtual database. Therefore, the simulator will have all the processing details and intermediate variables that are not recorded in real time processing.

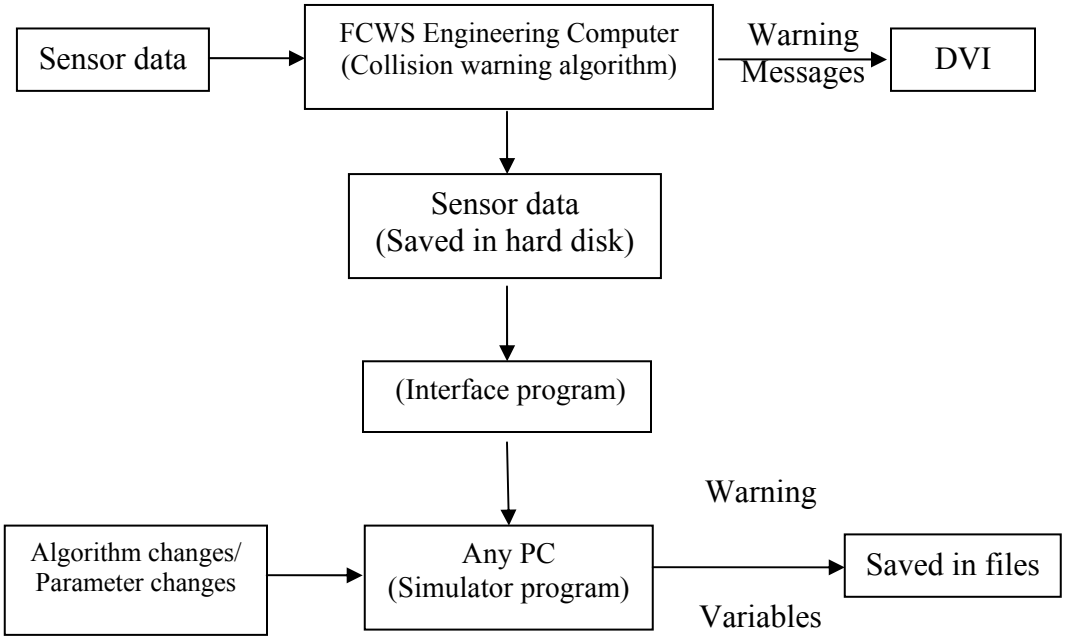


Figure 65 The simulation tool

An updated version of the playback tool utilizing the simulation tool is developed to help us comprehensively study a warning scenario. The display is divided into 5 sub-windows. Video from each of the three cameras is displayed in one sub-window. It projects the RADAR and LIDAR targets into the video frames, using simple visual marks to indicate which objects in the frames have been detected by which RADAR or LIDAR. The tool can decode and play back MPEG movies in Windows™. Two virtual DVI bars are added in the front-looking sub-window. Whenever there is a warning, it will be displayed accordingly as shown.

Host-bus measurements including bus speed, bus front wheel angle, bus accelerations and brake pressure, are displayed in the lower right sub-window simultaneously during video playback.

The right part of the lower left sub-window shows a birds eye view of the bus and targets around it. The larger blue box represents the bus, squares in green represent stationary/stopped obstacles in front of the bus, squares in yellow represent moving targets, squares in red represent warning trigger, which could be stationary, stopped or moving objects. The left part shows the prediction of target tracks (up-left) and the bus state estimation, which is in blue. They are all played simultaneously during video playback.

For example in Figure 66: at 10:21:18 there is a warning triggered by a subject vehicle (ID 185). Its raw data LIDAR measurement is mapped onto the front view window (small red circle), the text in red in the lower left window starting with the target ID-185 show the prediction of its relative position, speed of the target (relative to the bus). A birds-eye view of the scenario is shown in the big white circle with the same target ID 185.

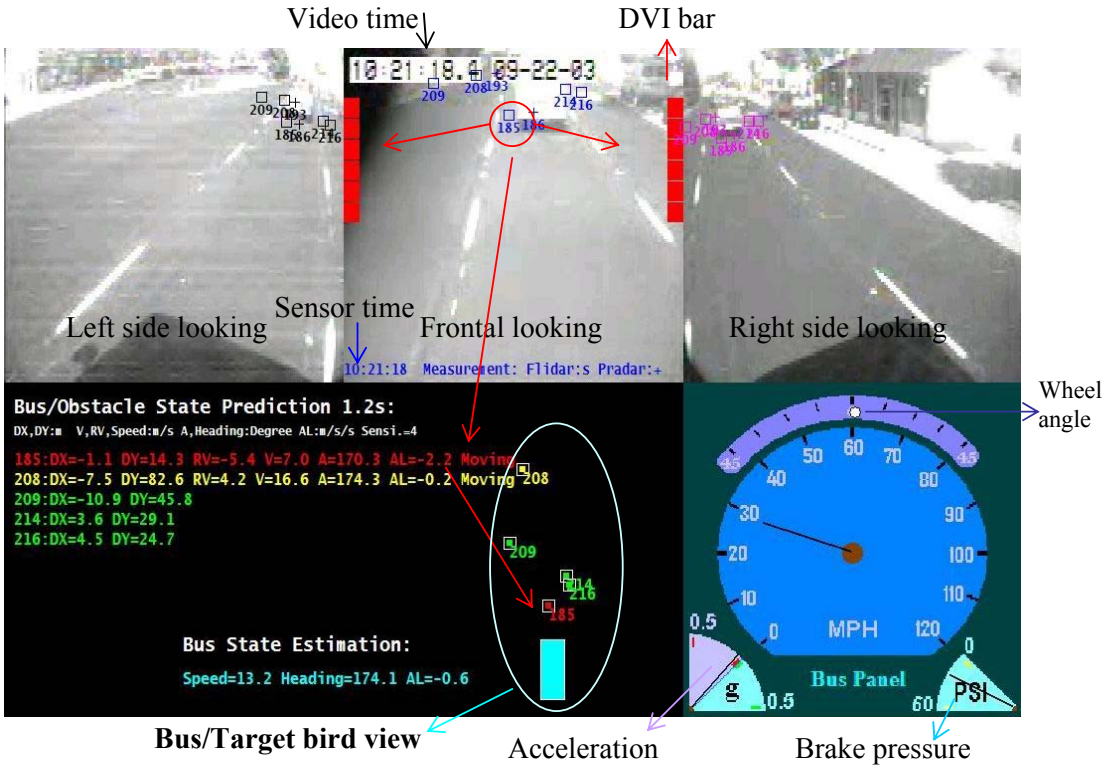


Figure 66 The updated playback tool

This tool provides the data reviewer a complete view of all the data collected at the same time. With the help of the simulator mentioned above, it also provides processing details, which include the bus/target state estimation and prediction. Furthermore, the tool provides the ability to understand sensor behavior, traffic scenarios, and the characteristics of targets.

1.21.3 The FCWS Video Data Marking Tool

This tool can decode and play back MPEG movies in Windows™. The display is divided into 7 sub-windows as shown in Figure 67 Video from each of the four cameras is displayed in one sub-window. DVI bars have been superimposed onto the forward view to display any warnings as they occur. Also projected onto the camera views are visual marks to indicate objects detected by the RADAR and/or LIDAR. Host-bus measurements including bus speed, bus front wheel angle, bus accelerations and brake

pressure, are displayed in the upper right sub-window simultaneously during video playback.

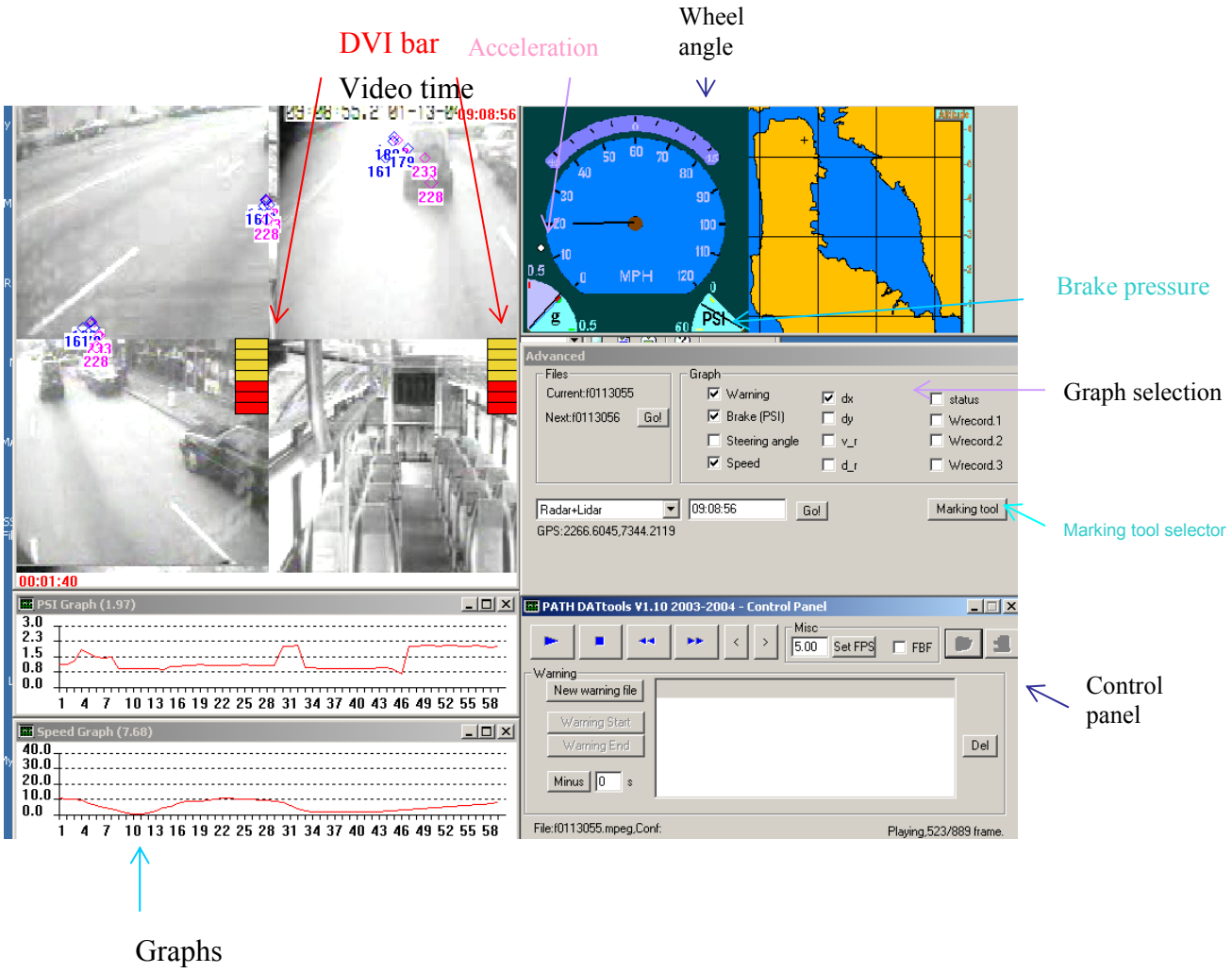


Figure 67 FCWS Video Data Marking Tool

The sub-window titled advanced allows the user to jump to the next saved file and to graph various bus states. 12 different items of data can be graphed. The first four graphs; warning, brake, steering angle and speed can be plotted directly from the engineering data. The remaining eight graphs require that the data is first post-processed. The three selectable graphs titled Wrecord1, Wrecord2 and Wrecord3 allow the user to plot data that has been post-processed off-line to determine what effect proposed changes in the algorithm or sensitivity levels would have. The graphs appear beneath the video sub-

window and plot data for 1-minute intervals simultaneously as the video plays. The graphs are useful as they allow the user to watch for trends in the drivers' behavior.

The sub-window titled Control Panel allows the user the following functionality:

- an option to set the frame rate (on a per second basis)
- an option to view the video frame by frame (i.e., each time the user hits play the video will advance one frame)
- a play/pause button
- a stop button
- fast-forward and rewind
- a tool to mark the beginning and end of all the warnings viewed by a user. Once a number of warnings have had their start and end times marked the user can open the "mark" file and select play to watch all of the marked sections.

The final sub-window is the mark tool sub-window as seen in Figure 68 , which is accessed by selecting the Marking Tool button on the Advanced sub-window. This tool allows the user to "mark" the data so that by selecting from any of the 5 rows of 6 buttons (30 buttons in total) when the video is stopped a new file is created that has the file name, the time, and which buttons were selected. The names of the buttons are changeable, the current pre-set set are:

Bus speed (in mph)

- 0–10
- 11–30
- 31–40
- 41+

Steer behavior

- yes prior to receiving a warning
- yes prior to and post receiving a warning
- yes post receiving a warning

- no steer around the time of a warning

Brake behavior

- yes prior to receiving a warning
- yes prior to and post receiving a warning
- yes post receiving a warning
- no brake around the time of a warning

Warning reason

- pulling into a bus stop
- a decelerating or stopped lead vehicle
- another vehicle cutting in from the left
- another vehicle cutting in from the right
- the road curving, trees and/or guardrails
- poles and/or signs

Driver comment (obtained by human factors researcher riding on the bus)

- The driver liked (thought the warning was appropriate)
- The driver did not know what the warning was for
- The driver made no comment
- The driver expected the warning
- The driver thought that the warning was late or wanted a warning

Using the above selections it is possible to synchronize driver feedback with video and engineering data to gain a more comprehensive understanding of patterns of drivers opinions of individual warnings. It is also possible to determine scenarios where drivers like and dislike warnings as well as take a look at scenarios where the driver wanted/expected a warning and was not given one.

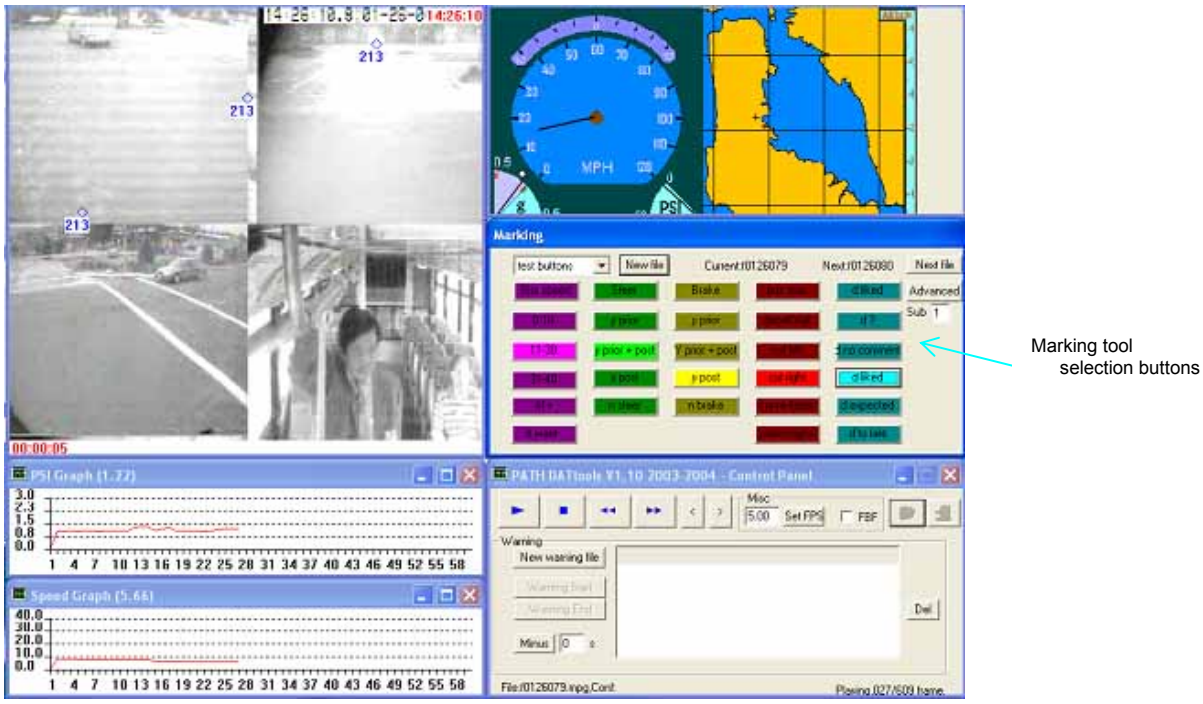


Figure 68 Mark tool sub-window

The simulator and the data-marking tool allow the user to study various variable of interest in an integrated way, providing the data reviewer with such a complete set of data collected at the same time

1.21.4 FCWS Analysis Procedure

A standard procedure is proposed for comprehensive analysis of four warning scenarios (True, Miss, False, and Nuisance warning). It includes basic technical analysis, warning timing/consistency analysis and driver feedback analysis. The basic technical analysis is to recreate the warning scenario in detail. By watching the video clip and analyzing the variables mentioned below, we try to evaluate the accuracy, smoothness and noise characteristic of the measurement, estimation and prediction of the host bus and targets (Bus track and target tracks) and try to improve the system in every aspect. The ultimate goal of the warning timing/consistency analysis is to achieve good warning timing and a high level of system consistency. For example, we would examine if a true warning is

issued too early or too late from both the technical point of view and the driver's point of view. If there is inconsistency, it could result from either threat assessment or the delay factors listed in the table, which would then be a factor for further testing and investigating. Furthermore, the driver's feedback is very important for us to adjust, evaluate and improve our system.

<table>
<tr><th colspan="5">True/Miss/False/Nuisance/ Warning Scenario Analysis</th></tr>
<tr><th colspan="3">Basic technical variables</th><th colspan="2">Timing/Consistency and Feedbacks</th></tr>
<tr><th>Measurement</th><th>Estimation</th><th>Prediction</th><th>From data</th><th>From drivers</th></tr>
<tr><td>Road Geometry</td><td>N/A</td><td>N/A</td><td>Brake Pressure</td><td rowspan="3">Comments for this particular warning</td></tr>
<tr><td>Weather(wiper)</td><td>N/A</td><td>N/A</td><td>Bus Heading</td></tr>
<tr><td colspan="3">Bus headway speed and yaw rate</td><td>Throttle Position</td></tr>
<tr><td colspan="3">Target lateral and longitudinal position and speed (relative to the bus)</td><td>Sensor Delay
Processing Delay
Driver Reaction time</td><td rowspan="3">Suggestions for similar warnings of this kind:
Warning timing
Warning level and Duration, etc.</td></tr>
<tr><td>N/A</td><td colspan="2">Bus/Target location, heading, headway acceleration, angle acceleration, height</td><td>Compensation/ Prediction time</td></tr>
<tr><td>N/A</td><td>N/A</td><td>ARQ, TTC,
Inv. TTC, etc</td><td>Sensitivity level
Warning duration
Starting/End time
Warning level</td></tr>
</table>

Table 25. Standard analysis procedure and main variables

1.22.

1.23. SCWS Data replay tools

The data collected by the SCWS is stored in multiple different files. Each file represents a single stream of information. These include:

1. Vehicle State

2. Raw range data from line scanners and curb sensors
3. Tracked and classified object information
4. Tracked and predicted curb information
5. Warning levels sent to the DVI
6. Auxiliary bus information, such as doors open/closed and signal status
7. DVI information, such as which lights have been lit and which buttons have been pressed
8. MPEG movies derived from the cameras pointed forward and backwards on each side of the vehicle.

Each data stream has the same underlying format created by the same underlying tools: A set of arbitrary data records where each record is not just tagged with the time of collection, but is indexed by it. The distinction between tagging and indexing by time is important: If each record was simply tagged with time we would have to search through an entire file in order to find a particular record at a particular time. Instead, we maintain a time based index for each data file that is loaded into memory at startup. When we want data from a particular time, we look up in the time index where in the data file the necessary record is, and retrieve that record from the file. The cost, of course, is in the up-front time and memory needed to load in the file index, but we find that modern systems can read in the index of a file with thousands of entries collected over hours in seconds without taxing the system's memory requirements.

The data replay system takes full advantage of the common, time-based data access and replay tools. At any given time there is a synthetic "replay time" estimate, i.e., the current time of the data we should be showing to the user. For each data stream that is being displayed, we simply use the index to look up and obtain the appropriate data for the current replay time. Not all data will be available at every instance in time, so when necessary we use common tools for shifting the data display by the appropriate vehicle motion. For example, if the most appropriate piece of object tracking information is 100 milliseconds before the current replay time, we can adjust the display of the object track display to account for the motion of the vehicle by shifting the display of all the tracked

objects by the distance and direction that the vehicle has traveled in the last 100 milliseconds.

Thus, the temporal and spatial synchronization of the many disparate data streams is achieved in a straightforward manner by the replay system. In addition, the approach to data replay lets us provide the users with movie player-like controls, whereby they can vary the flow of time, pausing, fast forwarding, slowing down, and even moving arbitrarily around in the data using a scroll bar. To the end user, it all looks like one unified data source that can be accessed like a single movie. For displaying data, we have taken as our inspiration web browsers, which provide a framework of common tools and constraints for displaying fairly arbitrary information with plug-ins.

In our data replay system we provide two main output modalities: The 2D OpenGL-based overhead view of the data and the data overlaid on the video we store from the forward and rear looking cameras.

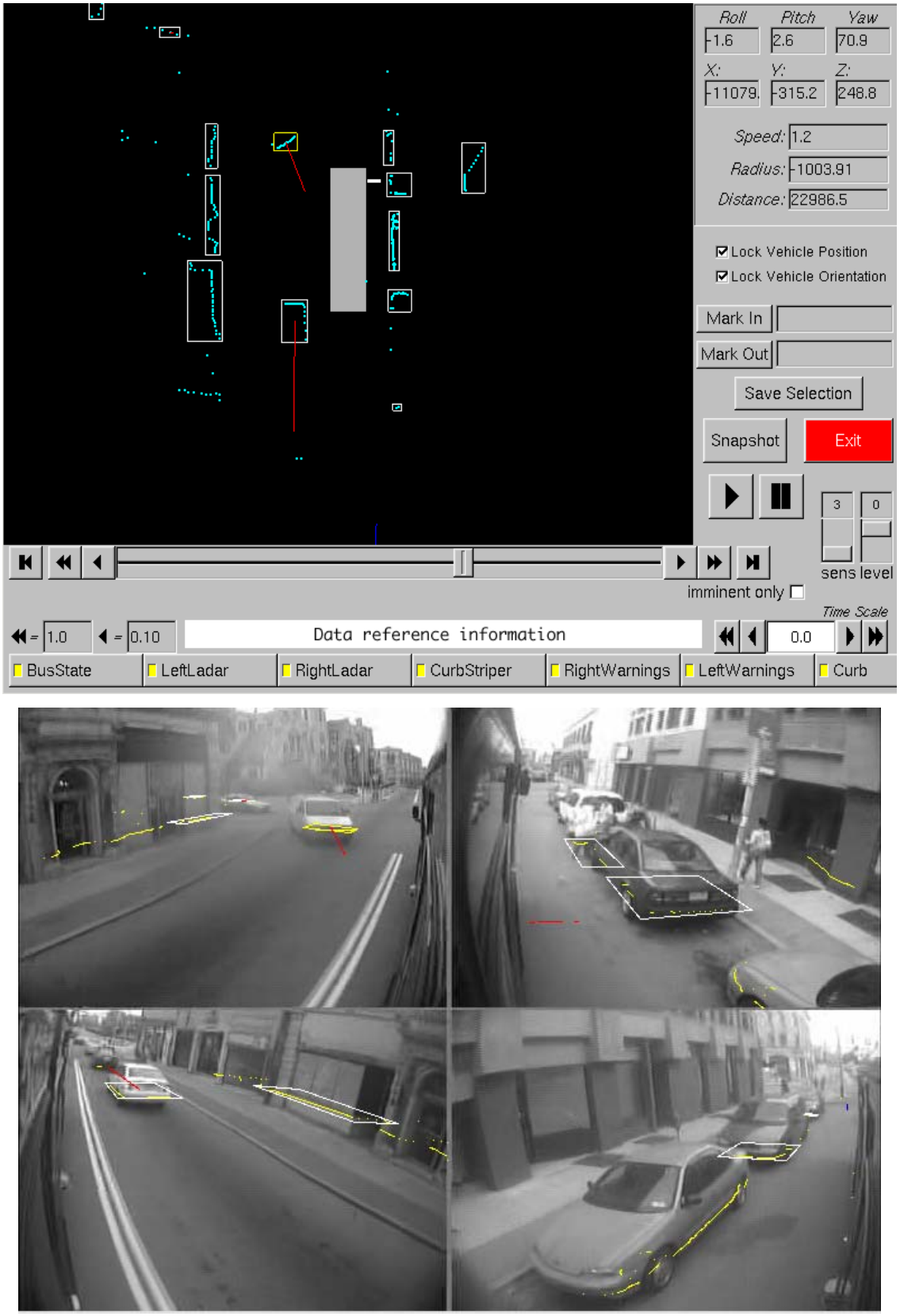


Figure 69 Example 1 of Overhead View and Overlaid data on video

Figure 70 Example 2 with bicycle

We provide a common framework to develop fairly arbitrary plug-ins for displaying data in these two output modalities. Each plug-in can implement methods for displaying overhead data or overlay data. The user configures the system to choose and configure a palette of these plug-ins for displaying the various data streams, and can select in real time whether to hide or show any individual data stream display. This allows us to have an almost arbitrarily expandable display system, where we can introduce new data modalities and manners of displaying data in an almost arbitrary manner. Thus we can have the display system easily evolve in the future while still being able to display today's data.

The data replay system can be used to explore our raw, collected data, but it can also be used in concert with analysis tools. For example, the user can select beginning and ending points in time and create a new data set just containing data in this time slice. This is not simply one file, but represents the appropriate sub-set of every data stream in the display palette. The smaller data sets can be easier to share and analyze for development purposes. In addition, our off-line analysis tools can go through a data set and generate a list of "bookmarks" which can be loaded into the replay tool. The user can then instantly navigate to these bookmarks to examine the parts of the data set that the analysis tools have marked as interesting.

DVI DEVELOPMENT

1.24. Background: Transit Collision Warning Nuances

There are a number of transit operations characteristics that make this development effort particularly challenging. First and foremost, transit operators routinely drive close to other vehicles, obstacles, and pedestrians. The former two are specifically related to the size and handling of the vehicles in question and the locales in which they operate. The latter is due to events near bus stops where drivers are expected to pull close to the curb, thus coming into close proximity to waiting patrons and other pedestrians (and fixed objects like bus shelters). The challenge that this operating environment presents to a collision warning system is to determine under what circumstances a driver is intentionally operating the bus close to other objects and under what circumstances a driver is not aware of an object that poses a potential threat that the driver should be warned about.

Also worth highlighting is the environment the driver operates in and the perceptual demands that accompany transit tasks. Instrument panels are often mounted very low and out of sight for most drivers. Shifts are long, and the environment is noise rich, with many other audible warnings, passengers, and cell phones. Visual search is extensive; bus drivers are required to track many more visual targets in their field of view than their counterparts in passenger vehicles.

Finally, transit operators often encounter risky behavior on the part of nearby drivers and pedestrians. For example, it is not uncommon for vehicles to speed past a bus on the left and then cut in front, only to immediately turn right.

1.25. Guiding Concepts

Previous work towards a driver-vehicle interface (DVI) under this program identified three major paradoxes present in transit collision warning interfaces: [13]

[13] Steinfeld, A. FCWS Driver-Vehicle Interface Driver and Trainer Input and PRELIMINARY Design. California PATH, U. C. Berkeley. Unpublished, 2001.

1. Drivers agree with the philosophy of earlier action rather than harder action yet they would like as few alerts and warnings as possible.
2. Nighttime drivers prefer audible warnings due to concern over glare while daytime drivers tend to focus on visual warning options.
3. The warning should be salient enough to elicit a driver response but should not be readily noticeable by passengers.

It is important to keep the tradeoffs inherent in these paradoxes in mind when developing such systems. While often suggested by technologists new to the field, vibration displays in the seat or steering wheel have traditionally been strongly discouraged during driver interviews due to long shift durations. For example, one driver commented, "After 8 hours I don't have any idea what's going on down there.".[14] In addition many drivers report that when doing long shifts they constantly change their seating position, often times sitting at an angle which would make positioning of a vibration display for forward and side warnings problematic.

Other items of note are concerns that warnings may act as a "starting gun" for fraudulent falls by passengers (a very real problem) and that a high rate of false alarms will lead to severe dissatisfaction with the system. These concerns point towards a DVI that is discreet, not obnoxious, and isolated to the driver's personal space.

Furthermore, it is necessary to provide a level of driver control so that individual differences and environmental factors can be accommodated. As such the FCWS DVI preliminary specification recommended that drivers have the ability to modify the brightness and volume of displays to suit their needs.[15] However, there was also specification that drivers should not be able to use such adjustments to disable the system. Outside reviewers of the preliminary specification concurred:

[14] Steinfeld, A. FCWS Driver-Vehicle Interface Driver and Trainer Input and PRELIMINARY Design. California PATH, U. C. Berkeley. Unpublished, 2001.

[15] Steinfeld, A. FCWS Driver-Vehicle Interface Driver and Trainer Input and PRELIMINARY Design. California PATH, U. C. Berkeley. Unpublished, 2001.

"The idea of 'only one modality can be off' may not only be smart but also wise. This option allows some accommodation to the perceptual diversity of drivers; some may prefer auditory over visual warnings and vice versa." [16]

As such, this design feature was carried forward for the integrated DVI. Subsequent feedback from PAT employees reinforced this philosophy.

1.26. Warning Design

There was specific care to utilize multiple levels of warning for both the side and forward components. This practice has been suggested and successfully deployed in other intelligent vehicle research (e.g., [17], [18], [19], [20]). Previous iterations of the forward warning systems investigated the use of a three color level warning system (red, amber and yellow). However this was reduced to two level after drivers commented that it made the display "too busy" and that they did not find the amber color alerting enough.[21]

DVI activation is consistent across the forward and side components. As the Under Wheel case is considerably more dangerous than Contact, it has been assigned the red option.

1. *Alert*: Yellow LEDs.
2. *Imminent Warning*: Red LEDs.

---

[16] Mitretek. FCWS Driver-Vehicle Interface Driver and Trainer Input and PRELIMINARY Design Comments and Observations. Unpublished 2001.

[17] Graham and Hirst. "The effect of a collision avoidance system on drivers' braking responses." Proceedings of the 4th Annual Meeting of IVHS America. Washington, DC: 1994: 743-749.

[18] Wilson, Butler, McGehee, and Dingus, "IVHS countermeasures for rear-end collisions, driver warning system performance guidelines." Proceedings of the 1996 Annual Meeting of ITS America. 1996: 949-957.

[19] Dingus, Jahns, Horowitz, and Knipling. "Human factors design issues for crash avoidance systems," Barfield and Dingus eds. Human Factors in Intelligent Transportation Systems. New Jersey: Lawrence Erlbaum Associates, 1998: 55-93.

[20] Steinfeld and Tan, "Development of a driver assist interface for snowplows using iterative design." Transportation Human Factors, vol. 2, no. 3. 2000: 247-264.

[21] Wang, Chang, Chan, Johnston, Zhou, Steinfeld, Hanson and Zhang. Development of Requirement Specifications for Transit Frontal Collision Warning System. Unpublished August 2003.

3. *Contact*: The triangles for the appropriate side blink yellow.
4. *Under Wheel*: The triangles for the appropriate side blink red.

The DVI hardware includes integrated speakers in the LED assemblies in order to reduce the installation requirements of the system. The use of sounds to augment the alerts is being examined in related simulator research. Currently, the plan is for sounds to be issued for all stimuli except Alert. However, sound is not present in the first version provided to the drivers due to the belief that sound should only be available if the warning algorithm is working well. Sound is currently planned for deployment later in the field test. For related reasons Contact and Under Wheel will be introduced later too. See the "*Plans for DVI evaluation*" section for additional detail.

No warning yields to warnings immediately. For each side, independently, the order of priority is as follows: Under Bus, Contact, Imminent, Alert, none. A 10% probability of contact (POC) hysteresis with a bias to higher POC is used for level decreases from Imminent or Alert to prevent border oscillations.

1.27. Interface Design and Placement

The DVI design implemented on the ICWS buses integrates the forward and side warning stimuli into a unified display (Figure 71). The forward portion is an adaptation of a similar design utilized for low visibility snow removal operations [Steinfeld00] while the side warnings were developed specifically for this platform and application. This display involves two LED assemblies, one mounted on the left A-pillar and the other mounted on the center window pillar. A control box was installed next to the instrument cluster.

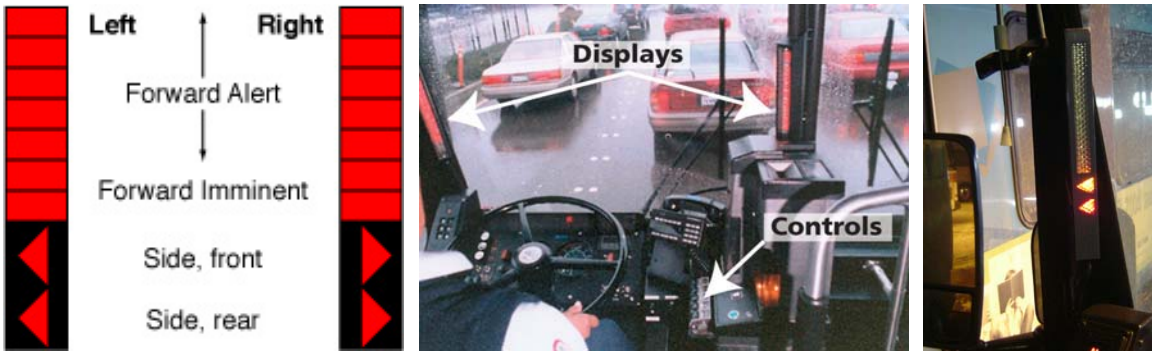


Figure 71 Integrated DVI. The forward LEDs grow downwards with threat level and "aim" at threat. The triangles point towards the relevant mirrors. Bars are mounted on the pillars of the driver's forward window

When viewing the DVI the physical "location" of the driver with respect to the spatial representations of the LEDs is in the middle of the two DVI bars, between the lowermost forward LED and the "Side, front" LEDs. The bars are designed for the window pillars immediately in front of the driver, thus providing a peripheral display that does not obscure the driver's external view of the road scene. The placement also supports rapid checking of the side mirrors – an action much more frequent in transit operations than in regular passenger vehicle operation. Digital DVI outputs are refreshed every 75ms.

Driver controls are mounted as a group in the instrument cluster (Figure 72). Volume, brightness, and warning sensitivity (high, medium, low) provide a level of driver control so that individual differences and environmental factors can be accommodated. However, the system is designed so that drivers are not able to use the volume and brightness adjustments to disable the system. Status lights for the three regions (left, front, right) are also provided for quick identification of system health. The controls include a Contact/Under Bus Override button for acknowledgement of these alarms.

Figure 72 The DVI control box. The toggle sets the sensitivity, the knobs control volume and brightness, and the lights provide status information. Overrides are activated with the red button

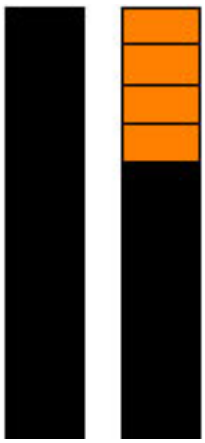
1.28. Examples of DVI Behavior

Forward component

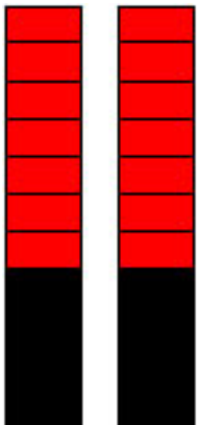
The bars illuminate sequentially from top to bottom to indicate an approaching threat. Depending on how imminent the threat is some combination of the first segment and the first four segments will sequentially illuminate amber. The greater the number of segments illuminated, the higher the threat. To indicate an imminent warning the segments will change color to red and as the threat becomes more time critical will grow to the full length of the display.

The two forward displays show the angle of the greatest threat to the bus. When the left display is lit the object is forward to the left of the bus. When the right display is lit the object is forward to the right of the display. When the object is directly in front of the bus both displays will be lit.

FCWS Alert Mode
Pictorially, forward alert on right side:



FCWS Imminent Mode
Pictorially, Forward imminent with no lateral bias:



Side component

The boundary line between side front and side rear is the plane that passes horizontally through the bus at the front wheel. The mapping of DVI side subcomponents to warnings is as follows:

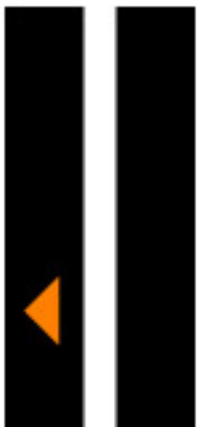
Condition		1	2	3	4	5	6
Left Front (front wheel forward)	Alert	Y					
	Imminent	R				P	
Right Front (front wheel forward)	Alert		Y				
	Imminent		R				P
Left Rear (front wheel back)	Alert			Y			
	Imminent			R		P	
Right Rear (front wheel back)	Alert				Y		
	Imminent				R		P
Contact	Left	BY	BY			A	
	Right			BY	BY		A
Under Wheel (less than 5mph)	Left	BR	BR			A	
	Right			BR	BR		A

Y = Yellow
R = Red
B = Blink at 2 Hz
P = Percussive sound (e.g., chime)
A = Aggressive sound (e.g., buzzer)

Table 26. Mapping of DVI side subcomponents to warnings

In the event that the side component detects an alert level threat it will trigger an Alert Side warning. The triangle shaped LED for the appropriate side and front/rear position illuminates.

SCWS Alert Mode
Pictorially, left side front alert:



In the event that the side component detects an imminent threat it will trigger an Imminent Side warning. The triangle LED for the appropriate side position illuminates red at highest brightness level. The Imminent warning sound plays.

SCWS Imminent Mode
Pictorially, right side rear imminent:

In the event that the side component detects a collision event it will trigger a Contact warning. Both triangles for the appropriate side illuminate yellow at highest brightness level and blink at 2 Hz and the Contact warning sound plays. The driver is then expected to check their mirrors and decide on an appropriate course of action. Should the driver determine that the warning is a false alarm, pressing the Contact/Under Bus Override button will turn off the alarm and suppress contact detection for 10 seconds. As previously mentioned, the button must be fully released before being activated again.

Under Bus warnings are the same as Contact warnings except the triangles are red and the Under Bus warning sound plays. Under Bus warnings only occur at speed less than 5 mph. The driver is then expected to check their mirrors and, if necessary, stop and exit the bus for closer inspection. Should the driver determine that the warning is a false alarm, pressing the Contact/Under Bus Override button will turn off the alarm and suppress contact detection for 10 seconds. As previously mentioned, the button must be fully released before being activated again.

1.29. Plans for DVI Evaluation

On-board collection of driver behavior data will provide insights to the utility of an assistance system and the potential for safety benefit. Such data is most effective when collected during field-testing in real world driving conditions as is currently underway.

DVI evaluation will include a longitudinal human factors analysis of driving behavior. The two states of data collection are (A) Baseline DVI off, but system on and recording and (B) Full System DVI and system on and recording. These states are being cycled for periods of about 3 months where (A) will only be the first few weeks of each cycle. The initial baseline (A_1) will be slightly longer in order to ensure considerable initial baseline data.

This experimental design will allow measurement of system benefit (A_i vs. B_i), behavior shift (A_1 vs. A_2), and system dependence (B_1 vs. A_2). These will be crossed with specific scenarios that are identified as interesting with respect to integrated CWS transit DVIs.

As previously mentioned, sound and the Contact and Under Wheel alarms are not in the initial version (A_1). These features will be deployed in the second or third cycle/version. Besides providing room to allow robust warning, this also permits limited comparison of *visual+audio* and *visual only* transit CWS and the impact of added alarms (e.g., A_1 vs. A_2, etc).

Surveys and interviews will also be employed to collect data on the DVI in order to collect driver, maintenance, and operations perceptions of the system. This technique is

also useful for identifying system weaknesses and areas where training and documentation for the system may need to be modified. Additional insight on extrapolations to larger populations can also be achieved through such documentation.

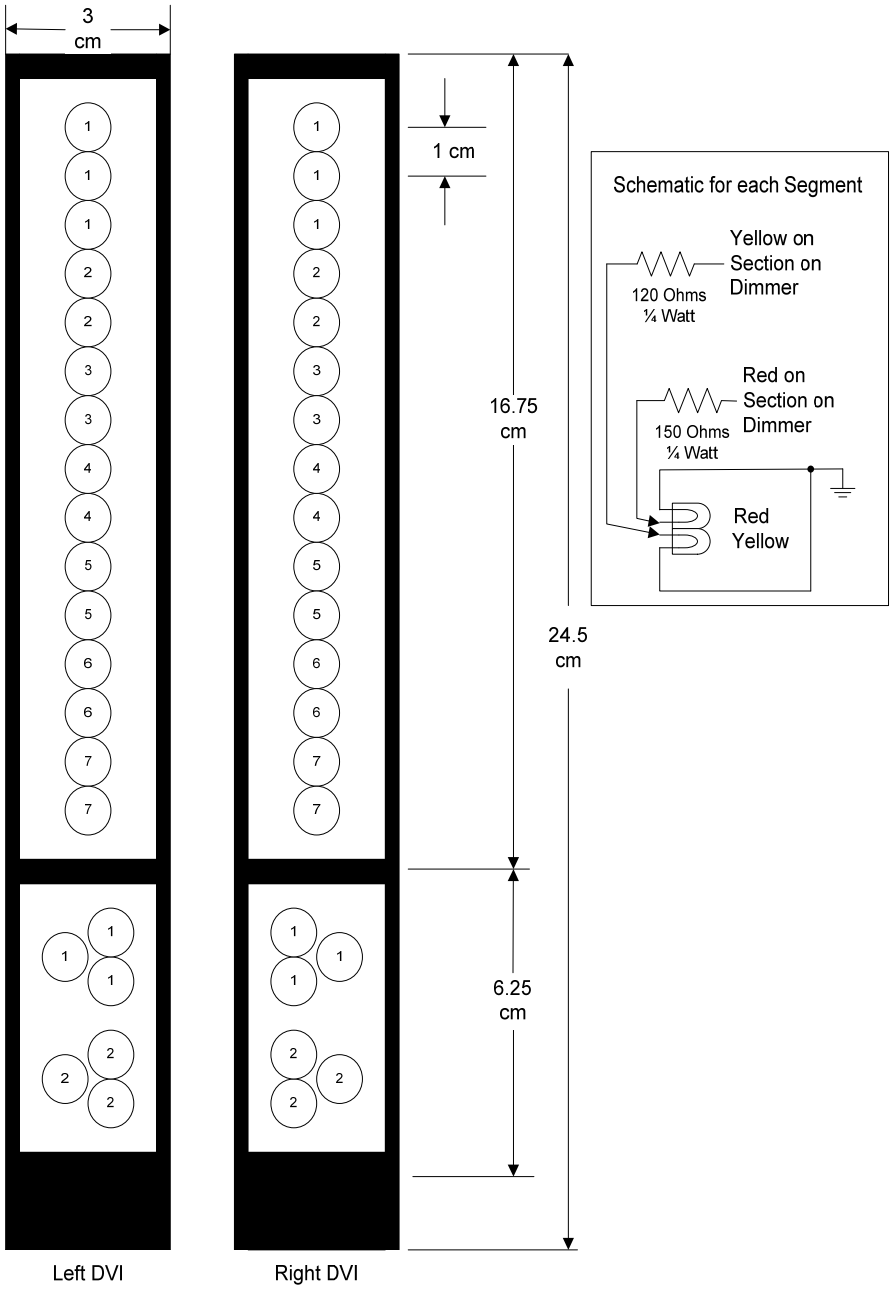


Figure 73 Component diagram of LED assemblies

DATA ANALYSIS AND EVALUATION

1.30. FCWS Data Analysis

The main purpose of the following data analysis is to recreate the warning scenario in detail from a technical point of view. By watching the video clip and analyzing the variables, such as host-bus/target speed/heading/acceleration, we gain an understanding of the accuracy, smoothness and noise characteristic of the measurement, estimation and prediction of the host bus and targets (Bus track and target tracks). By this means, we determine areas in which improvements to components of the system should be made. For example, by looking at the raw speed data we found that the speed measurement resolution was degraded at low speeds. In our new PC104 version, the three-channel speed measurements will improve the measurement resolution at low speeds to address this issue. Generally, improvements in hardware and system software lead to more precise measurement. Improvements in algorithm may lead to better estimation, prediction of the tracks, scenario parsing and threat assessment.

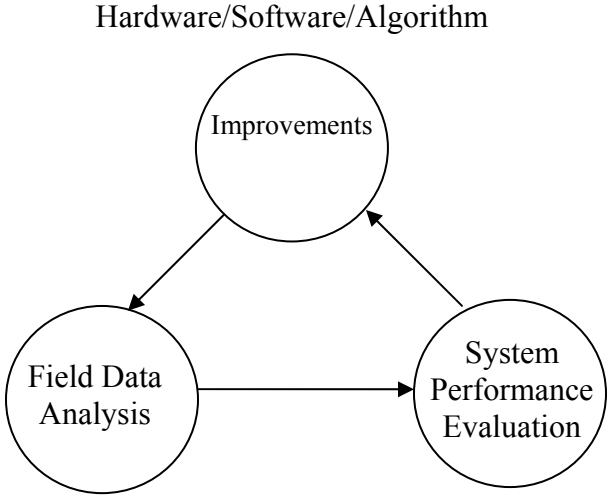


Figure 74 The goal of field data analysis

In this chapter, three typical categories of warning scenarios are analyzed using a three-step quantitive approach. They are:

1. Moving/stopped target in front on a straight road;
2. Stationary roadside target on curved road;
3. Overhead obstacles on declined/flat road

Warnings that fall within the first category are considered correct warnings. The second category warnings are considered false positive or nuisance warnings A nuisance warning is a warning given in a case that a collision is correctly forecasted, but that the operator does not consider the situation to be a true potential threat to the bus. The third category is considered a false warning as the bus is in no danger of hitting these overhead obstacles.. Characteristics of three categories of warnings are analyzed and possible solutions for the later two categories are proposed.

1.30.1 FCWS Three-Step Quantitive Approach

A Three-Step Quantitive Approach for data/warning scenario analysis is developed to analyze the warning scenarios.

1. Check if the weather is good when using LIDAR data to analyze the scenario, and then use a Fault Detection Tool to check the host bus sensor data and the LIDAR data, making sure they are not corrupted.
2. Use Scenario Analysis Tools which recreate the warning scenario by demonstrating the bus/target location/speed/acceleration/heading/ARQ (required acceleration)/raw data in both a snapshot and a trajectory manner.
3. Use a playback tool to review the video clip, showing the warning scenario in video format.

1.30.2 FCWS Warning scenarios categorization

Before any analysis is conducted, a comprehensive check of the data is necessary. If it is raining/snowing or foggy, the LIDAR will not function well and the RADAR will take its place. The host bus sensor data needs to be checked to ensure that all sensors were working properly. Generally, the FCWS warnings fall into three categories based on road geometry and target property (as shown in the following table). Scenario A, B and C are analyzed below with data from Sep. 22, 2003, which was a sunny day. Both host-bus sensors and LIDARs worked well.

Road / Target	Straight	Slightly Curved /Curved	Bridges/Traffic Signs Overhead Declined road/Flat road
Moving/Stopped	**A**	**-**	**-**
Stationary	**-**	**B**	**C**

Table 27. Warning scenario category

1.30.2.1 Scenario A

Scenario A is considered a TRUE WARNING. In the following example shown in Figure 75 of Scenario A, there are some road works ahead of the bus and the leading vehicle was decelerating while the host bus maintained a nearly constant speed. The warning started at 10:21:17. The following camera shots show the "road work ahead" sign and a snap shot of the warning scenario.

In addition to the simulation tool introduced in previous chapters a further simulation program was developed in Matlab^{TM} to recreate warning scenarios. The simulation program is Matlab^{TM} version of the real time program installed on the buses with some slight differences. Using high-level language programming, it is much easier to show the bus/target location/speed/acceleration/heading/ARQ (required acceleration)/raw data and recreate the warning scenario in a trajectory manner. This tool is also used to test new algorithms, add more scenario parsing functions and other sub-routings before integrating them to the off-line simulator using C language.

Frontal camera view **Passenger side camera view**

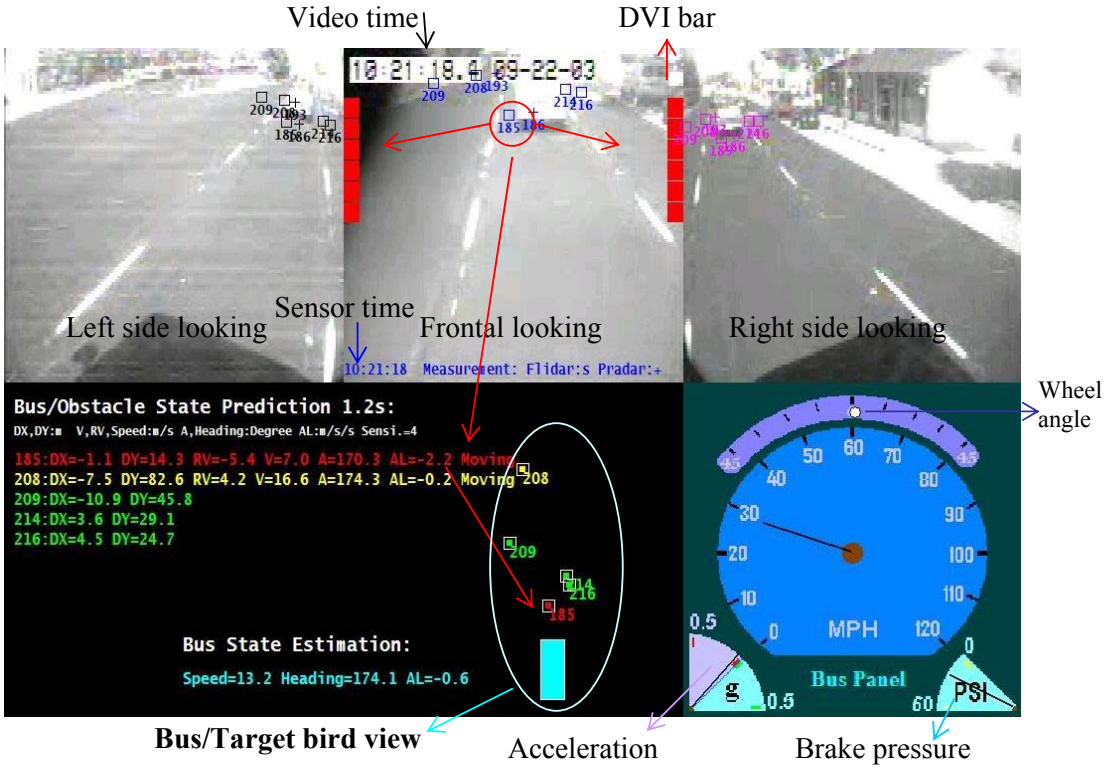


Figure 75 Warning scenario snap shop (note target ID 185: the leading vehicle)

Figure 76 shows ten samples of Target/Host location (top-left figure), Target/Host Speed (top-right figure), and Target/Host acceleration (middle-left figure) before and after the instant a warning was issued. Figure 76 also shows 20 samples of brake pressure before and after (middle-right figure), and required deceleration (bottom figure) before and after the exact instant when the warning was issued.

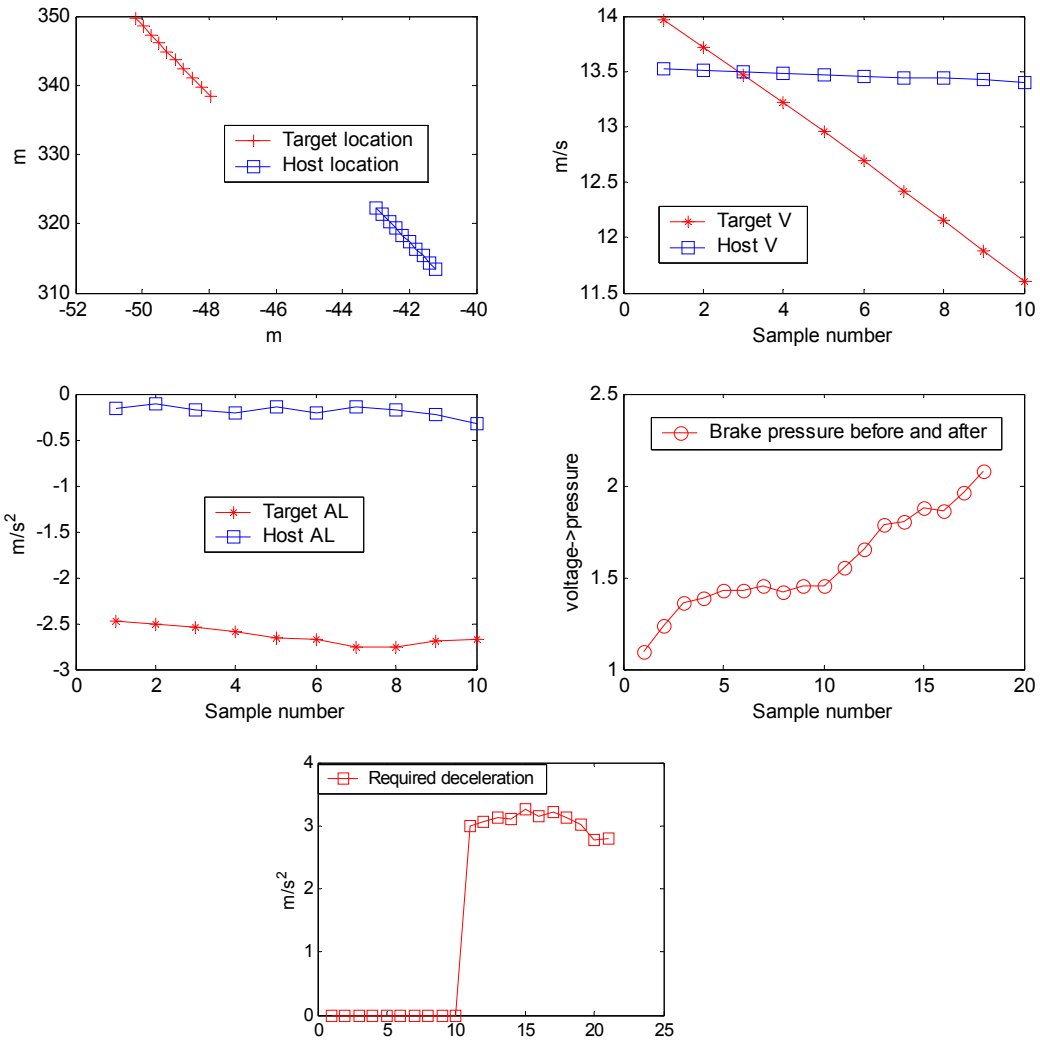


Figure 76 Technical variables

As we can see from the figures, the target vehicle was decelerating (the red line in the top-right figure keeps dropping) at a deceleration rate of about 2.5m/s/s. The bus, however, maintained an almost constant speed (dropping slightly) of about 13.5m/s with a deceleration ranging from 0 to 0.5m/s/s. At the tenth sample, the required deceleration exceeded the threshold of 1.8m/s/s, which means if the bus continues at the current driving status without more deceleration (for example, pressing the brakes more), 1.2 second later, a deceleration greater than 1.8m/s/s will be needed to avoid a collision. This is considered a dangerous situation. Therefore, the warning was issued starting from the tenth sample (at 10:21:17). (The sample interval is 75ms). From the brake pressure

figure, we can see that the brake pressure increased dramatically after the tenth sample, because the driver did press the brakes harder after the warning was issued.

The warning continued at 10:21:18 and 10:21:19:

Frontal view **Passenger Side view**

The warning ended at 10:21:20.

Frontal view **Passenger Side view**

Figure 77 Target vehicle Decelerating

Figure 78 shows ten samples of Target/Host location (top-left figure), Target/Host Speed (top-right figure), and Target/Host acceleration (middle-left figure) before and after the moment the warning ended. The figures also show 20 samples of brake pressure before and after (middle-right figure), and required deceleration (bottom figure) before and after the exact moment when the warning ended.

As we can see from Figure 78 , although both the host bus and the target vehicle are decelerating, the host bus speed is greater than the target vehicle (See the top right figure, both the blue line and the red line are dropping), the deceleration of the bus is changing from less than the target vehicle to greater than the target vehicle (see the crossing of the

two lines in middle left figure). At the tenth sample, the deceleration of the bus is about 0.4m/s/s greater than the target vehicle (a minus sign should be added if referring to the acceleration), the required deceleration is 1.5m/s/s (below the 1.8m/s/s threshold), which means if the bus continues its current driving status, 1.2 seconds later, the situation will no longer be considered dangerous. Therefore at this exact moment, the warning ended. From the brake pressure figure, we can see that the brake pressure started going down after the tenth sample and decreased dramatically from the 15th sampling point, since the driver did release the brakes after the warning ended.

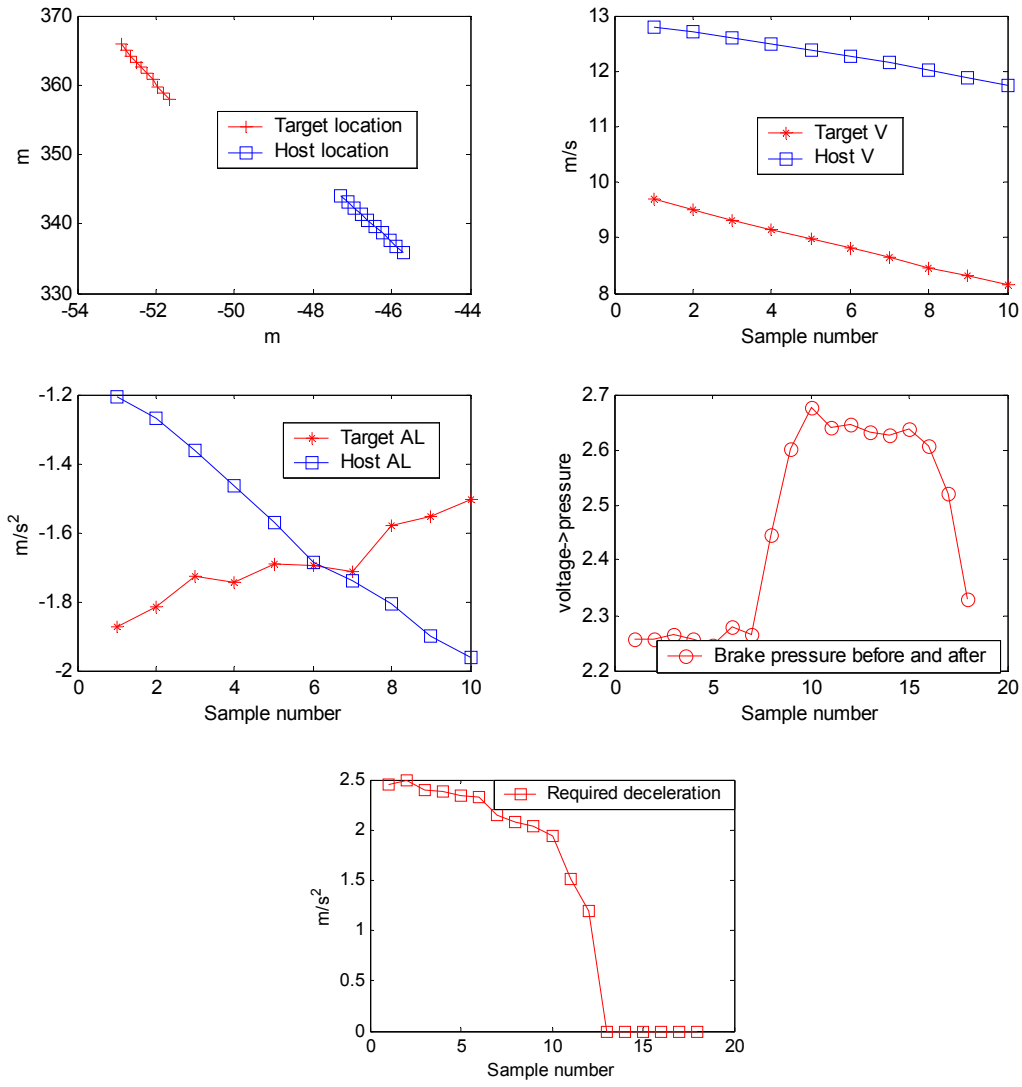


Figure 78 Technical variables

1.30.2.2 Scenario B

Scenario B is considered a NUISANCE WARNING. In this case, roadside objects, especially those hard reflective traffic signs and guardrails could be dangerous if the bus continued at its current heading, however in most cases like these the driver is aware of the street furniture, so a warning is not warranted. Minimizing the occurrence of these types of warnings has been one of the main issues in the development of the FCWS system. In the example, shown below in Figure 79, a warning was issued at 9:53:04. It was triggered by guardrails on the right side, not the vehicle on the other lane as shown in the following analysis.

Frontal view **Passenger Side view**

The warning continued at 09:53:06:

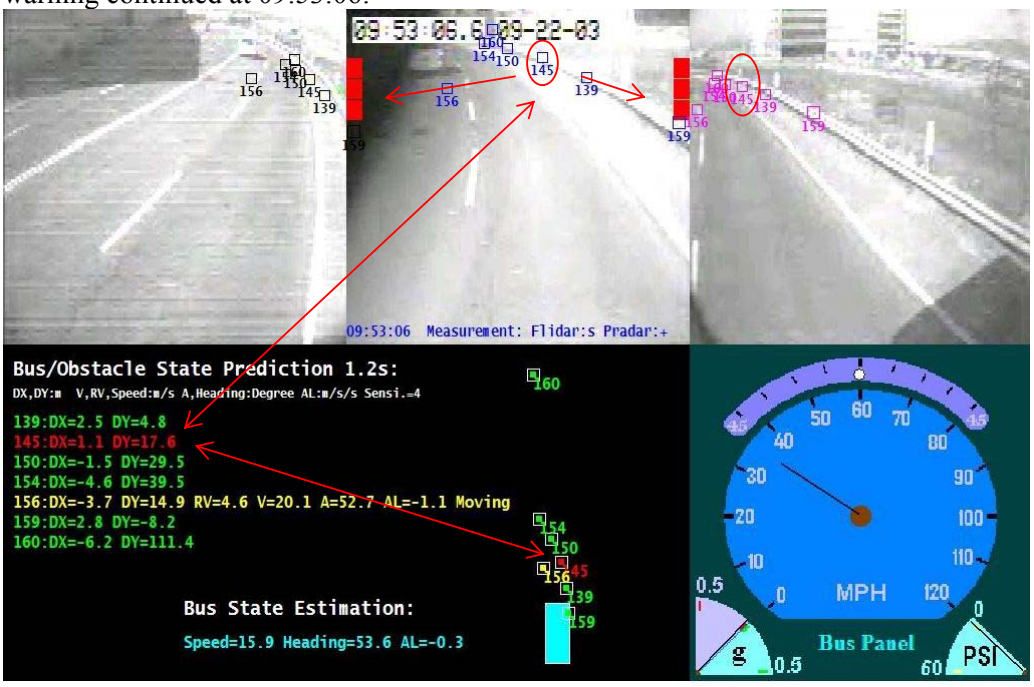


Figure 79 Warning scenario snap shot. Note the warning trigger- target ID 145 in red, not target ID 156 in yellow which is the leading vehicle

The warning ended at 9:53:08.

Frontal view **Passenger Side view**

Figure 80 Nuisance warning ends

Basically, when the road is curved or the bus driver makes a lane change (to the right-most lane), if there are stationary roadside objects ahead, especially those hard reflective traffic signs and/or guardrails, which are sensitive targets for the LIDARs, it will look as if the bus were heading towards those objects, also, since the system needs to predict 1.2 second ahead to compensate the sensor delay and give the driver enough time to react, a warning will be issued. It is called a nuisance alarm because if the driver were distracted, thus did not change the bus's heading but remained going straight (in the curved road situation) to the right (in the lane changing situation), the crash would happen. However, most of the time, the diver is vigilant and will change the heading when confronted with curved roads and will go straight again after changing lanes. These warnings are explainable but some drivers may find them annoying.

The trajectory of the bus and the target are plotted in Figure 81. Blue squares represent the bus trajectory, green squares represent the car running on the left side, yellow squares represent road side guardrails, "yellow" represents the object that triggered the warning. Note that the up and down motion of the guardrail was caused by noise in the measurement. As is shown in the figure, the car on the left did not trigger the warning. (The "green" represents a "safe" track, which did not trigger the warning). This figure also shows why a comprehensive analysis tool is a must. Without the tool it is hard to determine the correct cause of the warning.

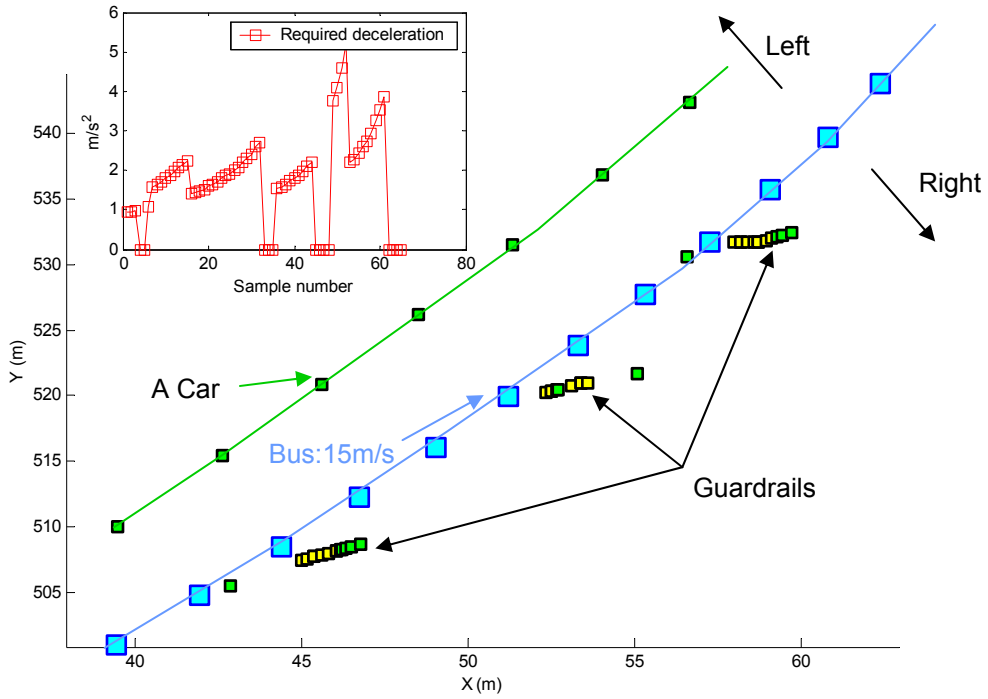


Figure 81 Trajectories of the bus and the targets around

There are three areas where additional information could help us solve the above challenges.

1. ***Road geometry.***

   For example, digital maps may tell us the curvature of the road ahead and help the system recognize if the obstacles are in lane or out of lane. If it is out of the lane, we could apply a smaller probability factor.

2. ***Driver status.***

   This information is hard to get. If the driver is vigilant, these nuisance warnings could be annoying or distracting. However, if the driver is distracted, these nuisance warnings will be good warnings. Given the driver status, the system could utilize this information and decide if it should issue these warnings.

3. ***Target property and crash data analysis result.***

   If street furniture such as guardrails could be identified by use of a GPS/digital map system, and crash data analysis showed that there is a very small possibility of a bus hitting a guardrail then the system could apply a small probability factor when seeing

those guardrails in a curved road, which may dramatically reduce the nuisance-warning rate.

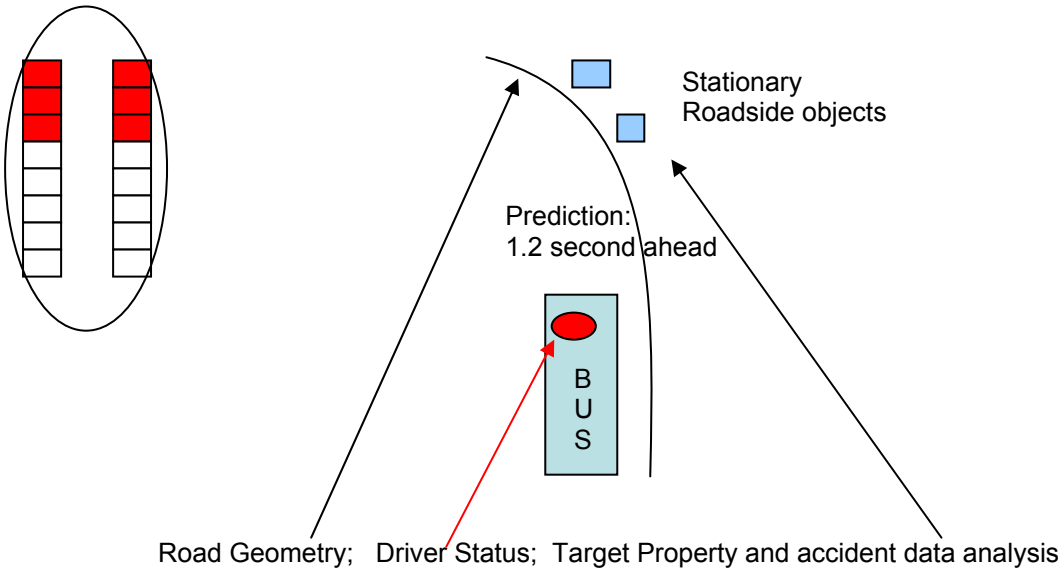


Figure 82 Problem: stationary objects along curved road

1.30.2.3 Scenario C

Scenario C is considered a FALSE WARNING. In the case shown below, the LIDAR detects objects right in front and it looks as if the bus were going straight towards the overhead obstacles. If it were not for the declined road, the LIDAR might not see the bridges since they are higher than the bus (above the ground). The required deceleration will rise sharply as the bus maintains constant speed while passing the bridge.

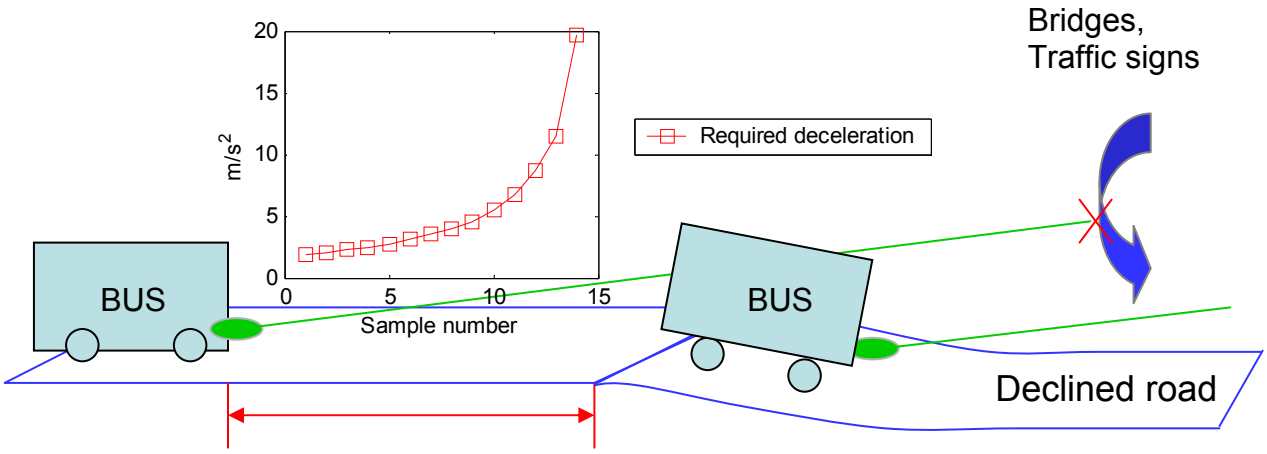


Figure 83 Problem: Overhead obstacles

As shown above, in this particular case, the declining road and the overhead bridges/traffic signs faked a threat to the host bus. The system incorrectly predicted a potential collision, as it did not have the information about the road geometry. Another similar case could occur with overhead traffic signs that are higher than the bus on a flat road. As the bus may pitch slightly due to different road surface condition, it is hard for the sensor to get the accurate height information of those "obstacles". Detailed road geometry information may help us solve this problem. With the help of a detailed digital map, if the system knows that the target detected right in front is an overhead bridge and that the bus is on a declining road, or the system knows that the detected target is a traffic sign hung above, it will not issue the false warning.

1.30.3 FCWS Summary

The FCWS warning scenarios are categorized and analyzed using a three-step quantitative approach. The three scenarios include: moving/stopped target ahead on straight road; stationary target roadside on curved road; overhead obstacles on declining/flat road are analyzed. Improvement was made to the algorithm to include features that turn the nuisance warning to a friendly reminder. It is believed that, road geometry information (e.g., more precise GPS and digital map system), driver status information, target properties and crash data analysis, some of the nuisance induced by curved roads and overhead obstacle problems could be overcome.

1.31. SCWS Data Analysis

Custom analysis tools have been developed for examination of data generated by the SCWS component. These tools can be used in conjunction with the SCWS Data Replay tool for visual inspection of events and/or system behavior. There are two types of tools in use: driver behavior analysis and system debugging and development. Both will be described here.

1.31.1 Driver behavior analysis

Part of the evaluation of the collision warning system is to assess if and how the behavior of the driver changes. There are several ways of doing this, e.g. one can monitor the frequency and severity of dangerous situations. This can be done for complete runs or for

particular maneuvers. We developed analysis tools with which we can pick out particular maneuvers and accumulate relevant statistics.

The analysis tools are flexible and can be customized to analyze many different maneuvers. From a high level, the data and video collected on the bus during operation is stored in a RAID on the CMU campus (Figure 84). This data, which is read-only, is fed to the Analysis Tools – custom filters and collators that compose data from the RAID based on specified Event Definitions. Compiled Event Data consists of data snippets collated as a series of events. Each event has a bookmark for the beginning time stamp of the event. Experimenters can review the video (stored on the RAID) for each event by jumping to the bookmark in question. For the purposes of behavior analysis, the Analysis Tools extract and compute selected driver behavior data for subsequent processing in traditional statistical analysis software.

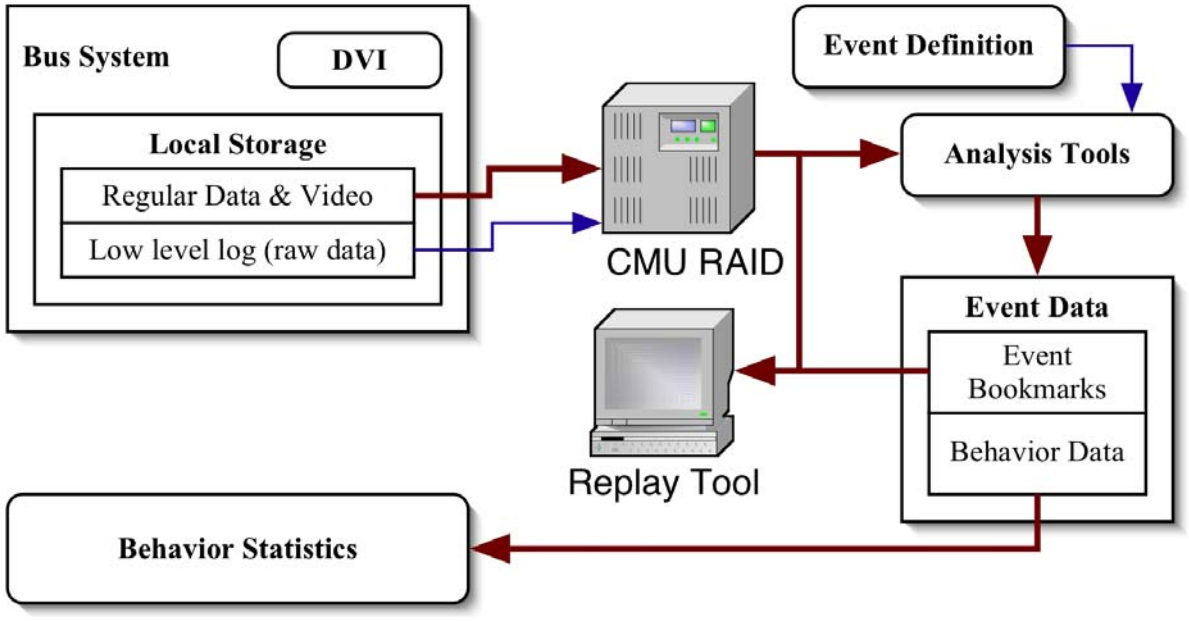


Figure 84 Data flow for driver behavior analysis

The most important part of this process is the specification of the event the experimenter wishes to examine. As an example, we will look at the evaluation measure “Time within each CWS DVI category” (see Table 28. Evaluation Metrics (MOE's)]) for the scenario

where a bus pulls out of a bus stop and we want to monitor the level of danger. We specify the scenario in the following way:

1. The **priming condition** "bus stopped and door is open" has to be fulfilled. This ensures that the bus has stopped at a bus stop. Events need to be specified mathematically. As such, we would specify that bus speed is below 2 mph and the door open flag is true.
2. When the **trigger condition** "bus starts to move" is fulfilled (e.g., speed > 2mph), we start to record data: time, speed, turning radius, and probability of collision (see section 1.18 SCWS Warning algorithm). The probability of collision is the measure of danger.
3. The **stop condition** is fulfilled when the bus has traveled a set distance and we stop to record data. For example, we may indicate that the bus has traveled more than 5 m.

The Analysis Tools then compile the recorded event data and additional (specified) optional computations may be run to see what danger levels are present for other system or custom sensitivity settings. Bookmarks are also stored so experimenters can quickly jump to the relevant events. The output data file can then be imported into any statistics program (tab separated values) and time within DVI category can be computed. Independent variables, like driver set sensitivity level and location, can be included in this file so that behavior analysis can be parsed accordingly. A primary independent variable, test or baseline data collection (DVI enabled/disabled) can be used for direct analysis of system effectiveness.

Should macro scale data be desired (e.g., average warning level for an entire month, regardless of scenario) then the event definition can be set to a wide level. For example, the priming condition could be system is powered up, the trigger condition be the departure of the bus from the bus yard, and the stop condition the arrival at the bus yard.

The analysis tools are also used to monitor the system for unusual statistics that can be due to system failures. Metrics of this type include wild fluctuations, infeasible warnings, lack of warnings, etc.

The behavior analysis example here is one of the many proposed metrics that will be examined in the evaluation phase of this program. The following matrix lists the additional evaluation criteria that will use these tools. An evaluation report will be written at the conclusion of this program reflecting these metrics

Task	Performance MOE	Before / After MOE	Measures of Interest
2 Closed Course	Time of alert Time of warning Time of notify Latency Rainfall performance Environmental effects (salt spray, etc)		Daily precipitation Daily High/Low temps
3 Detect Analysis	True positives False negatives True negatives False positives		Fault tree distribution Scenarios parsing (multiple events where at least one is bad)
4 Driving Behavior*		Behavior when within CWS DVI activation range Time within each CWS DVI category [alert, warn] Hard accelerations (braking & swerving) Frequency of warnings over time Normal following distances (front) Probability of collision over time (side)	Driver sensitivity setting
5 Surveys Interviews	Nuisance alarms Driver sensitivity ratings/reports Driver and management perception of safety benefit Satisfaction with system performance Perception of system accuracy	Did system prevent an accident? Self-reports of alterations in driving behavior	Relaying of passenger queries and comments
6 System Failures	MTBF Software detected component failures (perform appropriate actions upon failure) MTTR Operability Time [correct, degraded, incorrect, not at all] vs. On time vs. Vehicle deployed time [Agency data]		Failure mode taxonomy Component repair cost

Table 28. Evaluation Metrics (MOE's)

* Binned by *DVI off* (baseline) and *DVI on* time periods

1.31.2 System debugging and development

As previously mentioned, analysis tools for testing new algorithms are also used. These involve generation of new data that are used in place of collected data (Figure 85). Raw low-level data from the lasers or other sensors can be used to simulate new data, and subsequently, new warnings or object traces. These can be visualized in the SCWS Data Replay tool for performance assessment or compared directly to the real counterparts.

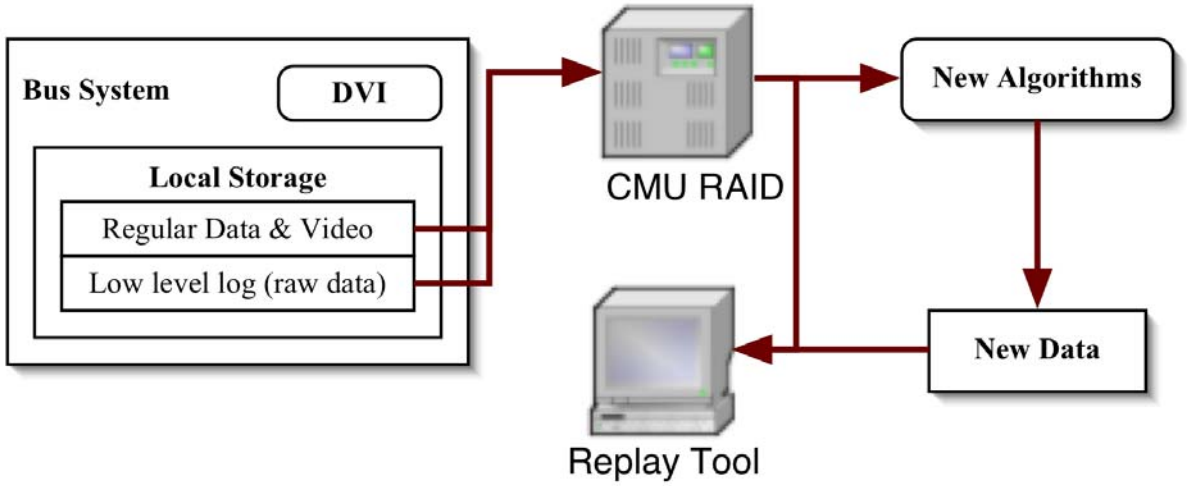


Figure 85 Fusion of new data from test algorithms with real data

This process is especially powerful in testing new DATMO and SCWS warning algorithms. The ability to view the results fused with original video and supporting data (e.g., speed, etc.) provides a good first pass for qualitative performance judgments before conducting labor-intensive comparison analyses. For example, a particular data segment may include a stereotypical false alarm that algorithm developers are attempting to prevent. Visualization of algorithm performance during this segment can be especially telling when trying to determine the root causes of the false alarm and progress towards handling them.

CALIBRATION AND TESTING

1.32. SICK Laser Scanner

1.32.1 SICK resolution and accuracy

The basic properties of the laser scanner are:

Angular range:	180^o
Angular resolution:	0.5^o or 1.0^o
Range:	up to 80 m
Range resolution and accuracy:	1 cm
Update rate:	37.5 Hz or 75 Hz (depending on angular resolution)

The manufacturer claims that the resolution and accuracy of the SICK laser scanner is 1 cm. In the following sections we will test this claim.

1.32.2 Definition of terms

Following are important terms for our discussion:

Resolution: Minimum separation necessary to distinguish two objects or minimum displacement necessary to notice movement of an object.

Error, uncertainty, deviation, accuracy: Synonyms for differences between measured and actual property.

Standard deviation: Quadratic average of the differences:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}} \qquad (1)$$

where n is the number of measurements x_i and $\bar{x}$ is the mean of the measurements.

If the function *e(x)* describes the error distribution, the standard deviation is:

$$\sigma = [\frac{1}{N}\int e(x)^2 dx]^{\frac{1}{2}} \qquad (2)$$

with the normalization factor

$$N = \int e(x)dx \qquad (3)$$

1.32.3 Error characterization

The basic nature of errors of a sensor can be inferred from its working principle. The laser scanner scans a range of angles and for each angle it determines the distance to the closest object by time-of-flight (TOF). Since the sensor scans the angles, it makes only sense to talk about its angular resolution (which can be chosen to be 0.5^o or 1^o) but not its accuracy. According to the manufacturer, its range resolution and error is 1 cm, independent of the absolute distance. When a distance *d* is measured, the actual distance is *d* ± 0.5 cm with all distances within this error range being equally likely. For later comparison, it is useful to calculate the standard deviation (see Equation (2)):

$$\Delta d_s = \sqrt{1/3} \cdot 0.5cm = 0.29cm \quad (4)$$

A discussion about the characteristics of a similar laser scanner can be found in the footnote below.[22]

1.32.4 Experimental confirmation of resolution

In order to confirm the claims about measurement errors in the previous section, a straight fixed object was placed in front of the sensor and the distance to the object was measured several consecutive times. Figure 86 shows this data for the laser scanner and a linear fit through the data. The object extends for an angular range of about 50^o. Upon close inspection of the points one can notice small steps, which are the result of the 1cm resolution mentioned in the previous section.

[22] Jensfelt and Christensen. "Pose Tracking Using Laser Scanning and Minimalistic Environmental Models." IEEE transactions on robotics and automation Vol. 17, No.2. April 2001.

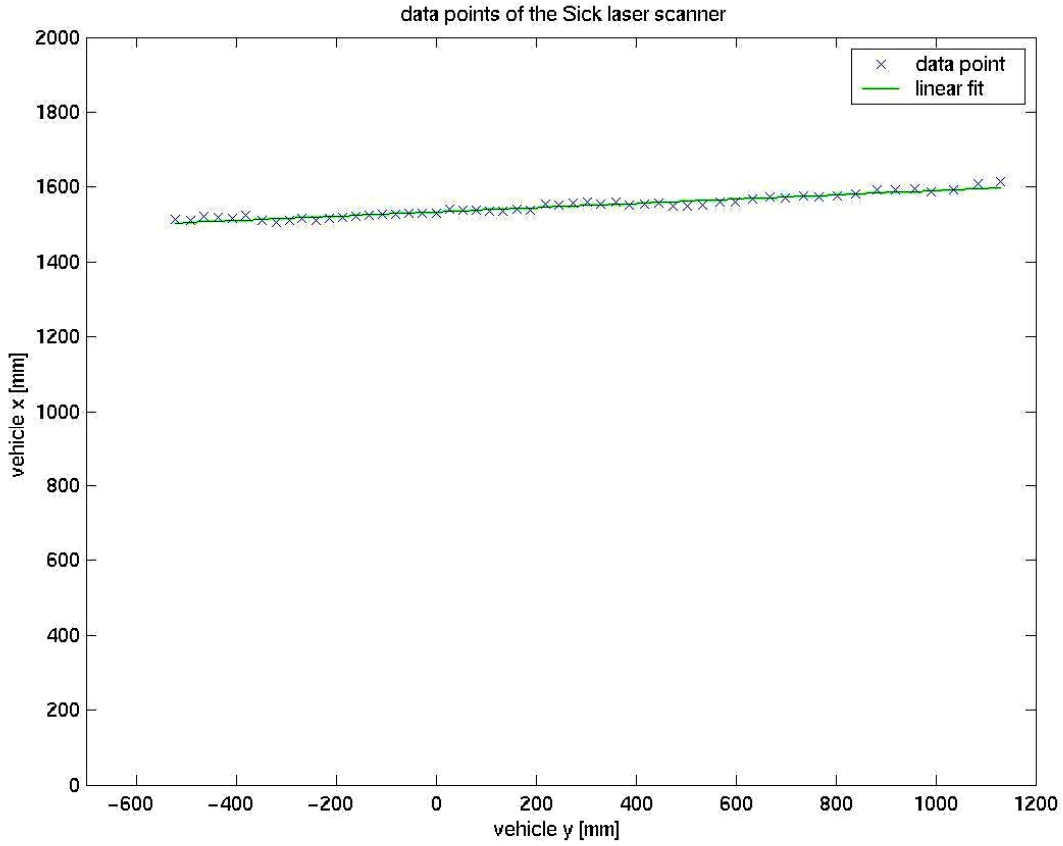


Figure 86 Distances to a straight object measured by the laser scanner. The green line is the linear fit to the data points.

The standard deviation of the points to the linear fit is 0.68 cm, larger than expected from Equation (4). But the object was not perfectly straight and it is likely that the difference can be attributed to this reason.

Next the measurements of the same location at different times were compared. The standard deviation of points measured at different times is shown in Figure 87 under the label "temporal".

Also shown for each location is the (temporal) mean of the distance minus the average of its neighbors:

$$\Delta d_n = \overline{d_n - \frac{(d_{n+1} + d_{n-1})}{2}} \qquad (7)$$

Both quantities are 1cm or less for all points, consistent with the resolution of 1cm.

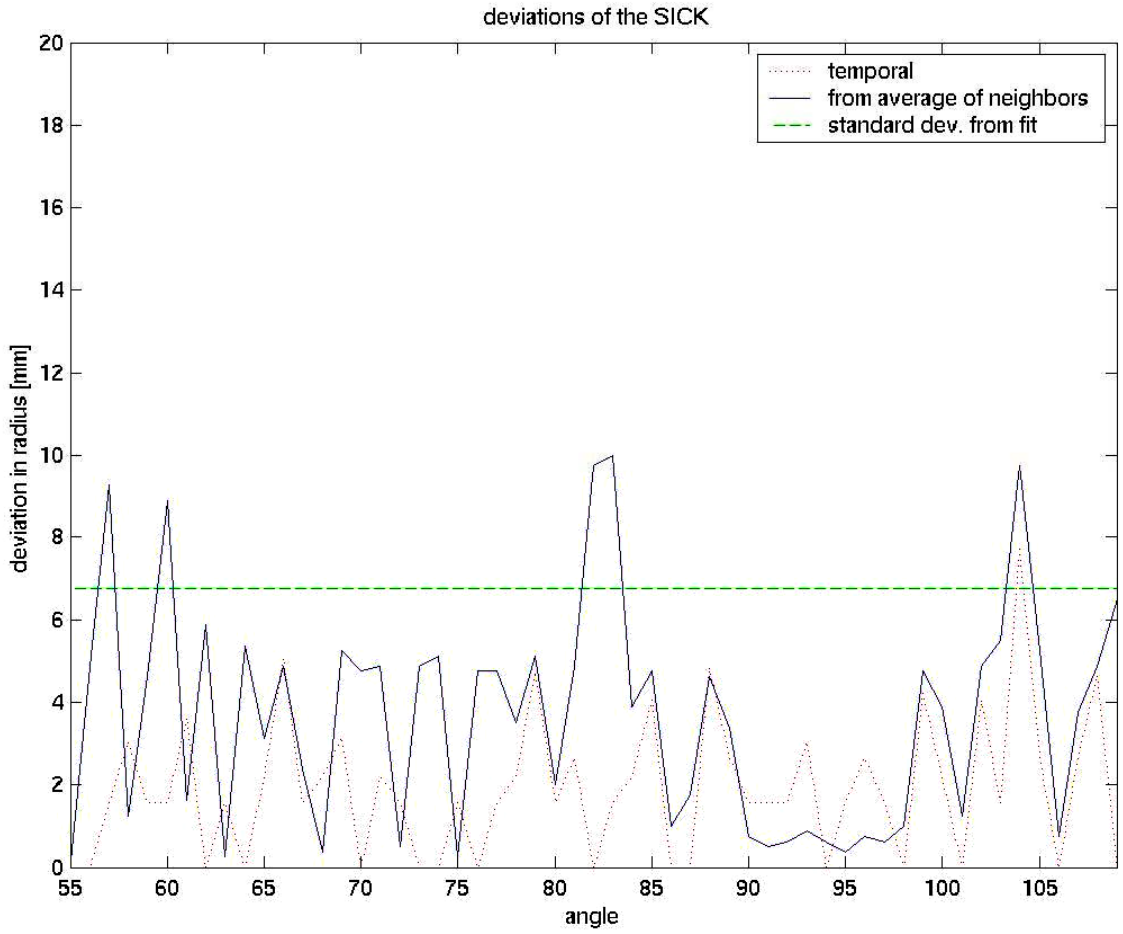


Figure 87 Error comparisons for the laser scanner.

1.32.5 Experimental confirmation of accuracy

Next we want to test for accuracy for different distances. A target was placed at several distances between 1 m and 45 m. The distance was first measured with a measuring tape and then compared to the distance measured by the SICK laser scanner. The result can be seen in Figure 88 the standard deviation of the difference between the two measured distances is 1.5 cm. This deviation contains the uncertainties related to the target. The target was not entirely flat and it was only eyeballed to ensure that it is vertical. The standard deviation of 1.5 cm can therefore be considered consistent with an accuracy of the SICK of 0.3 cm (Equation 4) over the range of 45 m.

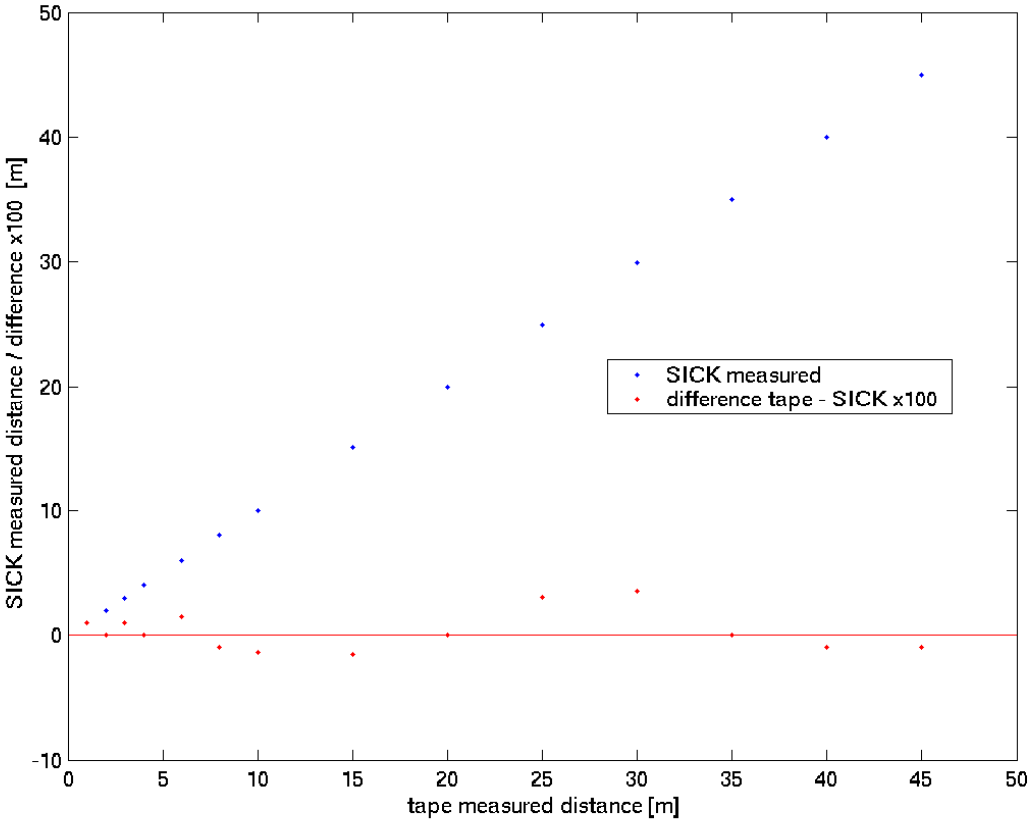


Figure 88 Distance to the target measured by the SICK versus measured by tape.

It needs to be mentioned, that the SICK is only accurate when the light pulse hits a flat surface. It has difficulties at edges of objects when the footprint of the laser pulse hits targets at different distances. In that case it can produce a ghost point in-between the two targets.

1.32.6 Summary

The claim of 1 cm accuracy and resolution has been confirmed for ranges of distances (45 m), angles (50^o), and time.

1.33. Calibration of Scanner Position and Orientation

For the side collision warning system two SICK laser scanners were mounted on the bus, one for each side. The position of the sensor with respect to the bus coordinate frame was determined using a measuring tape. The laser scanner was mounted on the bus in such a

way, that the orientation of its internal reference frames is either parallel or perpendicular to the axis of the bus reference frame. This way the rotation from one to the other coordinate system is easy to be determined. Small deviation from exact alignment of the yaw angle were determined in two different ways, the first was overlaying the scanner data on a calibrated image and the second was comparing the bus speed with the residual speed of fixed objects.

1.33.1 Calibration by overlay

In section 1.11.1.3 Calibration of sensors we describe how several sensors are calibrated together and finally their data are overlaid on an image. See the first figure in that section. If the yaw of the laser scanner is not properly aligned it will show up as a misalignment in the overlay. The yaw can be corrected by simple trial and error until the overlay is satisfactory.

1.33.2 Calibration by residual speed of fixed objects

When a bus drives by a fixed object, DATMO will find that the relative speed of the object is equal but opposite of the bus speed. If the laser scanner is not exactly aligned, then the relative speed of the object is not exactly opposite (*i.e.* rotated by 180^o), instead it is rotated by more or less than 180^o. This effect was seen during the evaluation of Detecting and Tracking of Moving Objects (DATMO), (see section 1.35.2.2 Error characterization of the full DATMO). We studied the potential of this effect in detail to see, if it can be used for automatic calibration of the laser scanner. This study can be found in the next section Automatic external calibration of a laser scanner.

1.34. Automatic External Calibration of a Laser Scanner

It is important to know the position and orientation of the sensors mounted on a test vehicle in order to be able to have all the available data in a common reference frame. The process of determining the position and orientation is called external calibration. It is desirable to make that process as easy as possible, in the best case it should be done automatically by the system. In this report we discuss the possibility of automatically calibrating a laser scanner.

1.34.1 Calibration approach

Our approach is to compare the dynamic variables of the vehicle or vehicle state (velocity, turning rate, etc.) with those of the sensor. At first we will only consider the two-dimensional case, i.e. the vehicle travels on a plane surface and the field-of-view of the laser scanner is parallel to that plane.

1.34.1.1 Determining vehicle state

The vehicle state is determined by odometry (change in position) and by a gyroscope (change in orientation). Usually there is a bicycle model incorporated in the vehicle state, namely that the lateral velocity (vy_v in the definition below) is zero. In the derivation of the method this assumption is not being made, therefore the method is general and can also be used to calibrate two laser scanners to each other.

1.34.1.2 Determining external sensor state

We are using SLAM (Simultaneous Localization And Mapping) and DATMO algorithms [Wang and Thorpe][23] to determine the external sensor state. In SLAM successive laser scanner readings of the surroundings from a moving vehicle are compared and matched to each other. If the surrounding is fixed, the movement of the vehicle can be inferred from the change in the sensor reading and the matched data gives a map of the surrounding and how the sensor has moved from scan to scan. It is therefore possible to determine for each time step the position and orientation of the sensor relative to its initial position and orientation. If moving objects are present, they need to be filtered out and tracked with DATMO. Details about the algorithm can be found in the publication.[24]

1.34.1.3 Reference system

The moving (!) reference system is defined as follows:

[23] Wang and Thorpe. "Simultaneous Localization and Mapping with Detection and Tracking of Moving Objects." IEEE International Conference on Robotics and Automation. May 2002.

[24] op. cit.

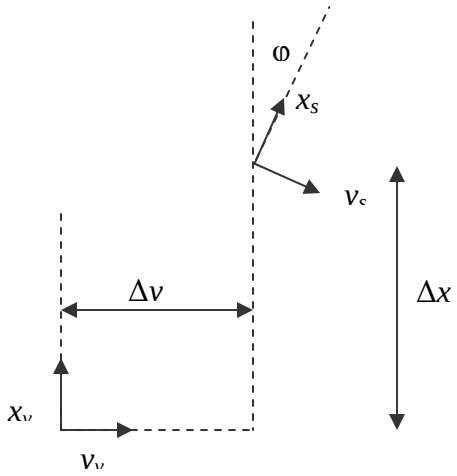


Figure 89 Vehicle and sensor coordinate frames

The vehicle coordinate frame is (x_v , y_v). The sensor coordinate frame (x_s , y_s) has its origin at (Δx, Δy) and is rotated by the angle φ. The relationship between a point in the sensor frame and the same point in the vehicle frame is:

Equation 22 $$x_v = x_s \cos(\varphi) - y_s \sin(\varphi) + \Delta x$$

Equation 23 $$y_v = x_s \sin(\varphi) + y_s \cos(\varphi) + \Delta y$$

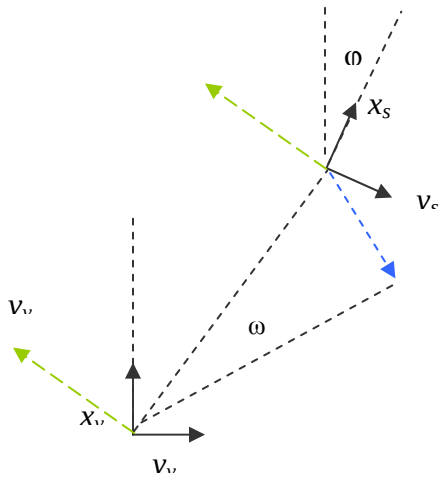


Figure 90 Moving vehicle and sensor coordinate frames

If the vehicle is traveling with the velocity $v_v = (vx_v, vy_v)$ and rotating with the angular velocity ω, then the origin of the sensor is rotating with the same angular velocity ω, but traveling with the velocity $v_s = (vx_s, vy_s)$ which is dependent on v_v, ω, Δx, and Δy:

Equation 24
$$vx_s = (vx_v - \Delta y \cdot \omega)\cos(\varphi) + (vy_v + \Delta x \cdot \omega)\sin(\varphi)$$

Equation 25
$$vy_s = -(vx_v - \Delta y \cdot \omega)\sin(\varphi) + (vy_v + \Delta x \cdot \omega)\cos(\varphi)$$

Having only two equations, the three unknowns Δx, Δy, and φ cannot be determined with one measurement. One needs to make measurements for different translational and rotational velocities.

1.34.2 Example implementation

We drove the Navlab 11 vehicle a distance of about 40 meters on a course of curves and straight lines and recorded for each the path and velocities (Figure 91). As expected, the angular velocities of sensor and vehicle are very similar; their difference is due to measurement errors.

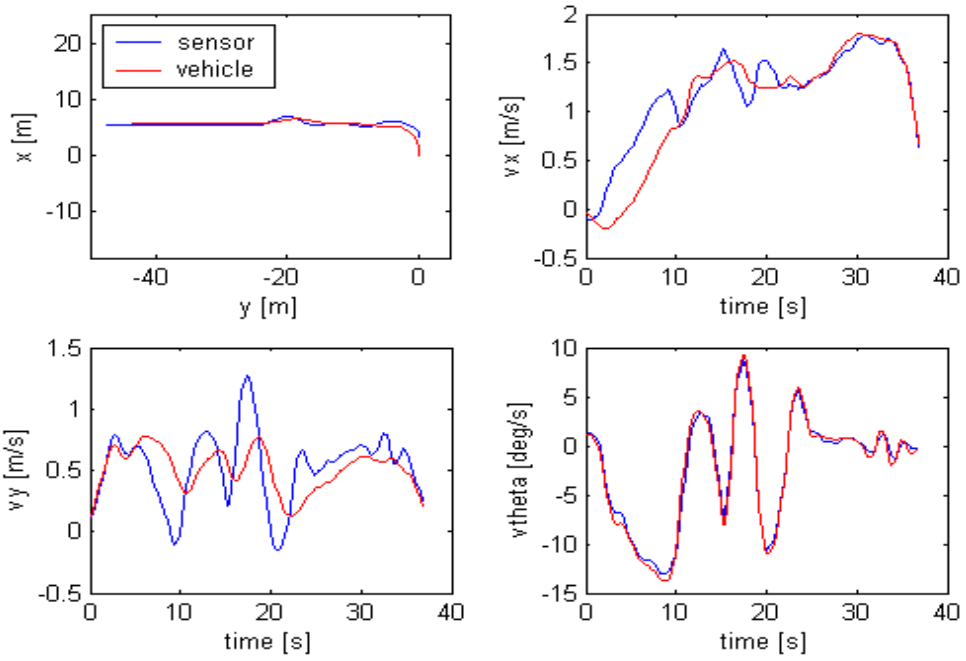


Figure 91 The path and the velocities recorded by the vehicle and the sensor in the fixed coordinate frame. The two paths are aligned according to the result of offset and orientation of the sensor.

1.34.2.1 Initial step

First we selected all the instances where the angular velocity is close to zero ($\omega < 2^o/s$). If we assume the angular velocity is exactly zero, Equation 24 and Equation 25 become:

Equation 26 $$vx_s = vx_v \cos(\varphi) + vy_v \sin(\varphi)$$

Equation 27 $$vy_s = -vx_v \sin(\varphi) + vy_v \cos(\varphi)$$

Which is simply the rotation equations and φ is therefore the angle between v_s and v_v. We calculate this angle for each of the selected instances and then get our initial estimate φ_0 from their mean value. In Section 1.34.4 we discuss various methods besides the mean value which can be used to determine φ_0.

1.34.2.2 Iterations

If one has φ, the values of Δx and Δy can be determined from Equation 24 and Equation 25:

Equation 28 $$\Delta x = \frac{1}{\omega}(vy_s \cos(\varphi) + vx_s \sin(\varphi) - vy_v)$$

Equation 29 $$\Delta y = \frac{1}{\omega}(vy_s \sin(\varphi) - vx_s \cos(\varphi) + vx_v)$$

On the other hand, if Δx and Δy are known, φ is:

Equation 30 $$\varphi = \arctan(vy_v + \Delta x\omega, vx_v - \Delta y\omega) - \arctan(vy_s, vx_s)$$

The three calibration parameters can now be determined iteratively:

1. Using all instances with small turning radii (< 10 m) and the previously determined value of φ determine Δx and Δy by forming the median of their distributions.
2. Using all instances with large turning radii (> 60 m) and the previously determined value Δx and Δy determine φ by forming the median of their distributions.
3. Repeat 1. and 2. until convergence is achieved.

In our example 3 iterations were sufficient.

1.34.2.3 Results

The distributions of Δx, Δy, and φ for the last iteration can be seen in Figure 92. The resulting values using the different methods:

sensor orientation	median: 0.50 deg	mean: 0.37 deg	std: 2.60 deg	error: 0.36 deg
gaussian fit		center: 0.49 deg	sigma: 1.02 deg	error: 0.14 deg
sensor del x	median: 3.439 m	mean: 3.424 m	std: 0.844 m	error: 0.113 m
gaussian fit		center: 3.393 m	sigma: 0.158 m	error: 0.021 m
sensor del y	median: -0.129 m	mean: -0.153 m	std: 0.296 m	error: 0.040 m
gaussian fit		center: -0.026 m	sigma: 0.304 m	error: 0.041 m

Table 29. Values for sensor orientation, Δx, and Δy

Summary:

$\Delta x = (3.39 +- 0.02)$ m $\quad \Delta y = -(0.03 +- 0.04)$ m $\quad \varphi = (0.49 +- 0.14)^o$

Remember that the errors are purely statistical and do not include systematic errors.

The following positions were measured with measuring tape, for the angle a target was placed directly in front of the vehicle and measured with the laser scanner itself:

$\Delta x = (3.35+-0.01)$ m $\qquad \Delta y = (0.0+-0.01)$ m $\qquad \varphi = (0.75+-0.5)^o$

The errors are estimates.

1.34.3 Special case: bicycle model

If one places the center of the coordinate at the middle of the rear axle of the vehicle, then there is never a lateral movement (vy_v=0). This fact simplifies the equations.
If the vehicle travels straight, then vx_v can be calculated from v_s:

Equation 31 $$vx_v = \sqrt{vx_s^2 + vy_s^2}$$

And φ can be calculated according to Section 1.34.2.1. "Driving straight" means that $v_s >> \Delta x\omega$ and $v_s >> \Delta y\omega$. Strictly speaking it is never possible to know if these conditions are fulfilled since Δx and Δy are not known and one can not measure if ω is exactly zero.

Nevertheless, one can always make some reasonable assumption, i.e. Δx and Δy are smaller than the size of the vehicle.

Once φ is known, Δx can be calculated from a simplified version of Equation 28:

Equation 32 $$\Delta x = \frac{1}{\omega}(vy_s \cos(\varphi) + vx_s \sin(\varphi))$$

It is therefore possible to determine φ and Δx without any vehicle state information.

1.34.4 Extracting the best value from a distribution

There are various methods to determine the best value and error of that value from a set of measurements. We will discuss here the mean, the median, standard deviation, and fitting a curve to the distribution of measurements.

.

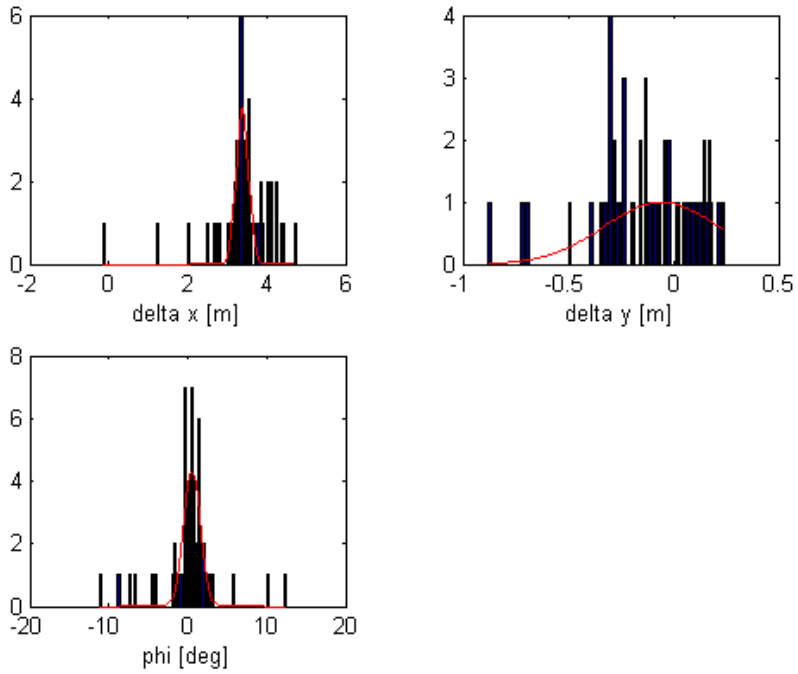


Figure 92 Distributions of Δx, Δy, and φ. Gaussian curves are fitted to each and are shown in red.

1.34.4.1 Mean value

The mean or average value gives a correct answer if the distribution of measurements is symmetric. Problems arise if there are outliers, i.e. few measurements which are far from

the center of the distribution. These outliers can distort the mean. Another situation that the mean value does not handle well is if the distribution is split, i.e. if we have more than one peak. This situation arises in our method when the angle φ we want to measure turns out to be around 180^o and we look at the distribution between $-180^o < \varphi < 180^o$. Since $+180^o$ and -180^o are equivalent, we will get a peak around each of the two.

1.34.4.2 Median value

The median value often does not give an answer as accurate as the mean value, but it is much less sensitive to outliers and the split peak problem

1.34.4.3 Standard deviation

The standard deviation gives a measure of the width of a distribution or the error of a measurement. It is important to note, that it gives the error for the individual measurements and not the error on the mean or median. The error on the mean or median is smaller than the error on the individual measurements.

1.34.4.4 Fitting a curve

If the underlying shape of the distribution is known, one can fit the appropriate curve to the distribution of measurements and thereby extract the best estimate of the value, width etc. There are two main problems, one is that the underlying shape is often not known and the other is that one can end up in a local minimum when doing the fit and thereby getting a false result.

We found that fitting a Gaussian curve to our distributions gives us good results. Fits to the distributions of Δx, Δy, and φ are shown in Figure 92.

1.34.4.5 Error of the estimated value

We have estimated the desired value by forming the mean, median, or fitting the distribution and we have the width or standard deviation of the distribution. In the ideal case when the error on each individual measurement is purely statistical and Gaussian, as opposed to e.g. a systematic offset, the mean and the fitting would give the same result and the error on each individual measurement is the standard deviation (same as the σ in

a Gaussian fit). Then, the mean is a combination of *N* individual measurements with error σ, and the error on the mean is:

Equation 33
$$\sigma_{mean} = \frac{\sigma}{\sqrt{N}}$$

1.35. Accuracy of Velocities Measured by DATMO

The raw data supplied by a laser scanner are distances from the sensor to objects. By observing the changes of distances over time, it is possible to determine the velocity of objects. The basic steps to measure velocities are:

1. Segment the raw data into objects
2. Track the objects over time
3. The velocity is the displacement of the object divided by the appropriate time

In the following sections we will discuss several different methods on how this can be done. The methods differ mainly on the third point, namely how the displacement is being measured. These four will be mentioned:

A. Center of mass tracking: The displacement is the difference in the location of the center of mass.
B. Closest point tracking: The displacement is the difference in the location of the closest point to the vehicle.
C. Point-to-point matching: The displacement is the best match between the points from one scan to the other.
D. Line-to-line matching: The displacement is the best match between the line(s) fitted to the points from one scan to the other.

1.35.1 General test procedure

To measure the accuracy of a velocity measurement one needs know the velocity of the object by an independent, preferably more accurate, method. In our case, one would have the sensor observe an object while at the same time we record the movement of the object. This has some technical difficulties, especially the synchronization of the sensor and the data taking of the object. Also, it will be quite time consuming if one wants to do this with several different object.

We chose a different method. The sensor was mounted on a vehicle and we observed stationary objects while at the same time we recorded the movement of the vehicle. Since the sensor was moving, the objects had an apparent velocity. This velocity was measured by the sensor and compared with the apparent velocity of the object calculated from the movement of the vehicle. In other words, a moving sensor observing stationary objects is functionally equivalent of a stationary sensor observing a moving object.

1.35.1.1 Velocity accuracy from location accuracy and update rate

The laser scanner has a distance accuracy of +-0.3 cm (standard deviation) and an update rate of 75 Hz (1^o resolution). If the change in location between two scans divided by the time between two scans is used as the velocity, then the accuracy in velocity is +-20 cm/s. If instead one uses scans separated by 1 second, then the accuracy is +-0.3 cm/s, but now the update rate is 1 Hz.

This error does not include the tracking error. Objects are extended and the scans often measure different parts of the object while tracking it. In the worst cases this introduces errors in the location of the object equal to the size of the object and accordingly an error in the velocity equal to the size of the object divided by the appropriate time (e.g. the time it takes to drive past an object).

In the ideal case one tracks one fixed point of an object (e.g. its center or one feature) and facilitates an appropriate filter. A typical filter would include a motion model of the host vehicle and the observed object.

1.35.1.2 Center-of-mass tracking of compact objects

In the following discussion we investigate the accuracy of velocity determination by using a center-of-mass tracker and compact objects. The sensor was mounted on Navlab 11. We choose a tree as the object to track (Figure 93).

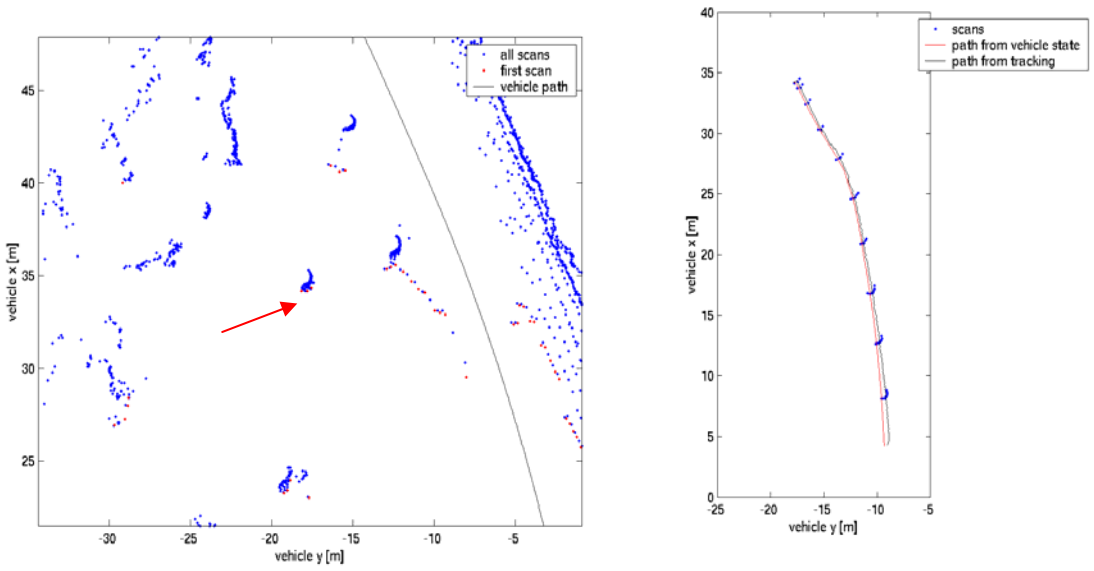


Figure 93 The left side shows the scans projected into a global reference frame. The chosen tree is in the center. The right graph shows the scans in the (moving) vehicle frame. The "path" of the tree determined by the vehicle state and by tracking are shown.

There are four basic steps to the center-of-mass tracking algorithm:

1. Location X_0 of the tree is given (user supplied for the first iteration).
2. All points $p_i=(x_i,y_i)$ of the next scan within +- 3m of X_0 are collected.
3. The new location $X_0=(x,y)$ of the tree is the center of mass of these points, i.e. the average of x_i and y_i.
4. 4. back to 1.

Figure 93 shows in the right graph the scans and the "path" of the tree determined by the vehicle state and by the tracking. The velocity at time t was determined as:

$$v = (X_0(t) - X_0(t\text{-}1s)) / 1s$$

I.e. the average velocity of the last 1s with an update rate of 35 Hz.

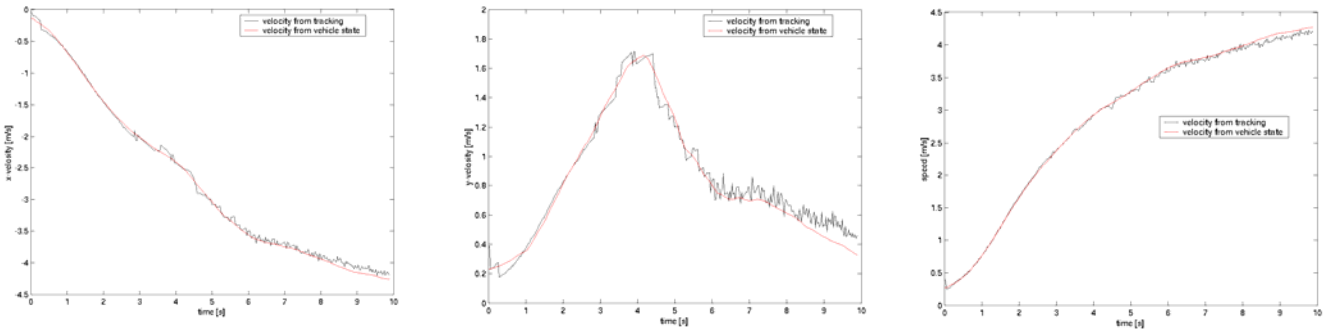


Figure 94 Velocity in x and y direction and the speed determined by tracking and by vehicle state.

Figure 94 compares the velocities and the speed determined by tracking and vehicle state. The speed is the quadratic sum of the two velocities. The standard deviations of the difference between the tracking and vehicle state estimates are 0.051 m/s, 0.062 m/s, and 0.038 m/s for x-velocity, y-velocity, and speed respectively. The error for the speed is considerably less than the ones for the velocities, indicating that the x – and y-velocities are correlated.

These measurements were repeated for a situation where the vehicle makes a sharp turn and another situation where the vehicle speed was almost 13 m/s. For the sharp turn situation the errors were 0.082 m/s, 0.075 m/s, and 0.109 m/s and for the higher speed situation 0.071 m/s, 0.167 m/s, and 0.065 m/s. These errors are larger than the previous ones and some of it can be attributed to a timing issue we had with the yaw measurement of the vehicle and a misalignment of the laser scanner.

Nevertheless, following numbers describe a conservative estimate of the error in the velocity of a relatively compact object:

Error in x velocity: 0.08 m/s

Error in y velocity: 0.17 m/s

Error in speed: 0.11 m/s

1.35.1.3 Tracking and matching algorithms for extended objects

The data presented in this section was taken with the side collision warning system mounted on the transit bus of the Port Authority of Allegheny County.

Each single scan from the laser scanner is segmented into objects. An object is the sum of points that are less than a threshold value apart from each other. An example can be seen in Figure 95:

Figure 95 On the left side is a single laser scan segmented into different objects. The images on the right are from two video cameras and show the corresponding objects. Notice that the segmentation is not perfect, there are single points which are designated as separate objects even though they originate form the same car as a large object. The red line inside the large objects indicate their velocities.

Objects are tracked between scans and, in the example shown in Figure 95, the velocity of the objects is determined by point-to-point matching. (i.e. for each point in the current scan, one finds the closest point in the previous scan while not exceeding a certain threshold). For this closest point one finds again the closest point in the scan before, etc. Finally, for each point in the current scan one has a chain of points reaching in the past and one can determine a velocity for each point. The velocity of the object is then the average of the point velocities.

Figure 96 shows the calculated velocity of the stationary car that is depicted in Figure 95. Because the car is stationary, the velocity should be zero for all times. Any deviation from zero is a direct measure of the error.

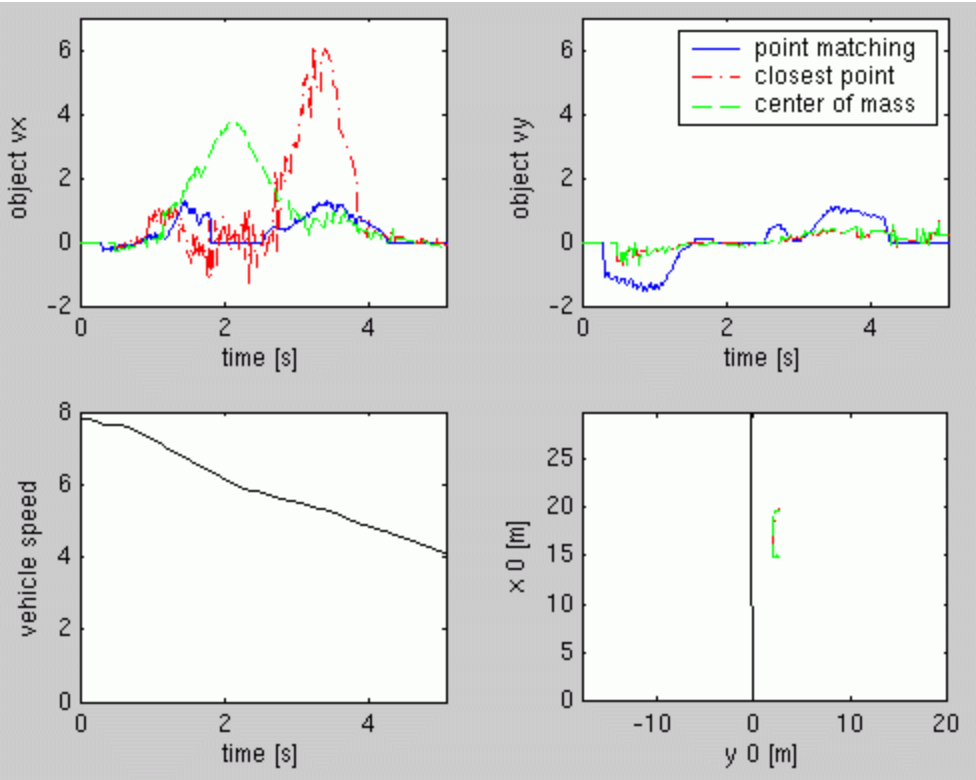


Figure 96 Velocity measurement of a stationary car passed by a bus. The top two graphs show the measured velocities parallel (vx) and perpendicular (vy) to the bus for three different methods. The lower left graph is the speed of the bus and the lower right graph is the path of the bus together with the locations of the closest points and center of masses.

The velocity determined by the point-to-point matching method is compared with two other methods: tracking the point of the object closest to the bus and tracking the center-of-mass of the point cloud. The quality of the point-to-point matching method is approximately the same for velocities parallel (x-direction) or perpendicular (y-direction) to the bus. The closest-point or center-of-mass methods are both much worse for the x-velocity, but much better for the y-velocities. The standard deviations are shown in following table:

	Point matching	Closest point	Center-of-mass

σ(vx) [m/s]	0.43	1.70	1.17
σ(vy) [m/s]	0.69	0.22	0.23

Table 30. Standard deviations of three matching methods for a stationary car

We wanted to investigate, if these numbers change under different circumstances. In the above example, the car is parked parallel to the bus, is not occluded, and the bus is driving straight. In the next example, the bus is turning left, the car is occluded for some times and the car is oriented at different angles relative to the bus.

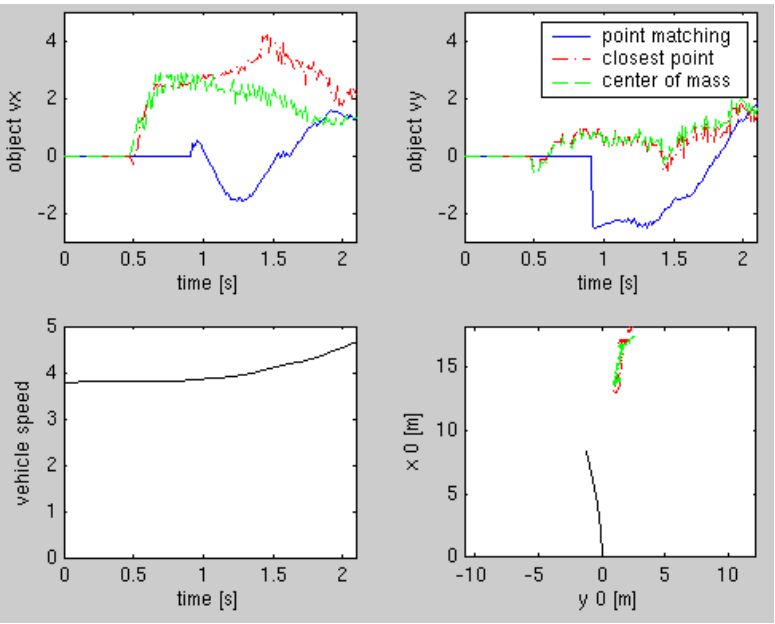


Figure 97 The same as Figure 96 but now for a situation where the bus is turning left.

The respective standard deviations are listed in following table:

	Point matching	Closest point	Center-of-mass
σ(vx) [m/s]	0.95	1.32	0.98
σ(vy) [m/s]	1.04	0.51	0.58

Table 31. Standard deviations of three different matching methods for bus turning left

Most of the values are worse (and sometimes more than twice as bad) than in the previous example. Only the determination of vx with the closest-point or center-of-mass method is better.

1.35.1.4 Conclusion

The source of the error for the closest-point and the center-of-mass methods are obvious, namely, the laser scanner sees different parts of the object and therefore the object seems to be moving. The results from the point-to-point matching, especially why it is so much worse for vy than the other two methods, are more complicated. The point-to-point method would work, either if the points would be randomly distributed on the surface of the object or if they would always be at the same location on the surface of the object. But in our case the points move systematically on the surface of the object and so we have this movement in addition to the movement of the object. An example can be seen in Figure 98.

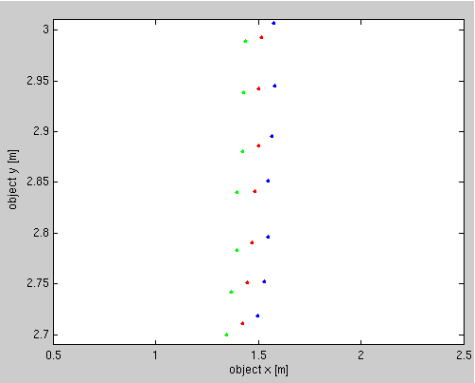


Figure 98 Three consecutive scans, blue, red, and green. The movement to the left is caused by the moving object itself. The small movement down is caused by the points moving on the surface of the object.

Neither of these three methods is good enough for our purposes, we therefore developed another method, the line-to-line matching method. This new method is described in detail in the section on the DATMO algorithm. In the next section, we discuss the accuracy of the resulting measurements.

1.35.2 Quantitative results of line-to-line matching

As before, we looked at the residual velocity of fixed objects to determine the accuracy of the velocity measurements.

We chose two situations. The first one is the same as the one we analyzed in the previous section, in which the bus drives straight while passing a parked car. The second one we

chose because it gave larger measurement errors than other situations we observed. This should show us what we could expect in a worse than typical situation. We suspect that this second situation gives worse results because the shape of the vehicle is more rounded and therefore lines do not fit as well as they would to a rectangular shaped vehicle.

Figure 99 shows the first situation. The following table lists the errors of the line matching algorithm compared to the point-to-point matching, closest point, and the center of mass tracking methods (everything in m/s):

	line match	point match	closest point	center of mass
σ(vx):	0.29	0.43	1.77	1.18
σ(vy):	0.09	0.69	0.22	0.24
max(vx):	0.81	1.29	6.08	3.78
max(vy):	0.39	1.53	0.73	0.73

Table 32. Line matching algorithm errors vs other methods

The velocity estimation is significantly better with the line matching algorithm than with any other method, no matter which criteria is used (standard deviation or maximum deviation in x or y direction). The error in the y-direction is expected to be less, because the object is less extended in the y-direction. The standard deviation of the velocity from the line matching algorithm is less than 0.3 m/s and the maximum (absolute) deviation is less than 1 m/s.

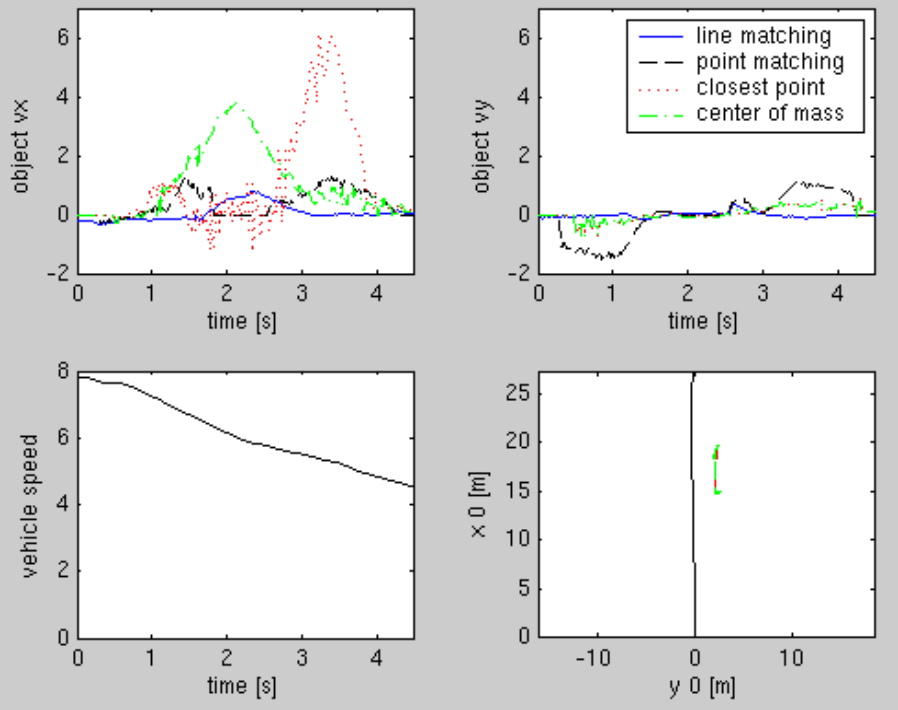


Figure 99 Velocity measurement of a stationary car passed by a bus. The top two graphs show the measured velocities parallel (vx) and perpendicular (vy) to the bus for four different methods. The lower left graph is the speed of the bus and the lower right graph is the path of the bus together with the locations of the closest points and center of masses

Figure 99 shows the comparison between line-to-line matching, closest point, and center of mass tracking methods for the second situation:

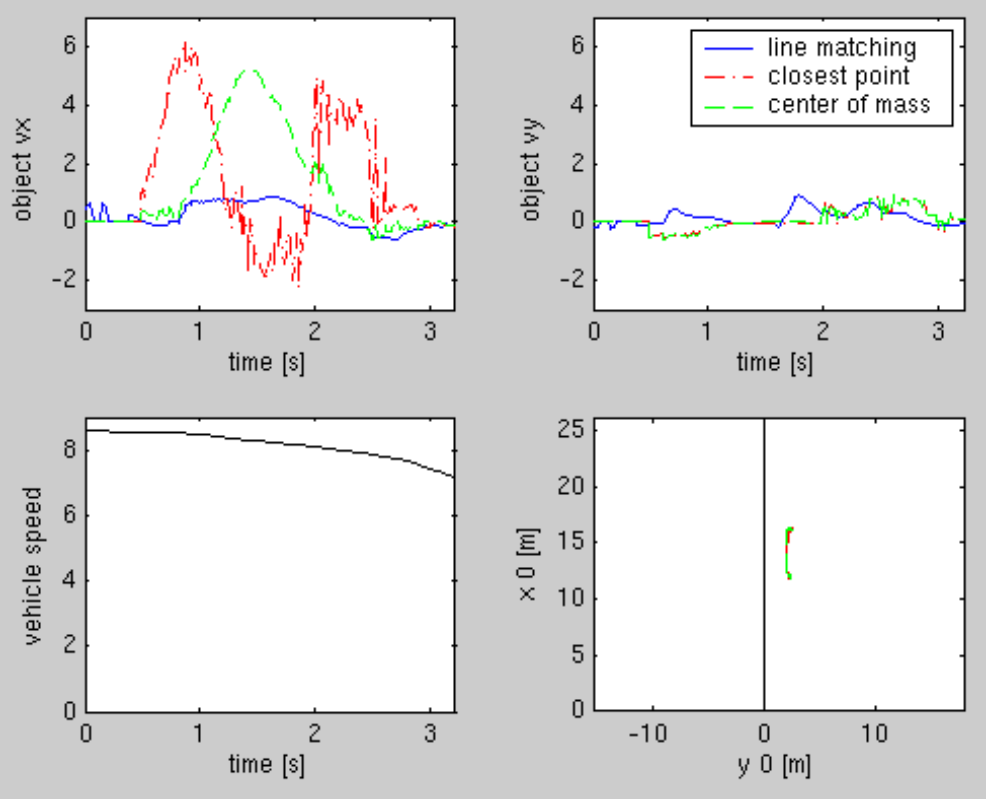


Figure 100 Same as Figure 99 but for a situation which gives worse error.

This table summarizes the errors in Figure 100 for the different methods:

	line match	closest point	center of mass
σ(vx)	0.44	2.18	1.77
σ(vy)	0.26	0.35	0.34
max(vx):	0.85	6.18	5.25
max(vy):	0.93	0.95	0.93

Table 33. Errors from the three different methods

The errors from the closest point and center of mass tracking methods are comparable with the previous situation, but the errors from the line matching in the y direction is twice as bad.

We analyzed a few more situations and always found similar results as reported above. In one of those situations the maximum speed of the bus was a little bit over 10 m/s and the accuracy was +-0.15 m/s.

1.35.2.1 Discussions about line-to-line matching

The line-to-line matching algorithm is clearly better than any of the three other methods. Its accuracy decreases if the observed object is not well described by straight lines. Even with this decreased accuracy, it is still better than the other methods. Another situation which is difficult to analyze is when the observed object is oriented in such a way, that the scanner can only see one line, i.e. when the surface of the object is perpendicular to the beam of the scanner. But in that situation any algorithm will have problem because of the lack of features (i.e. a corner).

1.35.2.2 Error characterization of the full DATMO

In order to get a better characterization of the error function of the full DATMO we looked at a 40 second long data set. During this time the bus was driving at about 10 m/s past a whole series of fixed objects: parked cars, mail boxes, and lamp posts. DATMO detected 312 different objects. The DATMO Algorithm section describes how the velocities of all the different objects are determined. The distribution of the measured velocities shows the error function.

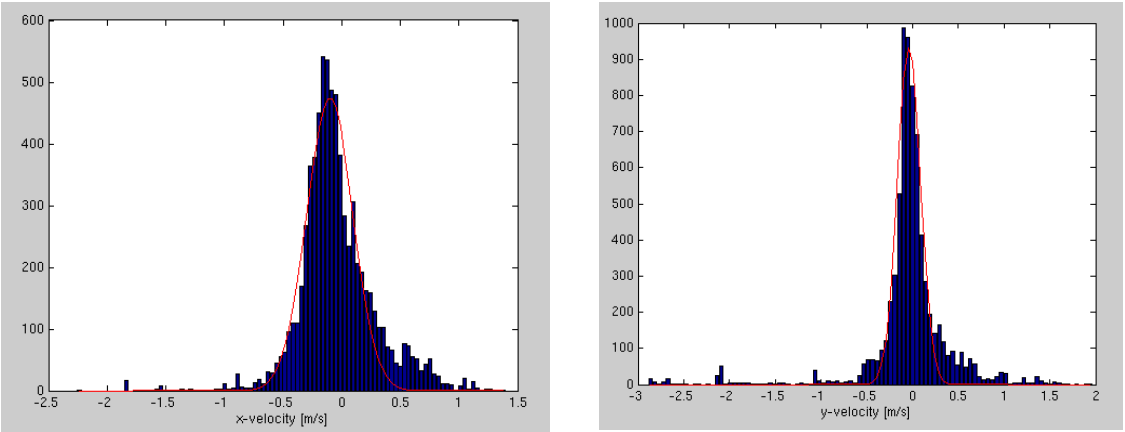


Figure 101 Distribution of the error in velocity. The left shows it for the velocity in x direction and the right for the y-direction. The red lines are Gaussian fits to the distributions.

Figure 101 shows the distributions for the x and y directions. Gaussian curves were fit to the distributions (shown in red) and gave following parameters:

x-velocity	center: -0.10 m/s	σ: 0.20 m/s
y-velocity	center: -0.04 m/s	σ: 0.13 m/s

The centers of both distributions are not exactly at zero. The offset for the x-direction can be explained by a 1% inaccuracy of the speed of the bus. The offset for the y-direction could be due to a small misalignment of 0.2^{o} of the laser scanner. Both of these errors are very small and well within the known accuracy of the bus speed and the sensor alignment.

The distributions are fairly well described by the Gaussian curve, except for their tails which are much stronger. These outliers can come from inconsistent scanner data, e.g. if the scanner sees different parts of an object or does not get any return from certain parts of a vehicle. We later discovered that the bus itself was not level and therefore the sensor plane was not parallel to the ground. This would explain why we didn't always get consistent returns, *i.e.* the scanner probed the objects at different heights depending on the distance of the objects.

Another source of errors is ground returns. Sometimes the laser scanner sees the ground at a very shallow angle. Since the angle is so shallow, any movement of the scanner results in a strong change in what the sensor sees, and therefore DATMO sees a fast moving and/or fast accelerating object.

1.35.2.3 Conclusion

In one of the sample situations and for the extended data set, the relative velocity of the car was more than 10 m/s. The car was moving towards and away from the scanner, and it was moving almost parallel to the scanner beams and perpendicular to them. All this is equivalent of saying:

Relative velocity:	vx = -10 ... +10 m/s
	vy = -10 ... +10 m/s

Depending on the situation, the accuracy is between +/- 0.15 and +/- 0.45. The accuracy is mainly dependent on how well straight lines can be fitted to the object. In general, the accuracy is described by a Gaussian distribution with $\sigma = 0.2$ m/s plus occasional outliers of a few m/s.

The algorithm runs about 5 times faster than real time on a standard PC.

1.36. Quantitative Evaluation and Testing of FCWS

In order to validate the Transit Bus FCWS performance extensive tests were undertaken in a known environment similar to urban and suburban driving conditions. The testing was conducted for two main purposes: (a) to preliminarily evaluate the performance of the FCWS algorithm including sensor detection, estimation and fusion for multiple target tracking and threat assessment based on those tracking algorithms developed at California PATH; (b) to test the measurement and estimation error characteristics based on vehicle on-board sensors in an known environment. The test data can be used for system tuning and further development purposes. A test of a FCWS was also conducted by CAMP project as reported in [1]. However the testing conducted by CAMP was Human Factors related to test a specific maneuver (driver last minute braking) and was conducted using drivers from different age groups as is reported in [2] The CAMP project was mainly for purposes of defining warning threshold criteria as opposed to a test of the technical characteristics of the system.

In any urban and suburban driving environment, objects or hazards in bus forward path can be divided into two categories: moving objects and static objects. The test environment described in this report was created purposely and thus known in the following senses: Moving object – its velocity and position with respect to an inertial coordinate system are synchronized and recorded in real-time together with those of the bus – the subject vehicle; Static object – its position is also recorded. If the subject vehicle moves in a specified manner from a known initial position, then its motion history is known at any time. In this way, a known inter-relationship between the subject vehicle and the environment (moving target vehicles and static object) is created. Those true values are thus used to compare with the corresponding detected/estimated values based on remote sensors.

The test was restricted to vehicle moving along a straight road instead of on curved road. However, similar tests can be conducted for any other environment in future development, for example on a curved road, or up/down hill sections. Our test site was at Crows Landing, an abandoned NASA airfield, which provides multiple straight lanes (runways) without extra disturbances.

This section will describe the test procedures and data collection methods as well as provide some preliminary data analysis from the testing.

1.36.1 Test Objectives

(1) To test sensor ***measurement error*** and ***time delay***, mainly from LIDAR and RADAR for target position and/or speed detection in a known environment;

(2) To test the ***estimation/prediction error*** and ***processing time delay*** in the algorithm. The algorithm takes sensor measurements as input and target position, speed, and acceleration as output. Those two factors are the most critical factors for threat assessment of warning issuance;

(3) Other on- vehicle sensor ***measurement errors*** and ***time delays*** including speedometer, yaw and yaw rate from the Gyro; (The relationship between steering angle and yaw rate is already known.) It is noted that, although the test is on a straight road, minor yaw movement would greatly affect the on-board sensor detection accuracy.

(4) ***Reliability and robustness***: Target missing rate in raw measurement and in real-time processing such as tracking. In general, there are two places in the system which could lead to a target being missed: the sensors themselves do not detect the target at all (This happens to both LIDAR and RADAR) and the algorithm fails to recognize it correctly from the sensor outputs. The target might be missed or its position might be miscalculated/estimated due to tracking, filtering and/or fusion algorithms problems.

The advantage of using a known environment is that it can provide a known reference which cannot be achieved based solely on current recorded data from the vehicle because we do not know if those data provide true measurement and if not, what are the characteristics of the errors. It can be seen that those tests are not just for evaluation, they also provide a quantitative test of sensor characteristics. The measurement error obtained can be used for future ICWS algorithm development/improvement.

1.36.2 Considerations for Designing the Tests

(a) The bus starts at the same point for each run. Thus the longitudinal position of the bus at any time is known if the speed was calibrated with the help of fifth wheel (true ground speed) and string pot (to be described later; See appendix for photos). To calibrate this, a car with string pot connection is run in front of the bus as a moving target. The ground run-distance can be calibrated using the fifth wheel of the car. The fifth wheel has sensors to count its number of teeth in unit time (converted to speed) and thus to estimate the covered ground true distance under the assumption that there is no tire slip. This assumption is reasonable partly because the road is dry asphalt and partly because the fifth wheel is passively dragged.

(b) The car in front of the bus has a string connection also to ensure that it starts at a known fixed point. To avoid damage to the string pot, another 6.38[m] of string is used as an off-set extension.

(c) Obstacles should be put far enough away for the Bus to accelerate to required distance. In our test, the objects were over 400[m] away.

(d) For each run, the relative positions – both lateral and longitudinal – of the obstacles with respect to the bus lane are known and recorded.

1.36.3 Hardware and Software Setup

These include

- A Lincoln Town car was used as the target vehicle. The Lincoln Town car was installed with an engineering computer running real-time QNX-4; A SamTrans Bus was the subject vehicle, which was installed with a engineering computer PC-104 running QNX-4.

Figure 102 Fifth wheel to measure true ground speed and string pot (Top of the bar)

- A Fifth wheel (Figure 102) was mounted on the Lincoln to measure vehicle ground distance free of any tire slip
- An AMETEK Rayelco Position Transducer (range between 0 – 50ft), or ***String Pot*** (Figure 103), was used for measuring inter-vehicle distance. A String pot was installed on the rear end of the Lincoln, (then hooked to the Bus) including software. The data recorded from the string pot was converted to relative distance between the bus and the Lincoln. Speed and distance measurements on the Lincoln were calibrated before the test
- On the Lincoln a Gyro was used to monitor the lateral movement of the target;
- Data recording: For synchronization, the information passed over from the Lincoln were saved with the other data in the main computer of the subject vehicle.
- Carton boxes covered with RADAR/LIDAR reflecting materials to enhance signal reception were used as static objects.
- Wireless communication system: A FreeWave card was installed on both the Samtrans Bus and the Lincoln running under QNX-4 real-time operating

system; The information passed from the Lincoln to the bus/ or from the bus to the Lincoln was:

- time stamp
- fifth wheel speed
- vehicle acceleration
- yaw rate from gyro scope
- string pot voltage (can be converted as inter-vehicle distance)
- Latitude (GPS)
- Longitude (GPS)
- UTC time (GPS)
- Altitude (GPS)

- 4 voice radios were used for coordinating operation between drivers, people responsible target disposition, recording, ground position measurement

Figure 103 Using string port to detect true inter-vehicle distance on-the-fly

1.36.4 Known Driving Environment

The known driving environment can be designed to include ***static objects*** Figure 104 and Figure 105) and ***moving objects*** (Figure 106). For static objects there is no need to pass

anything. It is only used to test the sensors measurement error and warning. These objects may be designed to include road side parked vehicles, mail boxes, traffic signs etc.. To present different objects, boxes with different size may be chosen. To make the objects RADAR/LIDAR sensitive, the boxes were wrapped with a reflecting cover.

Figure 104 View of the Static Objects from the Bus

A ***Moving object*** may include vehicles driving in different directions, in adjacent lanes and front vehicle. The target vehicle and the subject vehicles are connected with a

Figure 105 Detecting parked vehicles on both sides

Figure 106 Moving (front vehicle) and static objects

measurement string which can measure inter-vehicle distance in real-time. Wireless communication can be used to synchronize the measurements on those two vehicles. This

set up is to test real-time inter-vehicle distance measurement, estimation, prediction and filtering;

Host vehicles always start from a known position. Based on the ground position of the targets and the running distance of the bus at any time instant, we know the relative position between the bus and the targets, which is a critical point. All the measurements are with respect to a ground coordinate system as defined in Figure 107 .

1.36.5 Preliminary Test

A pre-test for the following items was conducted at SamTrans before the formal test:

(a) Re-calibrate the SamTrans bus and ensure that all the on-board sensors and computers working properly;
(b) Verify that the sensors and wireless communication systems were properly installed, calibrated and working on the Lincoln;
(c) To use laptop computer connected with the subject bus to use "run" instead of "auto-run" for manual data-saving interrupt for matching the saved data with the test maneuver;
(d) To make sure all the data saving are correct on the subject bus;

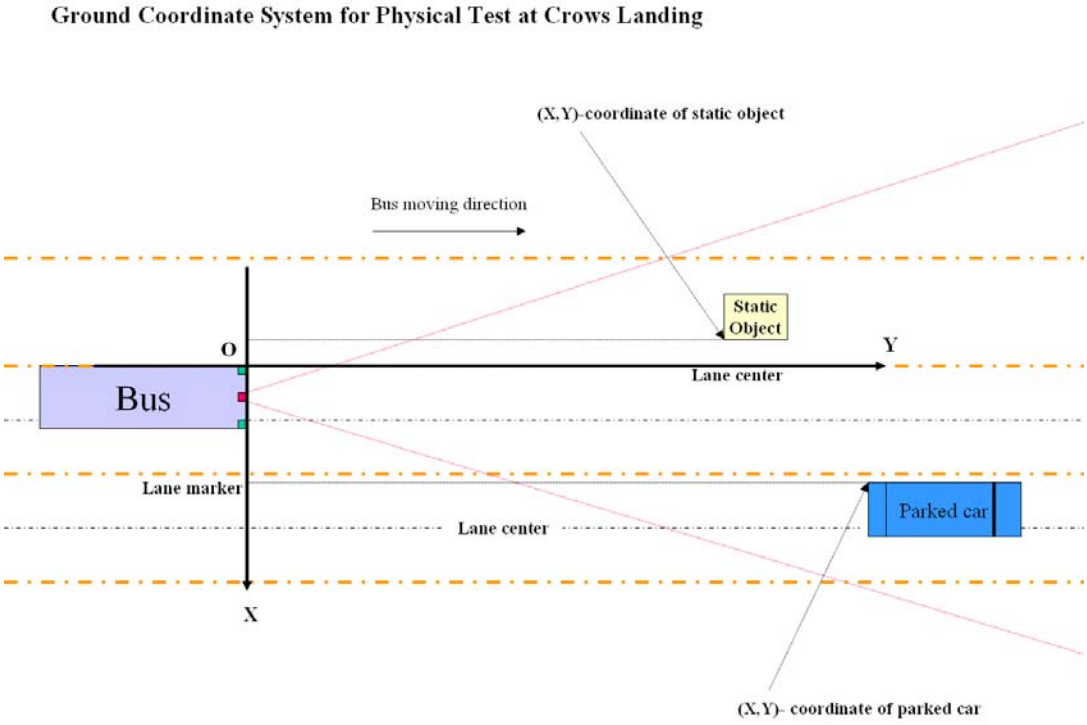


Figure 107 A Ground Coordinate System

1.36.6 Crows Landing Test

1.36.6.1 Relative speed and inter-vehicle distance error and time delay test without string but with wireless communication

This test can be used to figure out the relative speed error and measurement time delay with low relative movement. Without string, such movement can be made much larger and faster;

Maneuver 1: Vehicle following. (Figure 108) Use a leading vehicle in the front of the bus with FCWS to run at different constant speed: 5[mph], 10[mph], 27[mph], 40[mph], 55[mph] for some time. The bus driver was asked to determine a safe and comfortable inter-vehicle distance.

Leader vehicle approximate deceleration: 0.2[m/s^2], 0.8[m/s^2], 1.5[m/s^2]

Inter-vehicle distance: speed/relative-speed dependent on the vehicle speed and driver's choice – feel comfortable;

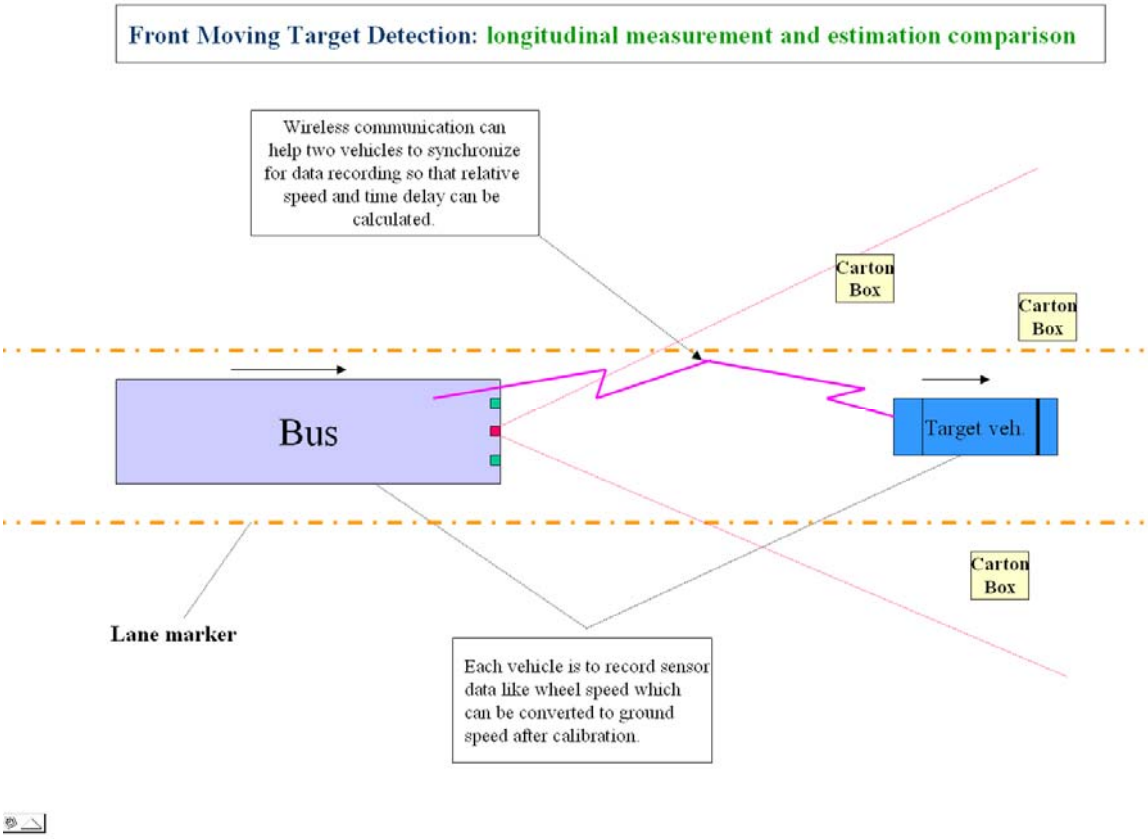


Figure 108 No string for vehicle following

1.36.6.2 Inter-vehicle distance error measurement (with string) and time delay test with variable speed and deceleration

Maneuver 2: Vehicle following (Figure 109): Use a leading vehicle in the front of the bus with FCWS to run at different constant speeds: 5[mph], 10[mph], 15[mph], 20[mph], 25[mph] for some time then the lead vehicle decelerates at approximately: 0.2[m/s^2], 0.5[m/s^2], 0.8[m/s^2]

Inter-vehicle distance: speed/relative-speed dependent. Because the total length of the string is 16 [m], an offset 6.38 [m] of the string is used to avoid break due to over-stretching.

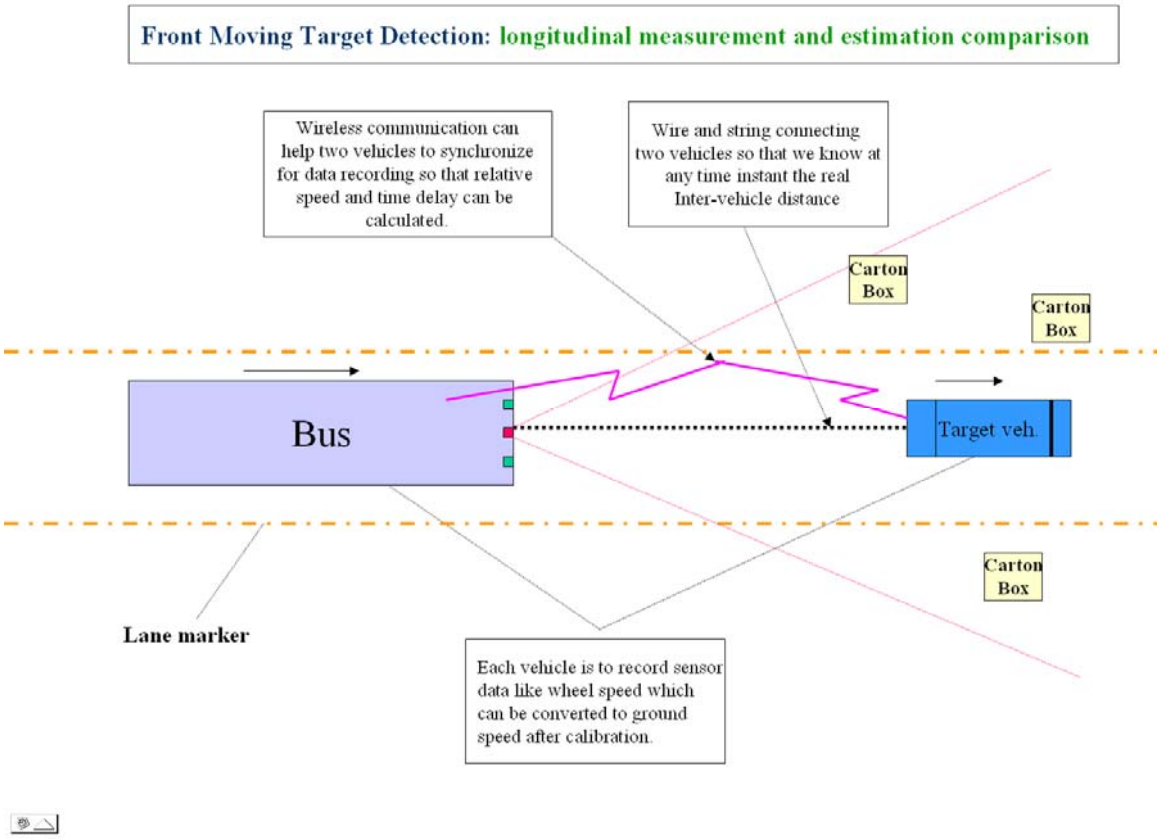


Figure 109 String Pot and wireless communication are used

1.36.6.3 Static object lateral distance measurement, prediction/estimation error test

Maneuver 3: Carton boxes covered with RADAR reflectors at certain heights are put in known places with respect to the center of the road (Figure 104). The Lincoln (Figure 102) is parked on the left or right side at certain distances with respect to the centre of the road: 1.4[m], 2.0[m], 3.0[m] measured to edge of the Lincoln; Drive the Bus straight ahead at different speeds: 5[mph], 15[mph], 27[mph], 35[mph]. The bus needs to run in the center of the lane or at the edge of the lane; The Lincoln driver opened the left door sometimes (Figure 111); Multiple cars and boxes used as objects to make sure there is no overlap. Heavy objects are put inside the boxes so they would stay in place.

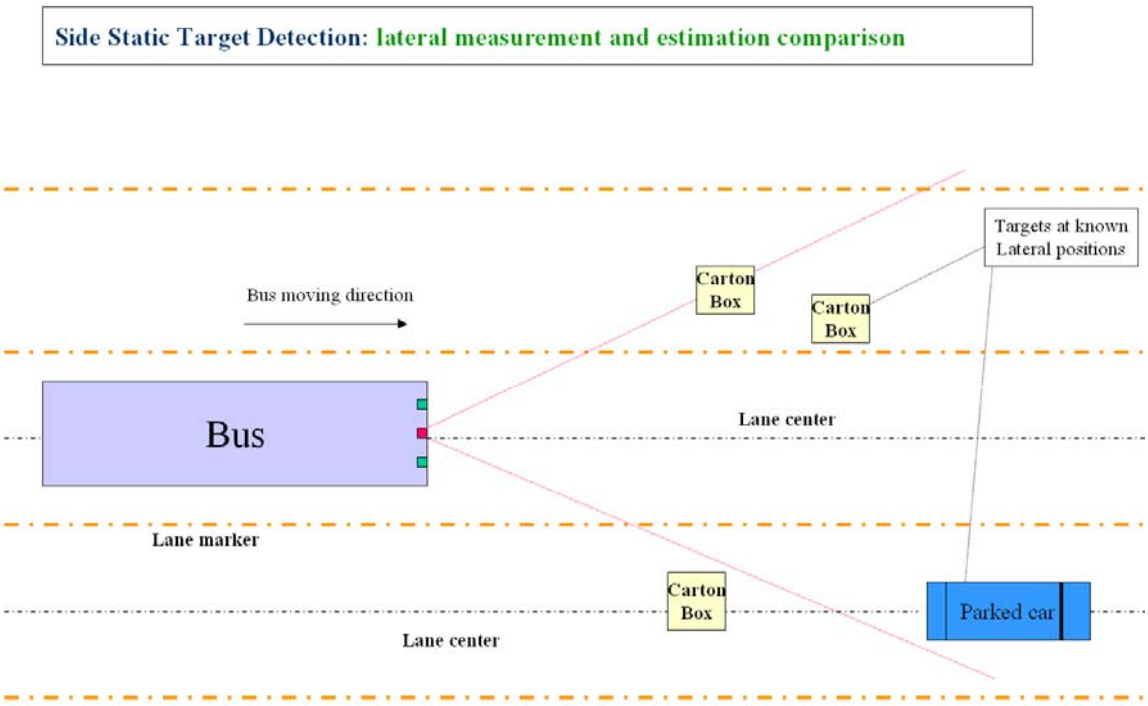


Figure 110 Parked car testing scenario

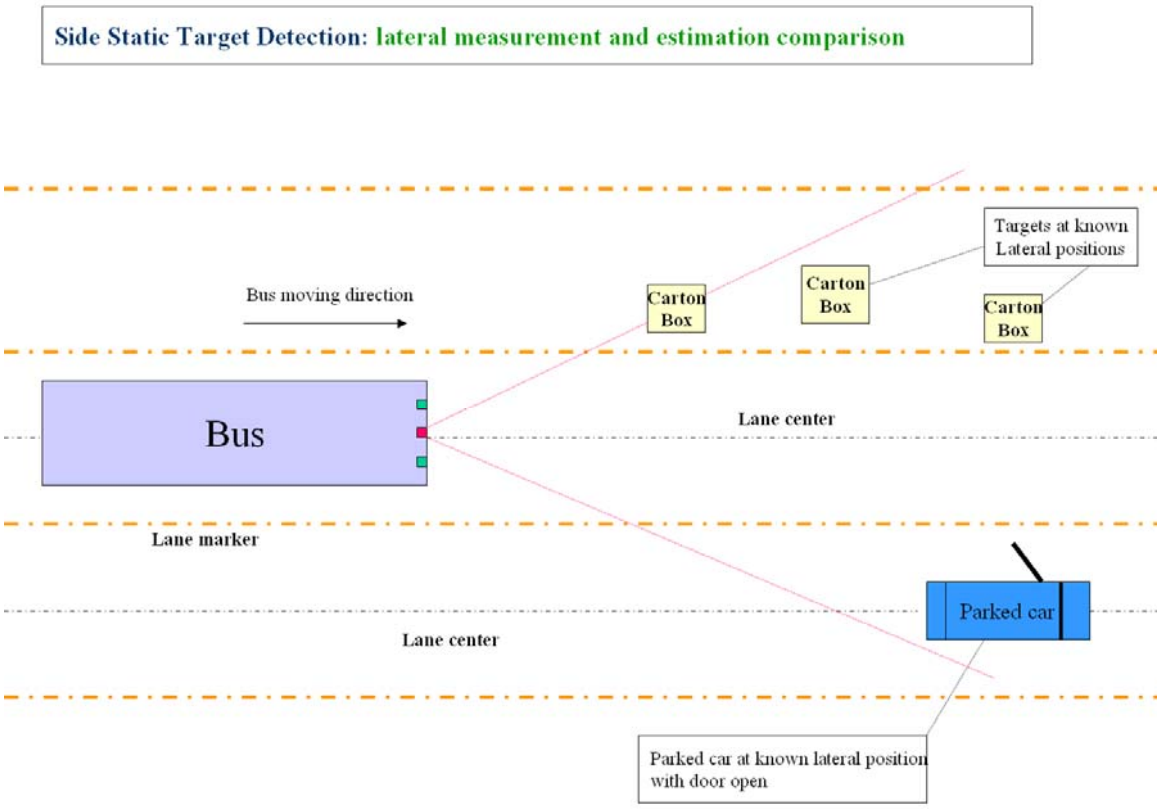


Figure 111 Park car door open test scenario

Maneuver 4: Two cars are running in left/right adjacent lanes but a known lateral distance in the same and opposite direction at different constant speeds: 10[mph], 30[mph]. The bus can run at slightly different speeds (non-constant) so that there is some relative movement when the vehicles run in the same direction (Figure 112)

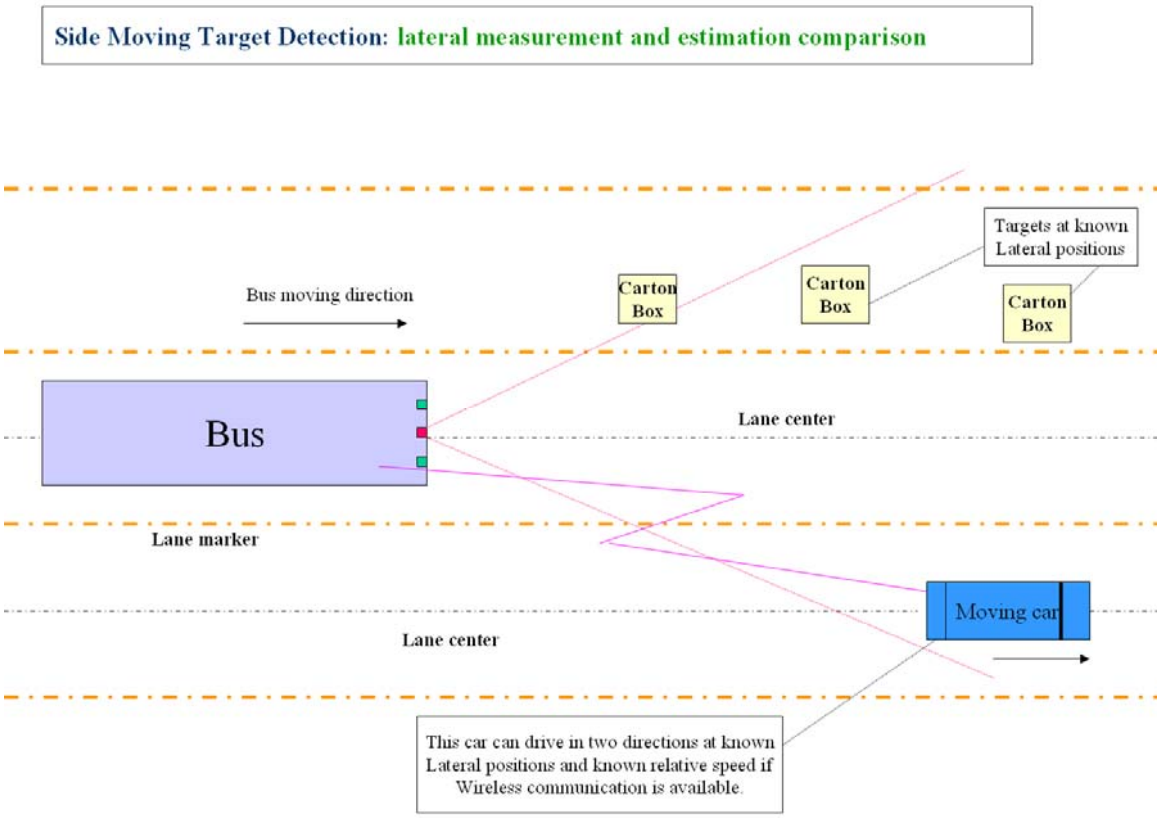


Figure 112 Side Moving Target Direction

1.36.6.4 Cut-in test

Maneuver 5: The Lincoln travels in the left/right adjacent lanes but at a known lateral distance in the same direction at different speeds: (10[mph], 20[mph], 35[mph]) for a while and then accelerates to take over the bus and cut-in (Figure 113). The speed variation of Lincoln is intentionally made. The bus driver is to decide an appropriate inter-vehicle distance.

1.36.6.5 Gyro rate and RADAR/LIDAR dynamic angle measurement test

Maneuver 6: Drive the bus straight at certain speed: 5, 15 [mph]; Once the bus arrives at a certain point, drive around and then return in the same lane in the previous direction and pass the objects again. In each run the objects will be viewed twice by the RADAR sensors.

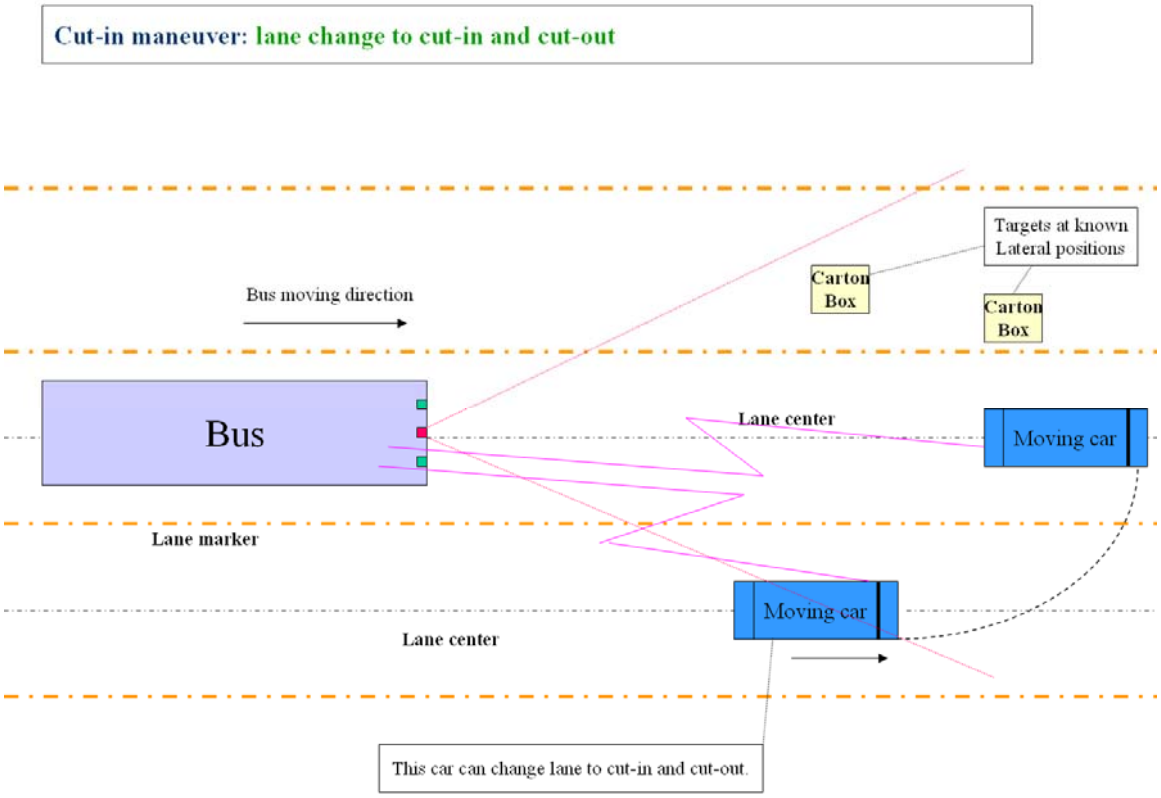


Figure 113 Cut-in and cut-out to test lateral movement detection

1.36.6.6 Low speed approaching/crashing to a static object

Maneuver 7: Carton box (covered with foam block) with RADAR reflectors at certain height are put in different places of the road and drive the Bus towards the object at different speed: 25[mph], 15[mph], 10[mph], 5[mph] to see the reaction of the warning system and driver's response (Fig. 114);

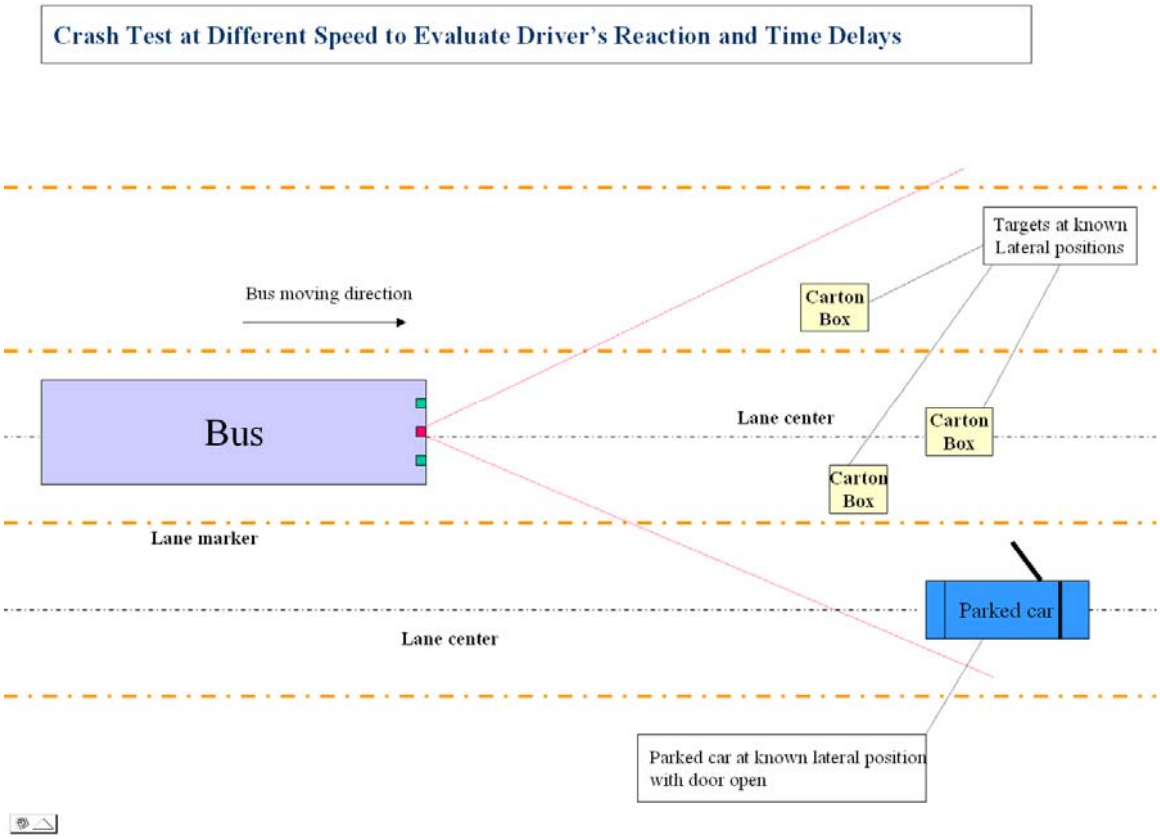


Figure 114 Crash Test; No string is used

1.36.7 Data Analysis

As mentioned previously, the test data can be used for two objectives: (a) To check LIDAR/RADAR measurement, estimation and target tracking; (b) To tune those parameters. The data collected through these experiments have shown that both of these objectives can be achieved. The following presents examples of measurement and estimation using LIDAR and RADAR compared with the independent measurement from the fifth wheel and the string pot measurements.

1. The following plots correspond to *Maneuver 2* (Figure 109) for target longitudinal measurement.

   In Figure 115, Figure 116 & Figure 117, both target vehicle and SV speeds are around 10[mph]. String pot is used to test LIDAR/RADAR longitudinal measurement and estimation including distance and speed. The fifth wheel speed and string length

are considered truth measurements after calibration. However, lateral position measurement is also plotted.

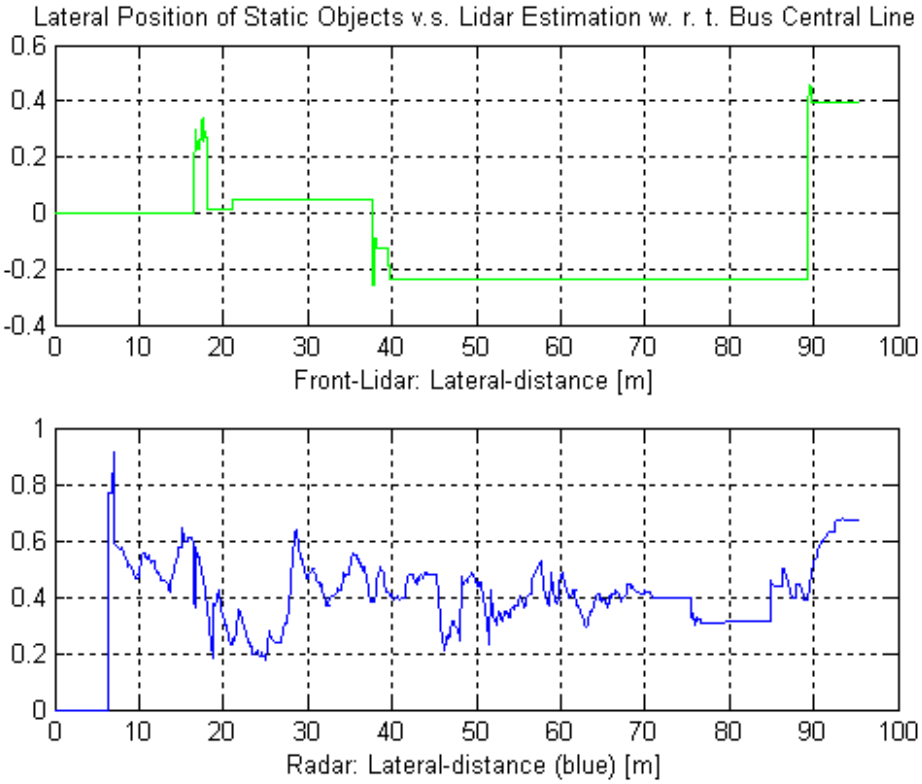


Figure 115 LIDAR/RADAR target lateral position measurement and estimation [m]

It can be seen from this figure that lateral measure of LIDAR is slightly more consistent compared to RADAR.

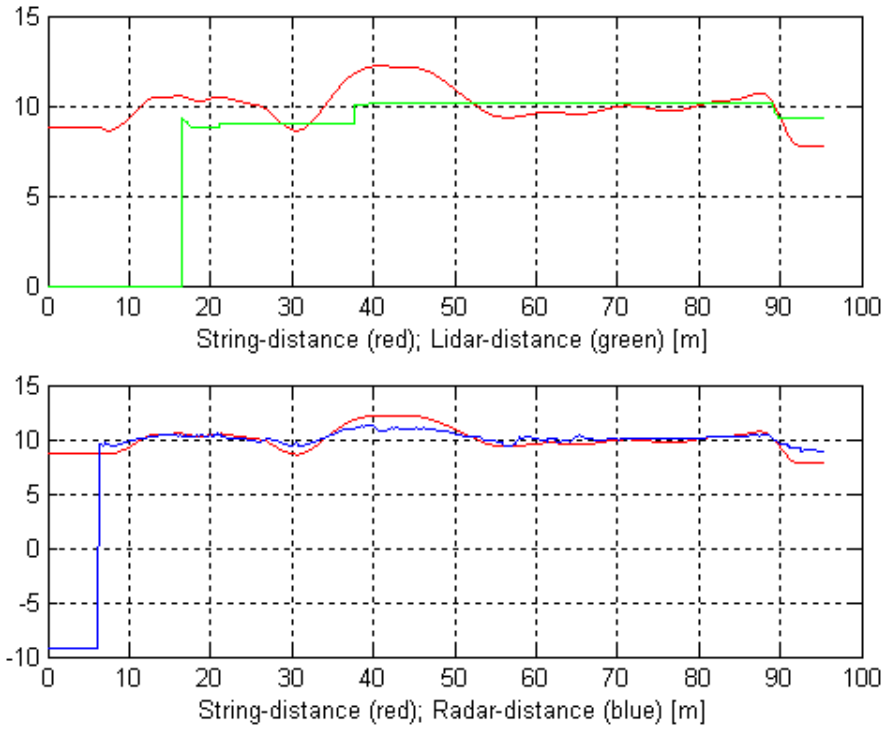


Figure 116 String (true) distance vs. LIDAR/RADAR distance estimation [m]

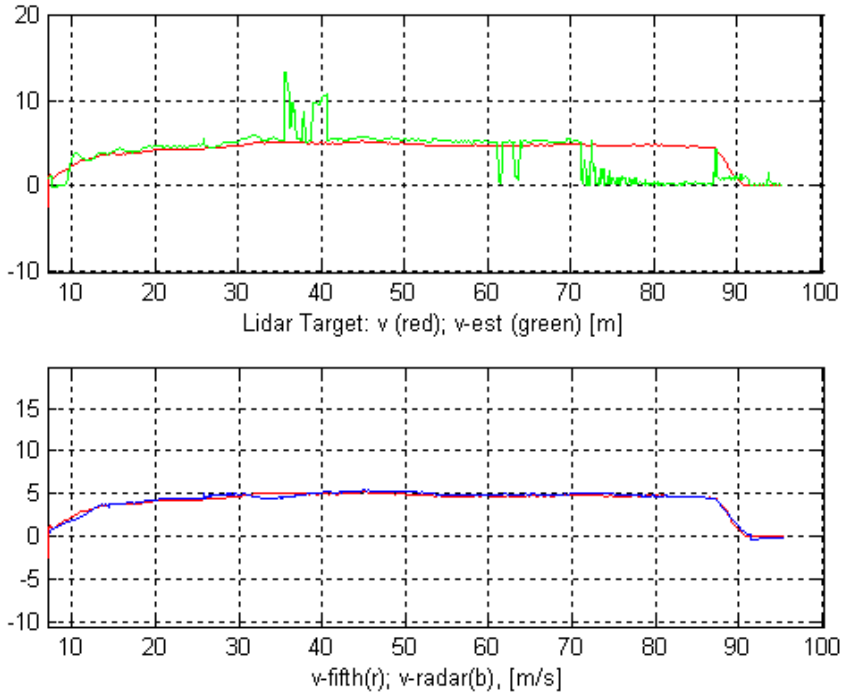


Figure 117 LIDAR/RADAR target speed estimation vs. fifth wheel [m/s]
The above two figure shows that RADAR distance and speed measurement in longitudinal direction could achieve better results.

1.36.8 Future work

PATH plans to utilize the data collected through the verification tests to develop sensor fusion approaches and improve the tracking and warning algorithms in order to achieve better measurement/estimation and system performance.

TRANSIT CWS SIMULATOR

As professional bus operators experience potential collision situations very rarely, it can be difficult to gather enough data to evaluate a systems performance. It is however, possible through the use of a bus simulator to present large numbers of drivers with potential collision situations in a much shorter period of time than drivers would normally encounter in daily driving. Such potential collision scenarios can be recreations of actual accidents or a composite of hazardous factors. Another advantage of a simulator is that it is possible to have a pool of drivers all experience identical situations to see how drivers' behaviors to the same incident differs. Lastly, a simulator allows drivers to be put in potentially hazardous situations without any risk to life.

We are planning to use the FAAC simulator at SamTrans to conduct further study into the collision warning system. It will be possible to research areas such as:

- brake reaction times – such information could be used to refine collision warning sensitivity parameters
- warning sounds – to determine whether drivers react faster to visual or audio cues of hazards and to determine optimum warning sounds
- to investigate the effects of false and nuisance warnings on operators trust in the system
- to determine if drivers' visual scanning patterns change with the addition of the system
- to optimize display techniques

1.37. The SamTrans simulator

The SamTrans Simulator is a FAAC™ simulator and is made up of the following components:

- A simulated Gillig bus driver's workstation, which includes all the normal controls, and seat as the Gillig buses that are in operational use by SamTrans

- Five 70 inch rear-projection visual displays to provide the driver with the “out the window” forward and side view.
- Two 42 inch plasma video displays to provide rear views, these are seen by the driver through use of the mirrors
- An overall 315 degree field-of-view
- An Instructor/operator station that is used to control the overall set-up
- An auxiliary driving station

The set-up of the simulator can be seen in the figures below:

Figure 118 Simulator set-up from the back

Figure 119 Trainer/Experimenter workstation

Figure 120 Driver Seat with forward view

Figure 121 Simulated view of the interior of the bus

Using the existing FAAC simulator of SamTrans buses, researchers at PATH have begun the initial development of a system to integrate the collision warning system into the simulator and to provide a method to collect driver behavior data (such as throttle position, steering wheel angle etc.) that could be analyzed to determine the consequences of implementing different warning systems. A brief outline of the method is presented in the next section.

1.38. PATH CWS/FAAC Simulator Integration

Currently, the PATH CWS operates only on physical buses using actual sensors (LIDAR/RADAR). It is desired to integrate the CWS with the simulator in order to quickly evaluate collision warning performance. The FAAC simulator computer at the SamTrans site broadcasts over a closed Ethernet network the state (i.e. position, velocity, heading, etc.) of the bus and other vehicles depicted in the virtual reality scenario (see Figure 122).

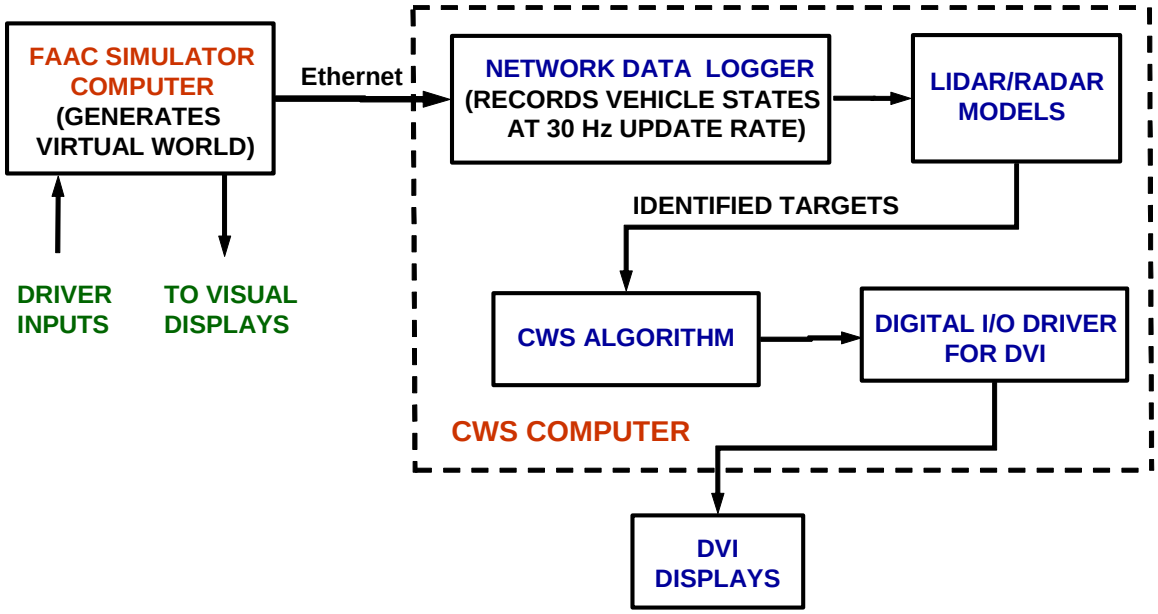


Figure 122 PATH simulator software architecture

Data is transmitted at 30 Hz and is read by a computer system running the CWS algorithm. Since this algorithm requires inputs from actual LIDAR/RADARs, a program has been developed that models the LIDAR/RADAR detections using virtual beams projected into the scene. If a frontal target is detected, this information is sent to the CWS algorithm for determination of the threat level. The LIDAR/RADAR model process provides inputs to the CWS algorithm in the same form as the actual sensors mounted on the buses. This alleviates the need to modify the CWS algorithm used on the actual buses and allows a transparent code interchange between the simulator and buses. If it is determined that a warning should be issued, the CWS process writes the threat level to a digital I/O driver that controls two DVI displays described below.

Inside the booth where the driver views the virtual scene, two visual devices or "light bars" for collision warning will be used, one of which is shown in Figure 123.

Figure 123 DVI light bar.

One light bar is mounted on the left-hand "A" pillar and a second mounted on a center windshield mock-up pillar or strut. Both are wired directly to and controlled by the CWS computer. Each has a number of vertically stacked rectangular light segments on top and a two triangular shaped lights on the bottom. The rectangular light segments correspond to frontal and frontal corner hazards, while the triangular lights refer to side hazards. For this study, only frontal hazards will be considered.

The light bars illuminates amber to indicate a less severe threat while red and ultimately, flashing red indicate a more severe or imminent threat. Based on the CWS algorithm, as a hazardous situation becomes more imminent, more light segments will illuminate, starting at the top and working downward. Thus, collision imminence (i.e. threat of a collision) is reflected in both the number of lights illuminated as well as the color of the lights.

1.39. Summary

Under this project, the SamTrans/FAAC™ simulator is being modified to incorporate CWS functions, which will allow us to create specific scenarios of interest to which large numbers of drivers can be exposed to, providing us with a much more extensive data set than we could obtain from in-service operation of two buses. The project team plans to conduct experiments using the simulator at the later stage of this project. From the simulator experiments, more extensive data sets will be obtained which will be used to analyze driver behavior change due to the introduction of ICWS and for further optimization of the warning algorithms and DVI.

RECOMMENDATIONS

The research and development of the ICWS has made significant progress toward deployment. However, due to the research nature, significant work is still needed in order to achieve a fully commercializable integrated collision warning system. The following outlines further development needed before commercialization can take place.

1.40. Develop ICWS Markets and Industrial Partnerships

Like any product, commercialization of ICWS requires both sizable market demand and willing suppliers. The crash data analysis under the early FCWS and SCWS studies have shown that transit collision warning system can enhance transit safety. A recent cost benefit analysis conducted by Volpe indicated that such safety systems can help the transit operators to reduce operation cost. For a specific transit operator, the extent of the cost saving will depend on level of deployment, which, transit operators say, is very much decided by the unit cost after the technologies meet their performance and technical requirements. The unit cost in turn will depend on the market size. The study conducted by the ICWS team indicated that ICWS can potentially benefit and be of interest of other commercial fleet operators such as UPS, which operate in similar environments. Under the current project, the ICWS team begins to reach out to transit and other fleet operators. The team recommends that this effort be continuously carried out until an initial market is established. In parallel to the market development, it is essential to work with industrial partners to commercialize the ICWS, starting from the phase of field operational tests.

1.41. Conduct Field Operational Tests

Under Phase One of FCWS and SCWS development, three revenue service buses were instrumented with frontal collision warning systems and a test vehicle was instrumented with a side collision warning system. These developments have led to the current ICWS efforts in instrumentation of two integrated collision warning systems onto a SamTrans bus and a PAT bus. Field testing is currently underway. Although the research team has carefully planned the field tests in order to collect data from multiple drivers on selected routes, the exposure to diverse driving behaviors, to different driving environments and to

hazardous conditions is very limited. It is the consensus of the research team and interested transit agencies that a larger scale Field Operational Test (FOT) needs to be performed in order to collect adequate data for verifying the effectiveness of the ICWS and for fine tuning the design parameters or making improvements. The ICWS research team recommends that one or two fleets of 50-100 revenue transit vehicles be equipped with a prototype transit ICWS on a variety of routes and operating conditions for a duration that can justify industry-wide acceptance.

1.42. Human Factor Studies Using Samtrans Driving Simulator

The field tests conducted under the current project provided useful results for evaluation of the effectiveness of the CWS system from Human Factors perspective. Because of small number of buses involved in the field tests within this phase of the project, it is difficult to conduct analysis of driver behavior changes for specific hazard scenarios. We therefore propose to conduct human factors studies using the Samtrans driving simulator to conduct further study of the integrated (forward and side) collision warning system. The following studies have been identified as research priorities.

- To investigate if an integrated (forward + side) collision warning system (CWS) affects distracted and non-distracted Transit Bus Operators response in imminent collision warning situations.
- To investigate if operators' visual scanning patterns change with the addition of the system. It would be useful to know if operators detect all warnings and whether the system causes the operator to become distracted. A similar issue was raised by Lee et al (2002) who, for a car collision warning study, determined that future research should investigate what happens if an operator is already braking when they receive a warning – do they continue to brake at for example the same rate?
- To further investigate what types of warnings bus operators view as nuisance warnings. Whilst some of the types of nuisance warnings have been identified in human factors ride-along, much variation has been seen both between and within operators' responses to each encountered scenario. Use of the simulator would enable

different operators to be exposed to the same scenario repeated times which would help to further clarify what aspects about a scenario feed into an operators' consideration of whether the warning is a nuisance warning. This type of study could also be used to determine the effect of false and nuisance warnings on operators' trust in the system.

- To determine optimal display techniques. This could include different visual display methods as well as audio warning sounds – to determine, whether operators react faster to visual or audio cues of hazards and to determine optimal warning sounds. Also of interest is where a visual display could be placed in a bus that does not have a center pillar. One solution, for this type of bus would be to place the display on the right pillar.
- To further determine optimal integration strategies for the integrated collision warning system. In the present system there is no prioritizing of warnings. It would be valuable to know what the human factors implications would be of the following scenarios: giving the forward system priority at all times, giving the side system priority at all time, giving the most critical hazard priority or having no priority (current system).

1.43. Finalize Performance Specifications

Learning from field operational tests and simulator studies, the performance specifications developed under the current ICWS research program should be updated and finalized in order to meet the transit and other fleet operators' needs. The ICWS Performance Specifications should have separate sections for the following:

1. Specifications related to frontal sensors and performance only
2. Specifications related to side sensors and performance only
3. Common specifications for frontal and side sensors and performance

This would allow transit agencies to purchase non integrated systems at a lower price if they have a lot of side collisions or frontal collisions only.

1.44. Hardware and Software integration of ICWS

The philosophy of building the first advanced prototype was to achieve functional integration and, at the same time, minimize the risk of system integration by having separate duplicate systems and data interfaces and to include comprehensive data collection capabilities. The duplicate systems would prevent one system from taking down the other system should a failure occur. It also minimized risk by making sure that each partner had available what they needed to deploy a system. The sensory data, additional engineering data and video streams collected are for thorough data analysis. In order to perform the FOT, a higher level of hardware and software integration needs to occur in order to achieve the level approaching a commercial prototype.

1.44.1 Eliminate Duplication of Hardware

The experience gained with each other's system can now be taken to the next step of integrating the testing prototype by eliminating duplicate hardware and combining algorithms. Duplications that could be eliminated are:

1. Creep Sensor Interface
2. Gyros
3. Separate electronic enclosures
4. Dinex Interface
5. Power supplies and power conditioning
6. Power up and Power down logic
7. GPS (May also be redundant with electronics for bus tracking and annunciation systems)
8. Cell phone interface (could be eliminated completely)
9. Reduce the number of processors (see next section)
10. Eliminate most of the video cameras and one digitizer (see eliminate video section)
11. Combine the data recording functions into one computer chassis (see next section)
12. If transit bus has stability control system, then bus state information may be available without additional gyros or creep sensor
13. Future drive by wire systems may include steering wheel encoders

Eliminating these duplications would increase the overall reliability of the system due to less electronics. It would decrease the overall cost of the system for the same reason.

1.44.2 Combine / Eliminate Processors

The current ICWS contains five CPU's to handle the top level processing tasks. This does not include the processors that are embedded in any of the sensors. The CPU's in the advanced ICWS prototype include:

1. FCWS Engineering computer
2. Left SCWS Engineering computer
3. Right SCWS Engineering computer
4. FCWS Video and Data recording computer
5. SCWS Video and Data recording computer

A minimal commercial prototype could eliminate both of the Video and Data recording computers since they are not necessary to generate warnings to the transit operator and as a minimum combine the left and right SCWS Engineering computers. The current barrier to combining the FCWS and SCWS Engineering computers is that each system runs different warning algorithms and data processing thus increasing the CPU loading above what one processor could currently handle (see future research for more information).

1.44.3 Eliminate Video

The elimination of collecting video information not only minimizes the CPU and digitizing hardware on a commercial system, it would also eliminate seven of the nine cameras installed as part of the advanced ICWS prototype. The two remaining cameras are used for the curb detection at the front of the bus (laser line striper) and curb detection ahead of the bus (fusion of video with other sensors).

As part of the advanced ICWS prototype, the cameras and video / data recording were necessary to allow the continuing development of algorithms and analysis of system data.

One of the questions that would need to be answered is whether the additional data recording could be used as a feature of the system, e.g. to limit transit liability in

collisions and helping to defend transit operators against fraudulent claims and recording vandalism. It could also be used for training purposes. This might be a feature for which some transit companies would pay the additional cost. It should certainly be part of an optional configuration, but may not be part of the base package.

1.44.4 Commercialize Laser Scanners

The most expensive components of the prototype ICWS system are the LIDARs (laser scanners). In the ICWS prototype the sensors alone account for over $ 15,000. That does not include the additional cost to mount them in retractable assemblies. To make the ICWS more economically feasible, LIDAR sensors should be designed for this specific application. Also, weaknesses of the current scanners significantly increase the system false alarm rate.

The main issues associated with this design are:

1. Overlapping fields of view.
2. Size
3. Reliable detection
4. Resolution
5. Range
6. Update rate
7. Expense
8. Synchronization of scanners
9. Eye safety

Overlapping fields of view: The current system uses three LIDARs. There is a LIDAR mounted on the left side of the bus, the front of the bus and the right side of the bus. The side LIDARs have 180 degree FOV's. The front LIDAR has a narrow FOV and is used to see far ahead in the lane. The LIDAR could be redesigned to mount on the left and right side of the front bumper with 270 degree FOV's. This could eliminate one LIDAR and provide better coverage than the current system in front of the vehicle. Even if the front look ahead LIDAR could not be combined with the side LIDAR, it could be replaced with a much less expensive adaptive cruise control unit since the object tracking could be

done with the other LIDARs. However, this roughly doubles the worst-case distance to cover the entire side of the bus. To get the same resolution we have now at the back of the bus, we would need twice the angular resolution. It seems plausible that coverage of the back half of the bus is not as important as in the front, but this would have to be looked at in more depth.

Size: The LIDARs used on the side of the transit bus are over six inches deep. Most transit buses are at the maximum width for roadway use already. Although exceptions are made for safety devices, such as mirrors, the addition of another foot of clearance needed makes the vehicle harder to operate in the urban environment and potentially more dangerous to pedestrians and other fixed objects and more prone to be damaged. For this project, these LIDARs were mounted in retractable / extendable assemblies. This adds cost, complexity, cpu loading and additional maintenance issues to the system. These were operated using the vehicle's air system. These assemblies were computer controlled in order to implement a reflexive behavior for self preservation, present a lower profile in tight situations and retract if it looks like it was going to hit something in its path. Using a fixed mounted front bumper system not only reduces the cpu loading but also the interfaces necessary to extend and retract the LIDARs.

Reliable detection: Reliable object detection is crucial for proper system operation. There are two types of detection failures we have observed fairly frequently that significantly degrade system performance: missing returns and ground returns.
With the SICK sensors, missing returns which occurs both due to weak returns from low reflectivity objects and due to too-strong returns from nearby high-reflectivity objects. We don't understand exactly why the LIDAR fails to detect, and can only speculate on possible fixes. It would help for the sensor to have a larger dynamic range and use a different wavelength.

Ground returns occur when the scanner sees the ground, either because the ground is not flat (a hill) or the bus tilts to the side (going around a turn.) In the current system, ground returns are interpreted as potential collisions, and are one of the largest causes of false

alarms. If the scanner had multiple beams spreading out vertically, or in some other way could measure multiple points vertically on the same object, this would greatly reduce false alarms from ground returns, because it would be easy to determine whether the object is more or less flat on the ground, or sticks up significantly. Multiple beams would also give us more chances to detect any given object, so would reduce missing returns as well.

Resolution: For the current side LIDARs one can set the angular resolution to 0.25^o, 0.5^o, or 1^o. The smaller resolutions have the tradeoff of reduced update rate and interlacing. The 0.25^o resolution has half the FOV. We are using the side LIDARs set to 1 degree azimuth resolution and 1cm range resolution. The position uncertainty is dominated by the azimuth resolution at ranges typically seen in the collision warning system. This means that in some sense the sensor is unbalanced for our purposes. The range resolution could be reduced without compromising performance, or alternatively the azimuth resolution could be increased to exploit the range resolution.

A characteristic of the SICK, and of many other possible similar designs, is that the range accuracy is roughly independent of range, whereas the position uncertainty due to azimuth resolution increases linearly with range. In any such scanner, there is one range at which the position error from range and azimuth is equal, where the scanner can be considered balanced. For the SICK with 1 degree resolution, this is approximately 2 meters (using a range accuracy of +/- 2cm to allow for noise.) To be balanced at 8 meters (a more typical range in the collision avoidance system), we would need to either increase the angular resolution to 0.25 degrees or reduce the range accuracy to +/- 8cm.

If the range accuracy was specified as percentage of the range, then the range error scales proportionately with the azimuth resolution uncertainty, so the measurement accuracy would be balanced at all ranges. The balanced RMS range accuracy as a percent of range is then about $25*sin(angular\ resolution)$, or 0.4% for one degree angular resolution.

The azimuth resolution can be increased to 0.5 degrees by using an interlaced mode where two consecutive scans are combined (reducing the update rate to 37 Hz.) We don't use the 0.5 degree interlace mode because it creates strong artifacts on moving objects, and also because the total amount of data is not actually increased (due to the drop in update rate.) With algorithmic improvements in the tracker, it should be able to tolerate the interlace artifacts, and then there would be some benefit to using interlace.

Range: The current side LIDARs are specified to be accurate to 50 meters and can see as far as 80 meters. As with range accuracy, maximum range should also be balanced with azimuth resolution. As range becomes large, the points become so far apart that any return becomes largely useless. We require at least three points on an object to create a track. Because of this, with 1 degree azimuth resolution, small objects such as pedestrians cannot be tracked above about 20 meters. For side collision warning, a reliable LIDAR range of 15 meters would be adequate. See however, the discussion of detection reliability.

Though we have seen detection fail at ranges of only a few meters, we do not suppose that the SICK is failing to meet it spec. The problem is that real-world objects may have reflectivity that differs significantly from the standard target used in the performance spec. A lower maximum range would not harm system performance as long as it did not further degrade detection reliability. The main conclusion here should be that the scanner range specification is not a valid indication of the actual detection range in the real world, and that although the SICK specification looks in excess of requirements, the observed scanner performance is one of the main limits on system performance.

Update rate: The current side LIDARs output 75 scans a second at 1 degree resolution. The scan update rate should be balanced against maximum speed and size of objects which we want to track. If an object moves too far between two scans, then it is difficult to create a track from consecutive measurements. With the current tracker configuration, we could tolerate an update rate as low as 25 scans per second and still track objects

moving at 20 meters/sec. At 10 scans/sec, the max speed would be reduced to 8 meters/sec. A lower update rate could help with cost reduction.

Expense: The current price of these LIDARs makes an advanced ICWS prohibitively expensive for commercial applications. Designing a commercially deployable sensor would require a certain amount of non recurring expense; the recurring expense could be reduced significantly.

Synchronization of scanners: Currently the data from each scanner is analyzed separately all the way up to the level of warning generation. If the scanners are appropriately synchronized, the raw data can be fused to achieve a virtual 360 degree scanner. This then allows a single algorithm to compute front and side object detections and velocities, with no discontinuities at the limits between two scanners.

Eye Safety: Some trade offs will have to be performed to ensure that the laser scanner will be eye safe. This is less of a potential problem with the current configuration of SICK laser scanners due to the rotating mirror. However, it needs to be part of the design specifications.

Essentially we are looking for a system with:

1. A lower profile so it won't stick too far out of the side of the bus (Coke can size with remote electronics may be one way to go)
2. About 270 degrees FOV
3. Weather resistant, since it would have to operate in the rain and snow
4. Update rate of at least 10 scans/sec, 25 scans/sec preferred.
5. Non-interlaced azimuth resolution of 1.0 degree or better.
6. Reliable detection of real-world objects (not standard targets) to a range of about 15 meters.
7. RMS range error of 0.4% of measured range or 4cm, whichever is greater (for balanced performance with 1 degree azimuth resolution.)

8. Although not required, system false alarms could be significantly reduced if the scanner had two or more scan beams spreading perhaps 0.5 degrees above and below the horizontal scan plane.
9. Be Eye-safe

All of these specifications have to be analyzed as to their effect on the warning algorithm performance and system cost. This is more of an engineering effort at this point and not research.

1.44.5 Integrate a Rear Collision Warning System

A Rear Collision Warning System could be integrated within the same framework as the FCWS and SCWS systems. For a minimal approach, the warning to drivers approaching the rear of the bus at an unsafe speed would not require transit operator involvement at all. A maximal approach would place two additional 270 degree LIDARs on the rear corners of the transit bus. The total of these two and the front two would provide redundancy and total surround sensing of the transit bus. This would make the algorithms more robust, especially for the side object tracking. It would also allow objects moving from the rear of the bus to the sides to be picked up more quickly and identified sooner. Some work would need to be done to see if the DVI would be modified to include rear objects. Although buses usually do not back up while in revenue service, it does sometime happen, so it makes sense to supply as a minimum a light to indicate an object is behind the bus.

1.44.6 Training

Buses equipped with the ICWS such as the advanced prototypes could be used not only for CWS functions, but could be used to provide training of transit operators. Through the use of feedback from the cameras and bus state information, instructors could provide feedback of how operators performed on training courses or on the road. As a training device, transit agencies may opt for more functionality and a higher price tag.

1.45. Areas for Future Research

Although the current phase CWS project has made significant progress for an ICWS that can effectively provide drivers with warnings and alerts under hazardous situations, some technical issues still remain and deserve additional research. The project team has identified the following research areas:

1.45.1 Transit bus data

A considerable amount of data will have been collected by the end of the ICWS project. In fact, the volume will be so great that many interesting secondary analyses will not be feasible to conduct due to time and resource limitations in the ICWS project. In this section we will identify a few potentially interesting analyses that could be explored at a later date. This is not an exhaustive list – it is only a small sampling of opportunities.

1.45.1.1 Inputs for operator training

Given the highly instrumented nature of the bus it is feasible to identify opportunities for new or modified operator training. For example, improved documentation of specified scenarios could be used to guide mirror use training. Another example would explore whether it is possible to induce safer pedestrian behavior as a result of door opening or bus stop approach actions.

1.45.1.2 Inputs for public education

During the course of safety analyses it may become obvious that certain behaviors by the driving public are extremely indicative of potential harm, such as cutting in front of a bus and braking. Isolating and breaking down such actions can identify and verify practices that may be in need of public education.

1.45.1.3 Inputs for roadway infrastructure

Using the data set we will be able to identify and verify roadway fixture geometries that produce difficult bus operations (e.g., road geometry garbage cans placed too close to curb, parking spots too close to corners, etc). These can be used to assist infrastructure specifications and parking enforcement activities (e.g., ticket and tow cars illegally parked near corners).

1.45.1.4 Verification of risky behavior predictors in the driving public

As a result of the sensor data we will be able to characterize how the driving public behaves irrespective of the bus. From this we may be able to identify and verify characteristics of vehicle motion that are indicative of potential dangerous behavior. For example, a vehicle that is tracked for 30 seconds may only exhibit dangerous behavior during the last 5 seconds (e.g., tailgating). It may be possible to correlate distinctive motions (e.g., rapid lane changes) or vehicle characteristics (e.g., dented body panels) with confirmed risky behavior. Certain unverified suspicions could be examined using real, anonymous data.

1.45.2 Unify the FCWS and SCWS Tracking and Warning Algorithms

Currently, the Advanced ICWS uses different object detection and tracking algorithms and different warning algorithms for the forward looking sensors and the side looking sensors. The development of a common object detection, tracking, and warning algorithm using the 360 degree virtual sensor would greatly reduce the complexity of the software, with all the benefits of reduced development time, increased robustness, and less maintenance. It will probably also give the driver a better intuition about the whole system, because the front and the side behave in a more consistent way.

1.45.3 Integrate ICWS with other electronic vehicle systems

The following systems offer an opportunity for standardization and cost savings:

1. Annunciation Systems – This would provide dual usage of GPS based information.
2. Bus Tracking Systems – This could add dispatch capability. The cell phone interface could call home if an incident occurs
3. Provide inputs to bus electronics standards J1939 – As standards evolve, they should begin to accommodate the collision warning functions. Perhaps a separate safety bus should be defined.

1.45.4 Improvements to the object tracking algorithms (DATMO)

Improvements to the warning algorithm heuristics and object models for pedestrians, bicyclists and vehicles could be made. Areas for improvement would be the ability to recognize parked cars and longer distances from curb.

In the SCWS the warning algorithm can accommodate models for the bus and the objects. Currently we have an enhanced model for bus behavior, but only very simple models for pedestrian, cars, and fixed objects. There is no separate model for other objects like bicyclists, motorcycles, animals, and vegetation. Models for all objects can be developed or enhanced. The warning algorithm can also make use of environmental information like the position of the curb. Possible enhancements to the system are:

1. Increase the look ahead of the curb position and identifying parked cars alongside the road. 4: Knowledge - Knowledge about road and route could be used to eliminate false alarms triggered by road-side objects or out-of-lane objects.
2. Use more sophisticated algorithms to improve the response time of the turn rate and acceleration estimates. These currently are only marginally useful.
3. Improve the segmentation procedure so that it works better in highly cluttered environments (where objects are closer than 0.7 meters.)
4. Assign classifications such as car, pedestrian, bicycle, wall, and ground return to tracks based on the change in shape and motion over time. This would allow us to predict motion more accurately by using appropriate distinct dynamic models, and could also reduce false alarms by detecting tracks that change in ways atypical of good tracks.

1.45.5 Improvements to FCWS warning algorithm

Improvements to the FCWS warning algorithm would also be desirable in order to enhance the performance of the CWS system. The improvements are mainly in the following areas:

1. Transition of vehicle models: It was found that nonholonomic model is good for moving targets in terms of estimating yaw-rate and moving direction. However at lower speed, due to short displacement in processing time, it is hard to detect

moving direction. In this case free moving model is better. The transition of vehicle models from higher speed to lower speed and vice versa needs improved.

2. Scenario parsing: This has been a topic since the beginning of the project. However it is not well resolved yet. It needs to consider the relationship among all objects and subject vehicle and infrastructure. Current algorithm only detects straight road in-lane objects, and cannot avoid false warnings due to lack of lane information and driver status.
3. Driver model: Driver's field operational data were analyzed leading to the empirical threshold settings. However more complex driver model may help to tell whether driver is attentive. Collision warning is supposed to be issued only when driver is inattentive.
4. Knowledge - Knowledge about road and route could be used to eliminate false alarms triggered by road-side objects or out-of-lane objects.

1.45.6 Sensor Fusion

Each of the sensors that are currently available for obstacle detections collision warning system has its advantages and disadvantages. For example, LIDARs provide good range and azimuth measurements but do not function properly under the bad weather conditions. RADARs on the other hand, work with most of weather conditions but do not provide the level of accuracy that LIDARs provide. Field testing also indicates that additional information about road geometry and roadside furniture may help to reduce false detections. It is likely more than one type of sensors will be used in order to enhance the reliability of the system. When sensor options are considered, sensor fusion can help to maximize the benefits of these sensors. This is an research area that is currently being investigated under the ICWS program and it likely will require continuous investigation beyond this program.

1.45.7 Develop an under the bus sensor

The current SCWS algorithms employ an inferred under bus logic which looks for the disappearance of an object around the wheel wells of the transit bus. As described more fully in the text concerning the warning algorithms, a positive indication from a specific sensor would be a better indication of the presence of something in front of the wheels.

The inferred method we are currently using is fooled by occlusions, multiple moving objects in the same vicinity and the inability to resolve people boarding the bus and someone slipping near the doorway under the bus, since both objects disappear within the same vicinity. The current algorithms detect too many false positives to be used as a strong measure of a problem. If a sensor could be developed that gave fewer false positives, then stronger operator interactions such as getting out of the bus to verify could be implemented. As it stands, we can only give an indication that there might be a problem.

Appendix A: Acronym Definitions

ACRONYM	DEFINITION
APTA	American Public Transportation Association
ARQ	Acceleration Required
CALTRANS	California Department of Transportation
CMU	Carnegie Mellon University
CWS	Collision Warning System
DATMO	Detection And Tracking of Moving Objects
DTCMO	Detection, Tracking and Classification of Moving Objects
DVI	Driver Vehicle Interface
EODS	Enhanced Object Detection System
FCWS	Frontal Collision Warning System
FMI	Foster Miller, Inc
FTA	Federal Transit Administration
HF	Human Factors
IBEO	German Laser Scanner Company
ICD	Interface Control Document
ICWS	Integrated Collision Warning System
IRB	Institutional Review Board
IVN	In Vehicle Network
LED	Light Emitting Diode
PAT	Port Authority of Allegheny County
PATH	Partners for Advanced Transit and Highways
PENNDOT	Pennsylvania Department of Transportation
RAID	Redundant Array of Inexpensive Disks
RI	Robotics Institute
SAMTRANS	San Mateo County Transit District
SCWS	Side Collision Warning System
SICK	German manufacturer of laser scanners
SLAM	Simultaneous Localization and Mapping

SV	Subject Vehicle
TTC	Time to Collision

Appendix B: Related Documents

Assessment of Technologies Supplementary Report April 2002, Christoph Mertz

ICWS Driver-Vehicle Interface April 2003 Design Specification, prepared by Aaron Steinfeld, Carnegie Mellon University and Joanne Lins, UC Berkeley

Integrated Collision Warning Systems Interface Control Document dated August 2004 prepared by the California PATH Program, University of California at Berkeley and the Robotics Institute, Carnegie Mellon University

Evaluation of Integrated Collision Warning System Proposal prepared by the Robotics Institute, Carnegie Mellon University and the California PATH Program, University of California at Berkeley. *In collaboration with*

California Department of Transportation (Caltrans)
Gillig Co.
Pennsylvania Department of Transportation
Port Authority Transit (PAT)
San Mateo Transit (Samtrans)

Transit Bus Collision Warning Systems Integration Program Proposal dated 5/23/01 prepared by:

California PATH Program, University of California at Berkeley
California Department of Transportation (Caltrans)
Clever Devices, Inc
Gillig Co.
Pennsylvania Department of Transportation (PennDOT)
Port Authority Transit (PAT)
Robotics Institute, Carnegie Mellon University
San Mateo Transit (Samtrans)

Evaluation Report: Driver Experience with the Enhanced Object Detection System for Transit Buses Final Report dated December 12, 2003, Battelle / TRI

Transit Bus Frontal Collision Warning System Final Report dated August 2003, Xiqin Wang, Joanne Lins, Ching-Yao Chan, Scott Johnston, Kun Zhou, Aaron Steinfeld, Matt Hanson, and Wei-Bin Zhang

Side Collision Warning System (SCWS) Performance Specifications dated May 2, 2002 prepared by the Robotics Institute, Carnegie Mellon University

Transit Bus Collision Warning Systems Performance Specifications Interface Requirements (Draft) dated October 25, 2002 by the California PATH Program, University of California at Berkeley and the Robotics Institute, Carnegie Mellon University

Development and Validation of Functional Definitions and Evaluation Procedures For Collision Warning/Avoidance Systems dated August 1999, Kiefer, R. J., LeBlanc D. J. , Palmer M. D., Deering R. K., and Shulman M. A., NHTSA Technical Report

Forward Collision Warning Requirement Projects: Refining the CAMP Crash Alert Timing Approach by "Examining" Last Second Braking and Lane Changing Maneuvers Under Various Kinematic Conditions dated , Jan. 2003, Kiefer, R. J., Cassar, M. T., Flannagan C. A., LeBlanc D. J. , Palmer M. D., Deering R. K., and Shulman M. A., NHTSA

Appendix C: Published Papers

Publications funded by this program

Eye-Safe Laser Line Striper for Outside Use, C. Mertz, J. Kozar, J. R. Miller and C. Thorpe

Multiple Sensor Fusion for Detecting Location of Curbs, Walls, and Barriers, Romuald Aufreire, Christoph Mertz and Charles Thorpe

A 2D Collision Warning Framework based on a Monte Carlo Approach. Christoph Mertz

Simultaneous Localization, Mapping and Moving Object Tracking, C. Wang doctoral dissertation, tech. report CMU-RI-TR-04-23, Robotics Institute, Carnegie Mellon University, April, 2004

Development of the Side Component of the Transit Integrated Collision Warning System, Aaron Steinfeld, David Duggins, Jay Gowdy, John Kozar, Robert MacLachlan, Christoph Mertz, Arne Suppe, Charles Thorpe, Chieh-Chih Wang

Previous Publications

Dressed Human Modeling, Detection, and Parts Localization, Thesis for Liang Zhao (CMU-RI-TR-01-19) July 26, 2001

Driving in Traffic: Short-Range Sensing for Urban Collision Avoidance, Chuck Thorpe, Dave Duggins, Jay Gowdy, Rob MacLachlan, Christoph Mertz, Mel Siegel, Arne Suppe, Bob Wang, Teruko Yata

Facts and Data Related To Bus Collisions Interim Report April 11, 2002

A New Focus for Side Collision Warning Systems for Transit Buses, May 2000

Side Collision Warning Systems for Transit Buses, Christoph Mertz, Sue McNeil, and Charles Thorpe

Side Collision Warning Systems for Transit Buses: Functional Goals, CMU-RI-TR-01-11, David Duggins, Sue McNeil, Christoph Mertz, Chuck Thorpe, Teruko Yata dated 5/14/01

Simultaneous Localization and Mapping with Detection of Moving Objects, Chieh-Chih Wang and Chuck Thorpe

State of the Art of Technology Part I: General Overview, Christoph Mertz dated April 15, 2002

State of the Art of Technology Part II: Investigation of specific sensors, Christoph Mertz dated April 15, 2002

Static Environment Recognition Using Omni-camera from a Moving Vehicle, Teruko Yata, Chuck Thorpe, and Frank Dellaert

Stereo and Neural Network-Based Pedestrian Detection, Liang Zhao and Charles E. Thorpe, IEEE Transactions on Intelligent Transportation Systems, Volume 1, No 3 September 2000

"*Studies of Accidents and Cost data for Transit Buses*", Kun Zhou, Wei-Bin Zhang, Gary Glenn, Xiqin Wang, and Ching-Yao Chan, ITS World Congress, Nagoya, Oct. 2004

"*Development of Requirement Specifications for Transit Frontal Collision Warning System- Final Report*", Xiqin Wang, Joanne Chang, Ching-Yao Chan, Scott Johnston, Kun Zhou, Aaron Steinfeld, Matt Hanson, and Wei-Bin Zhang, PATH Technical Report, UCB-ITS-PRR-2004-14, May 2004

"*Development of Requirement Specifications for Transit Frontal Collision Warning System*", Xiqin Wang, Joanne Lins, Ching-Yao Chan, Scott Johnston, Kun Zhou, Aaron Steinfeld, Matt Hanson, Wei-Bin Zhang, PATH Technical Report, UCB-ITS-PRR-2003-29, November, 2003

"A new maneuvering target tracking algorithm with input estimation", Kun Zhou, Xiqin Wang, Masoyashi Tomizuka, Ching-Yao Chang, and Wei-Bin Zhang, American Control Conference, Anchorage, Alaska, 2002

"*Integrated Multi-Sensor System: A Tool for Investigating Approaches for Transit Frontal Collision Mitigation*", Xiqin Wang, Wei-Bin Zhang, Scott Johnston, Dan Empey, and Chinyao Chan, ITS World Congress, Sydney, Australia, 2001

Functional Analysis of Frontal Collision Warning System, M. El koursi, E.Lemaire, Ching-Yao Chan, Wei-Bin Zhang, ITS World Congress, Sydney, Australia, 2001

"*Studies of Accident Scenarios for Transit Bus Frontal Collisions*", Ching-Yao Chan, Kun Zhou, Xi-Qin Wang and Wei-Bin Zhang, ITS America Annual Meeting, Orlando, Florida, 2001

"Scenario Parsing in Transit Bus Operations For Experimental Frontal Collision Warning Systems", *Ching-Yao Chan, Xi-Qin Wang, Wei-Bin Zhang*, *IEEE Intelligent Vehicle Conference, Tokyo, Japan, 2001*

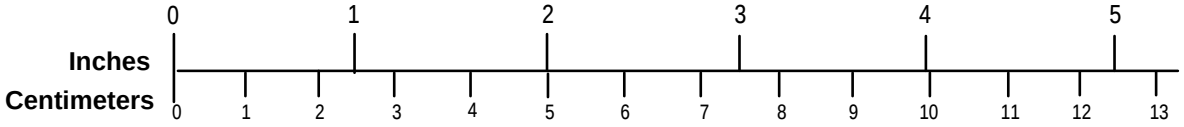
"*Develop Performance Specifications for Frontal Collision Warning System for Transit buses"*, Wei-Bin Zhang, et al. 7th Intelligent Transportation Systems World Congress Turin, Italy, November 6-11, 2000

"*Preliminary Safety Analysis of Frontal Collision Aoidance"*, El Miloudi El Koursi, Chinyao Chan, Wei-Bin Zhang, 3rd IEEE Inernational Conference on Intelligent Transportation Systems, Dearborn, MI, Oct. 1-3, 2000

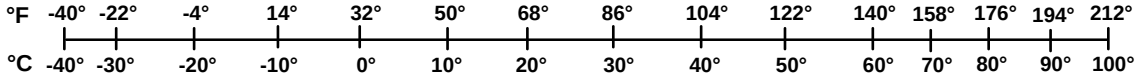
Appendix D: Conversion Tables

ENGLISH TO METRIC	METRIC TO ENGLISH
LENGTH (APPROXIMATE)	**LENGTH (APPROXIMATE)**
1 inch (in) = 2.5 centimeters (cm)	1 millimeter (mm) = 0.04 inch (in)
1 foot (ft) = 30 centimeters (cm)	1 centimeter (cm) = 0.4 inch (in)
1 yard (yd) = 0.9 meter (m)	1 meter (m) = 3.3 feet (ft)
1 mile (mi) = 1.6 kilometers (km)	1 meter (m) = 1.1 yards (yd)
	1 kilometer (km) = 0.6 mile (mi)
AREA (APPROXIMATE)	**AREA (APPROXIMATE)**
1 square inch (sq in, in^2) = 6.5 square centimeters (cm^2)	1 square centimeter (cm^2) = 0.16 square inch (sq in, in^2)
1 square foot (sq ft, ft^2) = 0.09 square meter (m^2)	1 square meter (m^2) = 1.2 square yards (sq yd, yd^2)
1 square yard (sq yd, yd^2) = 0.8 square meter (m^2)	1 square kilometer (km^2) = 0.4 square mile (sq mi, mi^2)
1 square mile (sq mi, mi^2) = 2.6 square kilometers (km^2)	10,000 square meters (m^2) = 1 hectare (ha) = 2.5 acres
1 acre = 0.4 hectare (he) = 4,000 square meters (m^2)	
MASS - WEIGHT (APPROXIMATE)	**MASS - WEIGHT (APPROXIMATE)**
1 ounce (oz) = 28 grams (gm)	1 gram (gm) = 0.036 ounce (oz)
1 pound (lb) = 0.45 kilogram (kg)	1 kilogram (kg) = 2.2 pounds (lb)
1 short ton = 2,000 pounds (lb) = 0.9 tonne (t)	1 tonne (t) = 1,000 kilograms (kg) = 1.1 short tons
VOLUME (APPROXIMATE)	**VOLUME (APPROXIMATE)**
1 teaspoon (tsp) = 5 milliliters (ml)	1 milliliter (ml) = 0.03 fluid ounce (fl oz)
1 tablespoon (tbsp) = 15 milliliters (ml)	1 liter (l) = 2.1 pints (pt)
1 fluid ounce (fl oz) = 30 milliliters (ml)	1 liter (l) = 1.06 quarts (qt)
1 cup (c) = 0.24 liter (l)	1 liter (l) = 0.26 gallon (gal)
1 pint (pt) = 0.47 liter (l)	
1 quart (qt) = 0.96 liter (l)	
1 gallon (gal) = 3.8 liters (l)	
1 cubic foot (cu ft, ft^3) = 0.03 cubic meter (m^3)	1 cubic meter (m^3) = 36 cubic feet (cu ft, ft^3)
1 cubic yard (cu yd, yd^3) = 0.76 cubic meter (m^3)	1 cubic meter (m^3) = 1.3 cubic yards (cu yd, yd^3)
TEMPERATURE (EXACT)	**TEMPERATURE (EXACT)**
[(x-32)(5/9)] °F = y °C	[(9/5) y + 32] °C = x °F

QUICK INCH - CENTIMETER LENGTH CONVERSION



QUICK FAHRENHEIT - CELSIUS TEMPERATURE CONVERSION



For more exact and or other conversion factors, see NIST Miscellaneous Publication 286, Units of Weights and Measures. Price $2.50 SD Catalog No. C13 10286 **Updated 6/17/98**