Engineering Design Synthesis of Sensor and Control Systems for Intelligent Vehicles

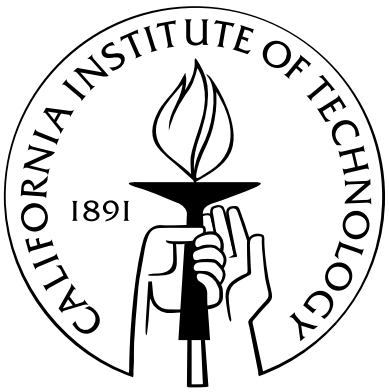
Thesis by

Yizhen Zhang

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy



California Institute of Technology

Pasadena, California

2006

(Defended May 3, 2006)

dedicated to my parents, my family, and my friends ...

Acknowledgements

There are a lot of people to whom I am grateful for their help and support.

First of all, I would like to especially thank my advisor, Professor Erik K. Antonsson, for his great guidance, valuable advice, full support, warm encouragement, and patience. He gives his graduate students quite a lot freedom in exploring and pursuing their own interests, and lets them enjoy their research in their own way as much as possible.

My co-advisor, Professor Alcherio Martinoli, introduced me to the field of intelligent vehicles and evolutionary computation, and helped me to start quickly with my graduate research during the early years. I have learned a lot from his research ideas and knowledge, as well as his research attitude, enthusiasm, and persistence.

Many thanks to Professor Karl Grote, who always tries to help me out whenever possible. Also I am thankful to Professor Maria Yang for her help and support.

I would like to thank my thesis committee members who are not mentioned above—Professor Richard Murray, Professor Joel W. Burdick, and Professor Ken Pickar, for their support and helpful discussions.

I am also grateful for the sponsorship of Caltech Engineering and Applied Science Division Fellowship, Delphi Delco Electronic Systems, Caltech Center for Neuromorphic Systems Engineering as part of the Engineering Research Centers Program of the National Science Foundation, NASA Glenn Research Center, and the National Center for Metropolitan Transportation Research (METRANS) under the California Department of Transportation (Caltrans).

Moreover, I appreciate the help and support from the people I have worked with, including Fabien Nicaise, Tomonori Honda, Bingwen Wang, Olivier Michel, Naveed Near-Ansari, Michael Potter, Dr. Sanza Kazadi, Piyush Prakash, Noé Lutz, Nikolaus Correll, Jim Pugh, Dr. Rodney Goodman, Dr. Ian Kelly, Kjerstin Williams, William Agassounon, Adam Hayes, Maria Koeper, Lynn Burgess, etc.

Finally I owe special thanks to my parents, my family, and all my friends for their endless love and support.

Abstract

This thesis investigates the application of formal engineering design synthesis methodologies to the development of sensor and control systems for intelligent vehicles with a series of meaningful case studies.

A formal engineering design synthesis methodology based on evolutionary computation is presented, with special emphasis on dealing with modern engineering design challenges, such as high or variable complexity of design solutions, multiple conflicting design objectives, and noisy evaluation results, etc., which are common when design and optimization of distributed control systems such as intelligent vehicles are considered. The efficacy of the evolutionary design synthesis method is validated through multiple different case studies, where a variety of novel design solutions are generated to represent different engineering design trade-offs, and they have achieved performances comparable to, if not better than, that of hand-coded solutions in the same simplified environment. More importantly, this automatic design synthesis method shows great potential to handle more complex design problems with a large number of design variables and multi-modal noise involved, where a good hand-coded solution may be very difficult or even impossible to obtain. In summary, the evolutionary design synthesis methodology appears promising to

- propose a variety of good, novel design solutions according to specified fuzzy fitness functions;
- deal with uncertainty in the problem efficiently;
- adapt to the collective task nature well.

In addition, multiple levels of vehicle simulation models with different computational cost and fidelity as well as necessary driver behaviors are implemented for different types of simulation experiments conducted for different research purposes. Efforts are made to try to get as much as possible out of limited computational resources, such that good candidate solutions can be generated efficiently with less computational time and human engineering effort.

Furthermore, different threat assessment measures and collision avoidance algorithms are reviewed and discussed. A new threat assessment measure, time-to-last-second-braking (T_{lsb}), is proposed, which directly characterizes human natural judgment of the urgency and severity of threats in terms of time. Based on driver reaction time experimental results, new warning and overriding

criteria are proposed in terms of the new T_{lsb} measure, and the performance is analyzed statistically in terms of two typical sample pre-crash traffic scenarios. Less affected by driver behavior variability, the new criteria characterize the current dynamic situations better than the previous ones, providing more appropriate warning and more effective overriding at the last moment. Finally, the possibility of frontal collision avoidance through steering (lane-changing) is discussed, and similarly the time-to-last-second-steering (T_{lss}) measure is proposed and compared with T_{lsb}.

Contents

Acknowledgements **iv**

Abstract **v**

1 Introduction **1**

1.1 Intelligent Vehicles 1

1.1.1 Background 2

1.1.2 Introduction 2

1.2 Motivation 3

1.3 Thesis Contributions 4

1.4 Thesis Overview 5

2 Evolutionary Computation Design Methodologies **7**

2.1 Evolutionary Computation 8

2.1.1 Genetic Algorithms 9

2.1.2 Evolution Strategies 10

2.1.3 Evolutionary Programming 10

2.1.4 Genetic Programming 11

2.2 Evolutionary Computation Theory 11

2.3 General Structure 12

2.4 Implementation in Design Synthesis 14

2.4.1 Encoding 14

2.4.2 Initialization 14

2.4.3 Crossover 15

2.4.4 Mutation 16

2.4.5 Evaluation and Re-evaluation 16

2.4.6 Selection 16

2.4.7 Termination 17

2.4.8 Final Evaluation 17

3 Multi-level Simulation and Modeling of Intelligent Vehicles 18

3.1 Literature Review . . . 19
3.2 Vehicle Simulation . . . 20
3.2.1 Point Models . . . 21
3.2.1.1 Theoretic Models . . . 21
3.2.1.2 Probabilistic Models . . . 22
3.2.2 Kinematic Embodied Simulation . . . 22
3.2.3 Dynamic Embodied Simulation . . . 27
3.2.3.1 Vehicle Dynamics Overview . . . 27
3.2.3.2 Basic Vehicle Model . . . 30
3.2.3.3 Joint Models . . . 31
3.2.3.4 Friction Model . . . 34
3.2.3.5 Rotational Motions . . . 36
3.3 Driver Models . . . 38
3.3.1 Helbing Model . . . 38
3.3.2 Rule-based Model . . . 39
3.3.3 PID Control Model . . . 42
3.3.4 Evolved Model . . . 44

4 Evolutionary Design of Sensory Systems 45

4.1 Introduction . . . 45
4.2 Background . . . 46
4.3 Problem Definition . . . 48
4.3.1 Sensor Parameters . . . 48
4.3.2 Evaluation Tests . . . 50
4.3.3 Fitness Function . . . 52
4.3.3.1 Preferences . . . 53
4.3.3.2 Aggregation . . . 54
4.4 Evolutionary Experiments . . . 56
4.5 Results and Discussion . . . 57
4.5.1 Comparison of Different Evaluation Tests . . . 57
4.5.2 Evolving Engineering Design Trade-offs . . . 61
4.6 Conclusion . . . 67

5 Evolution of Neural Controllers 68

5.1 Background . . . 68
5.2 Evolution of Artificial Neural Networks . . . 69

5.2.1 Encoding . . . 70
5.2.2 Initialization . . . 71
5.2.3 Genetic Operations . . . 71
5.3 Case Study 1: Collective Robotic Inspection . . . 72
5.3.1 Application Background . . . 73
5.3.2 Problem Formulation . . . 73
5.3.3 Hand-coded Controller . . . 75
5.3.4 Results . . . 76
5.3.4.1 Single Robot Single Object (SRSO) Scenario . . . 77
5.3.4.2 Single Robot Multiple Objects (SRMO) Scenario . . . 79
5.3.4.3 Multiple Robots Multiple Objects (MRMO) Scenario . . . 80
5.4 Case Study 2: Driver Behavior Modeling . . . 83
5.5 Conclusion . . . 85

6 Collision Avoidance System **86**
6.1 Previous Work . . . 87
6.1.1 Measures Defined . . . 87
6.1.2 Driver Reaction Time . . . 88
6.1.3 Collision Warning Systems . . . 90
6.2 Warning and Overriding Algorithms . . . 93
6.2.1 Mazda Algorithm . . . 94
6.2.2 Honda Algorithm . . . 94
6.2.3 Berkeley Algorithm . . . 95
6.2.4 NHTSA Alert Algorithm . . . 96
6.2.5 CAMP Alert Algorithm . . . 97
6.2.6 Other Alert Algorithms . . . 98
6.3 New Criterion Proposal . . . 98
6.3.1 Time-to-last-second-braking (T_{lsb}) Measure . . . 99
6.3.2 Scenario 1: Lead Vehicle Stopped or Moving Slowly ($a_L = 0$) . . . 99
6.3.3 Scenario 2: Lead Vehicle Decelerating ($a_L < 0$) . . . 101
6.3.4 General Scenario . . . 103
6.3.5 Error Estimation of the T_{lsb} Measure . . . 103
6.3.6 Warning/Overriding Criteria in terms of T_{lsb} . . . 105
6.4 Analysis . . . 107
6.4.1 Performance in terms of P_{miss} versus P_{FA} . . . 107
6.4.2 Comparison under Scenario 1 ($a_L = 0$) . . . 108

6.4.3 Comparison under Scenario 2 ($a_L < 0$) 109
6.4.4 Last-second Braking versus Steering 113
6.5 Conclusion 117

7 Conclusions **119**
7.1 Summary 119
7.2 Limitation and Future Directions 120

Bibliography **122**

List of Figures

2.1 The Evolutionary Optimization Loop Used in the Automatic Engineering Design Synthesis Process . . . 12
2.2 Illustration of One-point Crossover Scheme for Two Vectors of Different Lengths . . . 15

3.1 Sample Point Vehicle Models with Speed v and Collision Disk with Diameter D Moving on a Three-lane Highway . . . 22
3.2 Screen Shot of Traffic Simulation in Kinematic Embodied Simulator—Webots . . . 23
3.3 Close-up of the Real Khepera Robot (a) and its Simulation Model in Webots (b) with its Distance Sensor Rays Illustrated in Solid Red Lines . . . 24
3.4 2D Vehicle Occurrence PDF: The Host Vehicle Sits at the Origin Facing the Positive y Axis. . . . 25
3.5 1D Vehicle Occurrence PDF in Cartesian (a) and Polar (b) Coordinates . . . 26
3.6 Significant Forces Acting on a Two-axle Vehicle . . . 27
3.7 The Friction Ellipse Concept Relating the Maximum Cornering Force F_y to a Given Longitudinal Force F_x . . . 28
3.8 Cornering Characteristics of Car Tires . . . 29
3.9 Construction of a Friction Ellipse for a Given Slip Angle . . . 30
3.10 Screen Shot of the Dynamic Embodied Vehicle Model . . . 30
3.11 Sample Freeway Traffic Simulation with Dynamic Embodied Vehicle Models . . . 32
3.12 Schematic View of a Simple Two-axle Vehicle Model [Epiney 2004] . . . 32
3.13 The (a) Hinge and (b) Slider Joints Defined in ODE . . . 33
3.14 The Hinge-2 Joint Defined in ODE . . . 33
3.15 Variation of Friction Coefficient vs. Wheel Slip Curves on (a) Dry Asphalt and (b) Dry Asphalt, Loose Gravel, and Ice [Harned et al. 1969] . . . 34
3.16 Typical Relationship between Friction Coefficient μ and Wheel Slip i . . . 35
3.17 The Original (a) and Modified (b) ODE Friction Models: F_{app} represents *applied* tractive or braking effort on the wheel while F_{eff} represents *effective* force available from the ground contact, with f_m representing the maximum friction limit. . . . 36
3.18 Illustration of the Three Rotational Motions: Pitch, Yaw, and Roll . . . 37

3.19 The Simulated Vehicle Pitch, Yaw, and Roll Motions with Steering Angles 37
3.20 The Logic Scheme of a Simple Car-following Behavior 40
3.21 The Logic Scheme of a Simple Lane-keeping Behavior 41
3.22 The Logic Scheme for Initiating a Lane Change 41
3.23 The Logic Scheme of a Simple Lane-changing Behavior 42
3.24 (*Top*): Time Histories of Vehicle Lane Deviation Error (E), Integral Lane Deviation Error ($iE \equiv \int_0^t E(\tau)d\tau$), and Lane Position Y in terms of Lane Widths ($W_l = 3.5\,\mathrm{m}$); (*Bottom*): Time Histories of Vehicle Steering Angle δ and Wheel Speed ω 43

4.1 Sensor Parameters and the Target Detection Region 48
4.2 Graphical Representation of Different Types of Evaluation Tests Based on Point Model 51
4.3 Approximate Relative Time Costs of Different Evaluation Tests Plotted on a Log Scale 52
4.4 Designer's Fuzzy Preferences for $Coverage$ and $Total\_cost$ 54
4.5 Illustration of the One-point Crossover Scheme for Two Sensory Systems with Different Numbers of Sensors: The sectors (or lines) represent the sensor scanning areas (or rays). 56
4.6 (*Top*): Performances of the Best Design Solutions Evolved under Different Conditions and Evaluation Tests with each Final Noise Test Conducted under the Same Evaluation Test Used in Evolution, respectively; (*Bottom*): Numbers of Sensors Used by the Best Design Solutions Evolved with Variable Number of Sensors 58
4.7 Performances of the Best Design Solutions Evolved under Different Conditions and Evaluation Tests with the Final Noise Tests Conducted under the 2D Full Coverage Evaluation Test (*Top*) and the Kinematic Embodied Evaluation Test (*Bottom*), respectively 60
4.8 Evolution Process of the Population Mean Fitness and Preferences (*Top*) and the Best Sensor Configuration Evolved (*Bottom*) with $s = 0$ and $w = \dfrac{3}{17}$ 63
4.9 Evolution Process of the Population Mean Fitness and Preferences (*Top*) and the Best Sensor Configuration Evolved (*Bottom*) with $s = -\infty$ and arbitrary w (i.e., $Fitness = \min(\mu_{coverage}, \mu_{cost})$) 64
4.10 Evolution Process of the Population Mean Fitness and Preferences (*Top*) and the Best Sensor Configuration Evolved (*Bottom*) with $s = 0$ and $w = 4$ 65
4.11 Evolved Pareto Frontier for the Design Trade-offs Present in the Case Study 66

5.1 Evolutionary Run for Automatic Robotic Neural Network Controller Synthesis 70
5.2 Top View of the Structure Inspection Experiment Setup in the Kinematic Embodied Simulator (Webots): The bigger blue disks represent the cylindrical objects to be inspected while the smaller green dots are the miniature robots. 74
5.3 The Logic Scheme of the Hand-coded Rule-based Controller for Structure Inspection . 75

5.4 Screen Shot of the SRSO Scenario 77
5.5 Sample Robot Trajectories of the SRSO Scenario for 500 Time Steps (32 s) Using the Hand-coded Rule-based Controller (a) and the Best Evolved ANN Controller (b): "S" represents the constant starting point and "E" the ending points, with "+" symbols placed along the trajectories every 40 time steps (2.56 s). 78
5.6 Sample Robot Trajectories of the SRMO Scenario for 2000 Time Steps (128 s) Using the Hand-coded Rule-Based Controller (a) and the Evolved ANN Controller (b). The dashed lines delimit the wrap-around boundaries. "S" represents the random initial starting points and "E" the ending points. The trajectories are shown in gradually changing colors with "+" symbols placed along the trajectories every 40 time steps (2.56 s). 79
5.7 Sample Robot Trajectories of the MRMO Scenario for 800 Time Steps (51.2 s) Using the Hand-coded Rule-based Controller (a) and the Evolved ANN Controller (b). The dashed lines delimit the wrap-around boundaries, but some trajectories beyond the boundaries are kept to enhance the display. "S" represents the random initial starting points and "E" the ending points. Different robots' trajectories are shown in different colors with different markers placed along the trajectories every 40 time steps (2.56 s). 80
5.8 Coverage Values Achieved by the Hand-coded Rule-based Controller (hndcd) and the Best Controllers Evolved with Different ANN Architectures (refer to the symbols defined in Table 5.1) and Selected according to (a) Minimum and (b) Average Performance for the MRMO Scenario. Each column shows the coverage values (*the green dots*) obtained by one controller during the 100 evaluations in its final noise test and the error bars indicate the standard deviation. 82
5.9 Curved Road Shape for Driver Model Evaluation 84
5.10 NN Inputs and Outputs of a Sample Driver Model Evolved on the Curved Road Shown in Figure 5.9 84

6.1 Hypothetical Reaction Time Distribution [Green 2000] 89
6.2 Warning Curves (solid lines) Parametric in a_L ($v_0 = 48$ mph, $t_r = 1.5$ s, $a_{H_{max}} = -5\,\mathrm{m/s^2}$) with a Sample Time Trajectory (dash dot lines) of $t_h = 2$ s and $a_L = -3\,\mathrm{m/s^2}$ 93
6.3 Interpretation of the Mazda Overriding Algorithm 94
6.4 Interpretation of the Honda Overriding Algorithm 95
6.5 Interpretation of the Berkeley Warning/Overriding Algorithm 96
6.6 Interpretation of the NHTSA Alert Algorithm 96
6.7 T_{lsb} Contours in Seconds with CAMP Data under Scenario 1: Host Vehicle Approaches Stopped or Slow Lead Vehicle ($a_L = a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$). 100

6.8 CAMP Data Represented in terms of the T_{lsb} Measure under Scenario 2: Lead Vehicle Decelerates ($a_{H_{max}} = -5\,\mathrm{m/s^2}$). 102

6.9 Error Distributions of the Estimated T_{lsb} ($T_{lsb,est} - T_{lsb,true}$) due to Sensor Noise under Scenario 1 and Scenario 2 105

6.10 T_{lsb} Contours (solid lines) in Seconds with Various Warning (R_w, dotted lines) and Overriding (R_o, dashed lines) Boundary Curves under Scenario 1: Host Vehicle Approaches Stopped or Slow Lead Vehicle ($a_L = a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$). 110

6.11 T_{lsb} Contours (solid lines) in Seconds with Various Warning (R_w, dotted lines) and Overriding (R_o, dashed lines) Boundary Curves under Scenario 2: Lead Vehicle Decelerates 0 s (a) and 1 s (b), respectively, where both vehicles initially travel at the same speed 70 mph ($a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$). 112

6.12 T_{lsb} Contours (solid lines) in Seconds with Various Warning (R_w, dotted lines) and Overriding (R_o, dashed lines) Boundary Curves under Scenario 2: Lead Vehicle Decelerates ($v_L = 60\,\mathrm{mph}$, $v_H = 70\,\mathrm{mph}$, $a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$). 113

6.13 T_{lsb} and T_{lss} Contours in Seconds under Scenario 1: Host Vehicle Approaches Stopped or Slow Lead Vehicle ($a_L = a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$). 116

6.14 T_{lsb} and T_{lss} Contours in Seconds under Scenario 2: Lead Vehicle Just Starts to Decelerate ($v_L = v_H = 70\,\mathrm{mph}$, $a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$). 117

List of Tables

3.1 Average Values of Coefficient of Road Adhesion [Taborek 1957] 36
3.2 The IDM Parameters 39

4.1 Approximate Time Costs of Different Evaluation Tests 52
4.2 Evolutionary Algorithm Parameters 57

5.1 Different ANN Types Considered in the Evolutionary Design Synthesis 76

6.1 Various Threat Assessment Measures Defined in the Literature 88
6.2 Estimates of Unexpected Driver Reaction Time in Seconds 90
6.3 Input Noise Distributions 103
6.4 True Input Distributions 104
6.5 Error of T_{lsb} Estimation Due to Sensor Noise in Seconds 104

Chapter 1

Introduction

The core of this thesis research work is dedicated to the application of new, emerging, and advanced formal engineering design synthesis methodologies towards the development of underlying technologies essential for implementation of intelligent vehicles, which aim to improve traffic safety as well as fluidity by actively assisting human drivers with collision prevention and crash mitigation, if unavoidable.

1.1 Intelligent Vehicles

An intelligent vehicle that assists driving must give warnings in dangerous situations, act automatically to avoid collisions in emergencies, and facilitate a smooth traffic flow. The basic goal is to improve traffic safety as well as fluidity by reducing collisions and collision impacts if unavoidable. The specific idea of intelligent vehicles considered here is to equip current vehicles with necessary sensory and control systems that provide the following functions in real time and noisy traffic environments:

- Monitor traffic and road conditions as well as driving behavior and host vehicle states;
- Identify and assess potential threats;
- Generate appropriate warning signals;
- Brake automatically to avoid collisions if necessary.

Satisfying these functional requirements in a dynamic and noisy traffic environment presents extremely complex system design and control problems for human engineering design researchers to solve with traditional engineering methods. Instead formal engineering design synthesis methodologies are presented here to automatically develop novel and diverse engineering design solutions to these complex problems.

1.1.1 Background

Tremendous progress has been made in the last few decades in terms of traffic safety. Particularly in the automotive industry, improvements in passive safety features such as safety belts, air bags, crumple zones, etc., and active safety features such as anti-lock braking systems (ABS), traction control systems (TCS), and electronic stability control (ESC) systems, etc., have dramatically helped to reduce the accident fatality rate. For instance, the fatality rate per 100 million vehicle miles traveled in the U.S. had gradually fallen from 5.50 in 1966 to 1.44 in 2004. However, the number of police-reported motor vehicle crashes occurring on U.S. highways was still nearly 6.2 million in 2004. They kill more than 42,000 people, injure approximately 3 million others, and cost more than $230 billion each year. On average, a police-reported motor vehicle crash occurs every 5 seconds, a person is injured every 11 seconds, and someone is killed every 12 minutes in the U.S. [NHTSA 2006].

Driver error is the leading cause of highway crashes [Campbell et al. 2003]. Also the popularity of in-vehicle devices such as cellular phones increases driver distraction and the injury crash likelihood by as many as *four* times [McEvoy et al. 2005]. The mission of intelligent vehicles is to help reduce the number and severity of these crashes.

1.1.2 Introduction

The term "Intelligent Vehicle", or IV, refers to a range of systems from driver assistance technologies to autonomous vehicles and advanced mobile robots, which usually integrate some type of information or control system into the existing vehicle to enhance its performance. Intelligent vehicle systems are defined as systems that sense the driving environment and provide information and/or vehicle control to assist the driver in optimal vehicle operation [Bishop 2005]. Intelligent vehicle systems operate at the "tactical level" of driving (throttle, braking, steering), as contrasted with strategic decisions such as route selection, etc. For instance, an intelligent vehicle can assist the driver directly with making decisions related to the driving task, even taking action required to improve overall safety.

Intelligent vehicle systems are seen as the next generation beyond current active safety systems, such as ABS, TCS, ESC, etc., which provide relatively basic control assist but do not sense the environment or assess risk. Intelligent vehicle systems offer the potential for significant enhancements in safety and operational efficiency. As one important component of intelligent transportation systems (ITS), intelligent vehicle systems use sensing and intelligent algorithms to understand the environment immediately around the vehicle and make critical threat assessments and judgments in real time, either assisting the driver in vehicle operations (driver assistance) or fully controlling the vehicle (automation). Generally, the intelligent vehicle application areas can be divided into three groups of systems that

- advise or warn the driver (e.g., collision warning, lane departure warning, etc.);
- partially control the vehicle, either for constant driver assistance (e.g., adaptive cruise control) or as an emergency intervention to avoid a collision (collision avoidance);
- fully control the vehicle (e.g., autonomous driving, vehicle platooning, etc.).

In this thesis, the first two groups of intelligent vehicle systems are considered, which are also the current focus of the automotive industry.

Safety has always been a priority for automotive companies and their engineers. Using advances in sensors and semiconductors, engineers are developing and testing on-board, knowledge-based technologies for improving driving safety to a new level. Some collision warning devices have already entered consumer markets. For example, the 2003 Jaguar features a forward alert system from Delphi that advises a driver to brake in the approach of slow-moving traffic ahead [Sharke 2003]. If the driver does not adequately respond to warnings, collision avoidance systems might take control of the throttle, brakes, or steering, to maneuver the vehicle back to a safe state. For instance, Nissan's new gas pedal will lift itself to alert the driver of a possible collision, and the brakes are automatically applied if the radar sensor detects a possible collision ahead and when the driver's foot is off the gas [Kageyama 2006]. Taking a step further, the 2006 Honda Acura RL will even actively brake the car if an imminent rear-end collision is sensed [Honda 2005]. The so-called Honda Collision Mitigation Braking System^{TM} (CMBS^{TM}) predicts potential collision situations with a millimeter-wave radar unit, which monitors the distance between the RL and objects in front of the car, as well as closing rates. If CMBS determines the closing rate between the RL and the vehicle directly in front has increased beyond an acceptable level, visual and audible warnings prompt the driver to take preventive action. If the distance further diminishes, the system provides a tactile warning by gently retracting the seatbelt and then applies light braking. If an accident is determined to be unavoidable, the system applies strong braking and strong retraction of the front seatbelt to reduce the speed of impact and to mitigate the damage of a collision.

1.2 Motivation

Design has traditionally been a creative process that requires human ingenuity and experience. In a modern engineering design process, highly complex design tasks such as the development of intelligent vehicle systems are characterized by severe reliability and robustness requirements, where each unit consists of an intelligent vehicle and a human being. The main challenges in designing and optimizing such complex distributed control systems include, but are not limited to, the following difficulties:

- high, or sometimes even *a priori* unknown, complexity of good design solutions;

- multiple objectives, competing factors, and trade-offs;
- simultaneous hardware and software optimization requirements;
- dynamic and stochastic evaluation results instead of static and deterministic ones.

All these problems make it difficult for an engineer, using traditional engineering methods, to synthesize an appropriate design solution under complex system design requirements such as a traffic system.

Until now, no traditional engineering methods have been available for meeting all of the challenges mentioned above. Alternatively, stochastic simulation methods such as Monte Carlo methods can be used to explore possible designs randomly. However, a random, undirected approach such as this will be computationally expensive, and provides no systematic exploration of more promising regions of the design space. On the other hand, biological systems serve as a great source of inspiration. The principal advantage of a biologically inspired approach is that such techniques have stood the tests of eons of competitions and evolutions. Not only are these techniques robust and more efficient than random search, they also have the advantages of being fairly scalable and applicable to distributed systems that might consist of various heterogeneous agents.

Formal engineering design synthesis methodologies [Antonsson and Cagan 2001, Lee 2002] reduce the reliance on human resources and shorten design cycles. They can be used to computationally synthesize novel designs and assist the human designers in the engineering design decision-making process with more knowledge and reduced uncertainties.

1.3 Thesis Contributions

In my research, I applied novel and formal engineering design synthesis methodologies to develop underlying technologies essential for intelligent vehicles.

First, I developed a formal engineering design synthesis methodology based on evolutionary computation especially for design and optimization of distributed control systems in an autonomous way. I validated the efficacy of the evolutionary design synthesis method through multiple engineering design case studies in the framework of developing sensor and control systems for intelligent vehicles. I have shown that the evolutionary design synthesis method is able to generate a variety of novel design solutions that can reflect different engineering design trade-offs selected by the human designer, and have achieved performances comparable to, if not better than, that of hand-coded solutions in the same simplified environment. More importantly, this automatic design synthesis method shows great potential to handle more complex design challenges such as those mentioned in Section 1.2, where a good hand-coded solution may be very difficult or even impossible to obtain. Moreover,

the evolutionary design synthesis methodology appears promising to deal with uncertainty in the problem efficiently and adapt to the collective task nature well.

In addition, I implemented multiple levels of vehicle simulation models with different computational cost and fidelity as well as necessary driver behaviors for different types of simulation experiments conducted for different research purposes. I have tried to get as much as possible out of limited computational resources, such that good candidate solutions can be generated more efficiently with less computational time and human engineering effort.

Furthermore, I reviewed and discussed different threat assessment measures and collision avoidance algorithms. I proposed a new threat assessment measure, time-to-last-second-braking (T_{lsb}), which directly characterizes human natural judgment of the urgency and severity of threats in terms of time. Based on driver reaction time experimental results, I proposed new warning and overriding criteria in terms of the new T_{lsb} measure, and analyzed the performance statistically in terms of two typical sample pre-crash traffic scenarios. Less affected by driver behavior variability, the new criteria characterize the current dynamic situations better than the previous ones, providing more appropriate warning and more effective overriding at the last moment. Finally, I explored the possibility of frontal collision avoidance through steering (lane-changing), proposed the time-to-last-second-steering (T_{lss}) measure and compared it with T_{lsb}.

1.4 Thesis Overview

Chapter 2 briefly reviews the evolutionary computation methods and presents a specific automatic design synthesis methodology based on evolutionary computation principles, with some special features introduced to deal with the design challenges mentioned in Section 1.2.

Chapter 3 first reviews the various existing vehicle simulation models and driver behavior models presented in the literature, then describes the multiple levels of vehicle simulation tools implemented for different applications considered in this thesis. Finally, different methods are applied to implement the driver behavior models used in the simulation, such as car-following, lane-keeping, and lane-changing behaviors.

Chapter 4 presents a first case study of developing a collective sensory system for intelligent vehicles, using the evolutionary design synthesis method introduced in Chapter 2. It is demonstrated that a full family of engineering design trade-off solutions can be generated efficiently using aggregated fuzzy fitness functions with different weights and trade-off strategies selected by the human designer to reflect different preferences on multiple performance measures.

Chapter 5 presents two case studies concerned with synthesizing novel neural network controllers for intelligent vehicles in two different application backgrounds, using the same evolutionary design synthesis method introduced in Chapter 2. It is shown that the performances of various evolved

neural network controllers are comparable to, if not better than, that of a hand-coded rule-based controller under the same conditions.

Chapter 6 reviews different threat assessment measures and collision avoidance algorithms presented in the literature and their limitations, and presents a new threat assessment measure and corresponding warning/overriding criteria, whose performance is analyzed and its advantages over previous ones are discussed. Moreover, the options of frontal collision avoidance through either steering (lane-changing) or braking are compared and discussed.

Chapter 7 concludes the thesis with a summary and discussion of the limitations and future research directions.

Chapter 2

Evolutionary Computation Design Methodologies

Evolution is ubiquitous in nature. Natural systems are undoubtedly the most remarkable known to humans. However, nature cannot be considered a designer in the traditional sense. Natural designs are rather a result, not a goal, of evolution.

In his seminal work, *On the Origin of Species*, Charles Darwin described the process of natural selection, or "survival of the fittest" evolution, and introduced the foundations for evolutionary algorithms (EA) [Darwin 1859]. An EA describes how any system may evolve over time through repeated actions of *transmission*, *variation*, and *selection* of individuals with different traits. For example, in biological evolution, individuals reproduce and transmit their genetic code to offspring with some variation, then the combined pool is subject to a selective environment where only the fittest individuals are left to repeat the above evolution process. In general, transmission ensures that good traits are passed on to future generations while variation enables the discovery of new, better traits. Selection guides evolution by eliminating unfit individuals with weak traits. Evidently nature's design synthesis algorithm, the EA, has been extremely successful in generating novel and complex designs.

Human designers have also been trying to use evolution to achieve their own specific goals. Unfortunately, the problem of evolving non-biological systems is not trivial, i.e., implementing reproduction, variation, and selection to achieve desired design goals effectively. In nature this problem was answered by Mendel in his brilliant work with peas [Mendel 1866], which led to the development of genetics. However, it was not until the 1950's that evolution and genetics were reconciled, and a theory of "evolutionary synthesis" emerged. Since then, studies on stochastic search and optimization techniques based on the biological principles of the natural evolution process have led to the development of evolutionary computation (EC) methodologies.

Natural evolution has been an inspiration for developing automatic design synthesis methods and computing the solutions to problems that have previously appeared intractable. EC methods

can often outperform conventional optimization methods when applied to challenging and complex real world problems. Although EC algorithms are good design and optimization methodologies, the actual implementation of EC's still requires special attention and engineering ingenuity for effective application in engineering design synthesis.

Recent research has demonstrated the ability of EC methods to successfully synthesize novel design configurations in various engineering design application domains [Bentley 1999, Lee 2002, Lipson and Pollack 2000, Nolfi and Floreano 2000]. However, many conventional EC applications such as design parameter optimization have assumed a fixed design architecture with a single well-defined design objective, which is represented by a deterministic fitness function. Therefore, standard EC methods have to be appropriately adjusted to deal with the current engineering design challenges mentioned in Section 1.2.

In this thesis, an evolutionary design synthesis methodology is introduced especially for designing and optimizing distributed embodied systems in an autonomous way [Antonsson et al. 2003, Zhang et al. 2003b, Martinoli et al. 2002]. This methodology shows several characteristics that appear promising for the distributed control system design challenges. First, it works off-line: Solutions are first evaluated in realistic simulations, preventing the test of unsafe solutions directly on real hardware, yet are realistic enough to be transported to real hardware. Second, it is platform-independent and system-oriented, i.e., it can be applied to different platforms with respective special system constraints. Third, it can deal with uncertainty in the problem efficiently and easily adapt to collective tasks. Finally, in comparison to traditional hand-coded design, the design solutions are automatically synthesized and the human engineering effort involved is minimized to the mathematical formulation of the desired performance and to the encoding of real problems in the search space of the stochastic exploration algorithm.

In the following sections, the EC literature is reviewed and the evolutionary engineering design synthesis method is presented, including special features introduced to face the modern engineering design challenges.

2.1 Evolutionary Computation

Since the 1960's, there has been an increasing interest in simulating the natural evolution process to solve optimization problems, leading to the development of evolutionary computation methods [Bäck et al. 2000, Bäck 1996]. The basic idea of EC is to make use of the powerful process of natural evolution as a problem-solving paradigm by simulating it in a laboratory or on a computer. The general approach is to have a pool of candidate solutions evaluated in parallel, from which the "fittest" solutions are chosen to reproduce new candidate solutions using stochastic genetic operators such as *recombination* and *mutation*. This procedure is iterated until the population converges

or a preset condition is met. Therefore all EC methods involve *reproduction*, *random variation*, *evaluation*, and *selection* of competing individual solutions in a population, which form the essence of evolution. Moreover, before an EC method can be applied to a real design or optimization problem, appropriate *encoding* of the design variables in the feasible search space needs to be determined by the designer. In addition, *initialization* and *termination* are also two indispensable steps for EC methods. Although simplistic from a biologist's viewpoint, these algorithms are sufficiently complex to provide robust and powerful adaptive search mechanisms.

Some advantages of EC methods include that they

- optimize with continuous, discontinuous, or discrete search space;
- do not require derivative information;
- can deal with a large, even variable, number of variables;
- simultaneously search a wide sampling of the search space;
- can deal with extremely complex fitness landscapes;
- provide multiple novel solutions, not just a single solution;
- are well suited for parallel computers.

These advantages are intriguing and produce stunning results when traditional optimization approaches such as gradient descent or direct, analytical discovery are impossible. Combinatoric and real-valued function optimizations which deal with "rugged" optimization surfaces or fitness landscapes, possessing many locally optimal solutions, are well suited for EC.

Today there are four mainstream representatives of evolutionary computation methods. They are genetic algorithms (GA), evolution strategies (ES), evolutionary programming (EP), and genetic programming (GP), which is a derivative of GA's.

2.1.1 Genetic Algorithms

Genetic algorithms were first developed by John H. Holland at the University of Michigan in the 1960's [Holland 1975] and were later popularized by his student David E. Goldberg [Goldberg 1989]. The work of De Jong [De Jong 1975] showed the usefulness of GA for function optimization and made the first effort to find optimized GA parameters. Today GA's are the most widely known type of EC algorithms, receiving attention all over the world.

Besides the general properties of EC described above, genetic algorithms emphasize recombination (crossover) as the most important genetic operator, and apply mutation with very small probability. Individuals are chosen for crossover probabilistically: Each individual is assigned a

probability proportional to its observed performance (fitness value). Thus better individuals are given more opportunities to produce offspring (i.e., reproduction with emphasis). GA's usually maintain a fixed-sized population, and each new population is created by taking all the children generated and selecting from the old population for the rest if needed. The selection here could be stochastically or deterministically biased toward better individuals or be randomly unbiased. Finally GA's often use a binary representation, while other representations, such as real-valued parameters, are also used.

2.1.2 Evolution Strategies

Evolution strategies were first developed jointly by Peter Bienert, Ingo Rechenberg, and Hans-Paul Schwefel in Berlin in 1964 [Rechenberg 1965].

Evolution strategies use normally distributed mutations to modify real-valued vectors and emphasize mutation and recombination as essential operators for searching both in the search space and in the strategy parameter space at the same time. Hence the self-adaption of strategy parameters has also been implemented in the evolution process. The selection operator is deterministic, and parent and offspring population sizes usually differ from each other.

The general frame of evolution strategies can be easily presented by the symbolic notation described in [Schwefel 1977]. The abbreviation $(\mu + \lambda)$ ES denotes an ES that generates λ offspring from μ parents and selects the μ best individuals from the $\mu + \lambda$ individuals (parents and offspring) in total, where $1 \leq \mu \leq \lambda < \infty$. For instance, the simple ES can be expressed by (1+1) ES. In contrast, the abbreviation (μ, λ) ES denotes an ES that generates λ offspring from μ parents but selects the μ best individuals only from the λ offspring, where $1 \leq \mu < \lambda < \infty$.

2.1.3 Evolutionary Programming

Evolutionary programming was first introduced by Lawrence J. Fogel in San Diego, California, in 1960 [Fogel et al. 1966], and was extended and popularized by his son David B. Fogel in the late 1980's [Fogel 1992].

Evolutionary programming emphasizes mutation and does not use recombination at all. Similar to ES, EP also works with normally distributed mutations and extends the evolution process to the strategy parameters. The selection operator is probabilistic or deterministic. Unlike GA's, which typically involve encoding the problem solutions as binary strings, the representation of EP directly follows from the problem. For instance, a neural network can be represented in the same manner as it is implemented, e.g., a vector of its real-valued weights.

2.1.4 Genetic Programming

Genetic programming was invented by John R. Koza in the 1990's [Koza 1992, Koza 1994].

Genetic programming applies the evolutionary search principle to automatically develop computer programs in suitable languages, e.g., LISP. The data structures that undergo adaption in GP are executable computer programs, which are usually represented by tree structures of variable size. GP uses both recombination and mutation operators, which need to be especially designed for the data structure used. Fitness evaluation in GP involves executing these evolved programs and GP searches in the space of possible computer programs for ones that produce the best fitness.

2.2 Evolutionary Computation Theory

As evolutionary computation methods become more and more widely used for practical problem solving, increasing emphasis is placed on understanding the theories behind them.

The traditional theory of GA's [Holland 1975, Goldberg 1989] assumes that GA's work by discovering, emphasizing, and recombining good "building blocks" of solutions. In other words, good solutions tend to be made up of good building blocks, or *schemas*. Hence steps should be taken to ensure that better solutions reproduce more offspring to promote their good genes. These basic notions are powerful, important, and a key to understanding and implementing the GA's better [Goldberg 2002].

Modern theories consider EC as a Markov chain process [De Jong et al. 1995]. Possible population configurations during the evolution correspond to the states in the Markov chain process, and state transition probabilities then depend on the genetic and selection operators chosen. This relationship is often too complex to be characterized. In addition, the transition matrix rapidly grows intractable when the number of states increases.

It follows that the effectiveness of EC implementations cannot generally be predicted in advance, hence the need for empirical implementation and computer experimentation is justified. In addition, according to the No Free Lunch (NFL) theorem [Wolpert and Macready 1995], no single optimization algorithm is the best for solving all possible (optimization) problems. Two points can be derived from the NFL theorem. First, EC should not be blindly applied to any problem. Second, each particular EC algorithm design and implementation should be carefully customized for the specific problem. In other words, GA, ES, EP, or any derivative EC method is good at solving a certain class of problems, and they are not competitive for the classes of problems that have been solved using traditional optimization methods such as gradient descent. This implies that EC should be applied when other traditional methods have failed or simply do not apply, e.g., when discontinuous, nondifferentiable, multi-modal, noisy, and/or unconventional response surfaces are involved. The effectiveness and robustness of EC thus extend to a broader field of applications beyond the simple

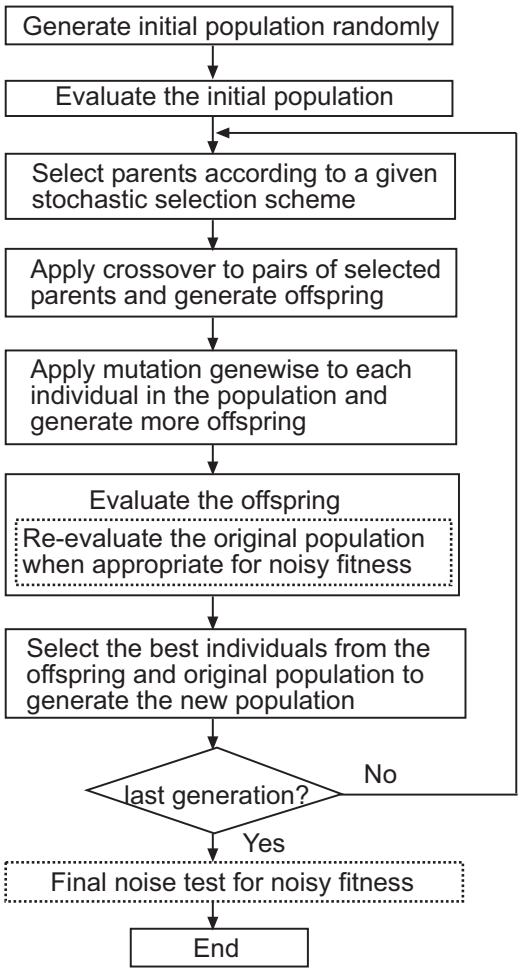


Figure 2.1: The Evolutionary Optimization Loop Used in the Automatic Engineering Design Synthesis Process

classes of problems solved by classical methods.

2.3 General Structure

In this thesis, an automatic engineering design synthesis methodology based on evolutionary computation is introduced and its general structure is presented below, along with its special features to deal with the current engineering design challenges [Antonsson et al. 2003, Zhang et al. 2003b, Martinoli et al. 2002].

Based on evolutionary computation methods reviewed in Section 2.1, the evolutionary optimization loop used is shown in Figure 2.1. First, an initial population of solutions is generated randomly. Then, each individual is evaluated under a performance test in terms of the specific design problem for one *evaluation span*. According to the evaluation results, i.e., the fitness value of each

individual, the *parent selection* scheme chooses pairs of parent solutions for crossover, promoting individuals with higher fitness. Crossover between the selected pairs of parents is conducted under certain crossover probability to generate pairs of offspring. Mutation is also applied to each gene of the original pool under certain mutation probability and generates more offspring. If the fitness evaluation function is deterministic, then only the offspring (from both crossover and mutation) are evaluated, otherwise the original parent population may also be *re-evaluated* to get a better estimate of their true fitness values. The best individuals are then deterministically selected from both the original population and the offspring, i.e., elitist *generation selection*, to constitute the next generation. Hence an offspring will only replace an individual of the original population if it has a higher fitness, conforming to the $(\mu + \lambda)$-selection scheme in ES, which ensures that the mean of the population fitness is generally non-decreasing[1] over generations. At the end of each generational loop the program verifies whether or not another generation is needed in order to meet a preestablished criterion for terminating the evolutionary run.

This evolutionary design synthesis methodology is especially developed to deal with the engineering design challenges mentioned in Section 1.2. First, the encoding allows variable-length chromosomes, making it possible to evolve design solutions of suitable complexity (e.g., an appropriate number of design parameters) and optimize parameter values simultaneously. In this case, the initial pool is randomly generated with solutions of diverse complexity. The crossover and mutation operators have to be adjusted from the standard ones to conform to the variable-length chromosome encoding, which will be explained in detail in Section 2.4.3 and 2.4.4.

Second, multiple competing design objectives can be expressed as *preferences* using fuzzy sets [Otto and Antonsson 1991, Scott and Antonsson 1998]: Each value of a design objective or performance indicator is assigned a preference value between 0 (totally unacceptable) and 1 (completely acceptable). Each objective may have a different level of importance, or *weight*. The fuzzy preferences can then be aggregated into the fitness function with different weights and compensation strategies, which can be tuned to evolve solutions with different engineering design trade-offs [Antonsson et al. 2003]. Simultaneous hardware and software optimization could also be addressed by co-evolution of the system morphology and controller [Bugajska and Schultz 2000, Lipson and Pollack 2000], which appears to be more promising than evolving the morphology or controller alone, but is not addressed in this thesis.

Third, as stochastic optimization methods, evolutionary algorithms are good at working in noisy environments and searching for robust solutions, in contrast to traditional optimization methods, which strongly rely on deterministic information to find optimal solutions. When the evaluation process and result are non-deterministic, i.e., dynamic and stochastic, as characterized by real traf-

[1] For the case of non-deterministic fitness function, the individual fitness might decrease after re-evaluation adjustment, which in turn might cause the mean of the population fitness to decrease in some rare case.

fic scenarios investigated in this thesis, multiple re-evaluations of the surviving individuals may be introduced when appropriate. Generally an aggregated fitness value from multiple evaluation results would give a better estimate of an individual's true fitness than a single evaluation. However, multiple evaluations of each individual would certainly increase the computational time, which is why only the surviving individuals would be considered for a possible re-evaluation, since more computational power is reserved for more promising solutions that have survived over multiple generations. But there is still a trade-off between the accuracy of the fitness estimate and computational time during the evolution process. In general, the more expensive the evaluation test is than the genetic operations, the smaller the number of evaluations that should be used [Fitzpatrick and Grefenstette 1988].

Finally, a fair final noise test consisting of 100 evaluation spans is performed on all distinct individuals in the final population in the noisy environment in order to assess the "best" and the most robust design solution as specified by the aggregation criterion according to design requirements.

2.4 Implementation in Design Synthesis

In this section, the implementation of the evolutionary design synthesis methodology is presented in detail. In general, real-valued vector representation is used with variable vector size, and both crossover and mutation operators are used to modify candidate solutions. Traditional fitness proportional (roulette wheel) selection with fitness scaling is used for parent selection, while elitist generation selection is chosen.

2.4.1 Encoding

Like ES and EP, the design parameters are directly encoded as real-valued vectors during the evolutionary synthesis. When the appropriate number of design parameters is *a priori* unknown, the sizes of the vectors in the evolutionary pool are also variable to allow design solutions of suitable complexity to be evolved [Lee and Antonsson 2000]. For instance, a neural network can be encoded as a vector of its real-valued weights, with variable sizes representing neural networks of different complexity and topologies.

The crossover and mutation operators must be carefully designed to make them meaningful for the variable-length real vector encoding, which will be explained in detail in Section 2.4.3 and 2.4.4.

2.4.2 Initialization

Generally, the initial population of design candidate solutions are randomly generated within the feasible search space. When variable size encoding is used, the initial population of individuals is randomly generated with candidate solutions of various complexity. For instance, when the topology

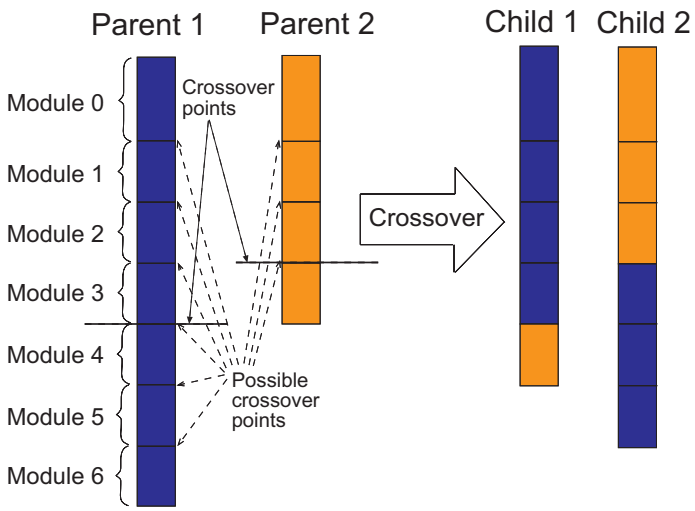


Figure 2.2: Illustration of One-point Crossover Scheme for Two Vectors of Different Lengths

of neural networks is also evolved, an initial pool of neural networks is generated with different topologies and weights.

2.4.3 Crossover

Standard crossover operators such as one-point or uniform crossover can be directly applied to two real vectors of equal length. When the vector size is variable, the crossover operator needs to operate on two vectors of different lengths.

A simple and efficient solution [Lee and Antonsson 2000] is to identify each parameter in a vector with an index value chosen from a preset range, from which the crossover point is randomly chosen and divides the index range into two sub-ranges. Then the two parent vectors swap their parameters of the same index sub-range to generate two children.

However, to protect possible modules in the candidate solutions, crossover points should not be arbitrarily chosen along the whole vector. As shown in Figure 2.2, the elements in the vector can be grouped into modules or blocks according to each specific underlying design problem. For example, the parameters of each sensor can be considered a module of a collective sensory system. Each module can be identified with an index value, and sequenced accordingly as shown in Figure 2.2. The new one-point crossover operator proposed here only allows interchange of modules between the parents. From all the possible crossover points, shown by dashed arrow lines in Figure 2.2, a random crossover point is selected for both parents and they exchange their modules below the crossover point to create two children. For the example shown in Figure 2.2, two parents of 7 and 4 modules produce two children of 5 and 6 modules, respectively.

2.4.4 Mutation

Mutation is also a powerful tool for creating new design solutions. Gaussian mutation is always applied to change values of the design parameters with a certain mutation probability for each parameter. When the complexity of design solution is variable, insertion and deletion are also used as mutation operators, in addition to the standard mutations that only change the parameter values, to change the number of design parameters, i.e., directly insert or remove a design module. For example, a sensor module can be added into or removed from a collective sensory system to generate a new sensory system design. For another example, a hidden neuron can also be added into or removed from a neural network design to create a new one.

2.4.5 Evaluation and Re-evaluation

The principle of evolutionary computation is "survival of the fittest". Evaluation is used to determine the fitness or performance of an individual design solution, i.e., to assess how "good" the solution is. Then the reproduction and selection schemes will be able to bias toward those "fitter" individuals with higher fitness values. Evaluation is problem-dependent and it is up to the human designer to decide how the candidate solutions should be evaluated to represent the specific design goals.

As mentioned above, the evaluation process and result for a given design solution could be intrinsically dynamic and stochastic instead of static and deterministic. In this case it is desirable to *re-evaluate* each candidate solution multiple times to get a better estimate of its real fitness, since a single evaluation could be deceptive due to noise. However, as mentioned in Section 2.3, a trade-off between the computational time cost and the accuracy of the true fitness estimate has to be made according to each specific design problem.

2.4.6 Selection

Selection involves both selection for reproduction (parent selection) and selection for survival (generation selection). A variety of selection schemes have been proposed, including both unbiased and biased selection (toward "fitter" individuals), which could be either deterministic or stochastic.

The most common selection operators are the *roulette wheel* (proportional), *rank*, and *elitist* selection [Mitchell 1996]. In the roulette wheel selection, the probability that an individual is selected is proportional to its *scaled* fitness value, promoting fitter individuals with higher fitness values. The original (raw) fitness values are usually scaled to avoid the problem of excessive or insufficient selection pressure due to disproportional fitness values. Rank selection, as the name suggests, ranks the solutions according to their fitness values and then probabilistically selects individuals according to their ranks. Finally, the elitist selection deterministically selects a certain number of the fittest individuals of the population; such individuals could be lost otherwise.

In the evolutionary design synthesis process shown in Figure 2.1 proposed in Section 2.3, the roulette wheel selection is used with a scaling factor 2 for parent selection, and the elitist selection is used for generation selection.

2.4.7 Termination

Usually the evolution is terminated when a preset number of generations is reached, or the desired design goal is met before that. The maximum number of generations is carefully selected to try to ensure that the evolution process has found the best design solutions it could for the particular evolutionary run. The same evolutionary process is also repeated multiple times to generate multiple (usually different) design solutions and avoid being unilateral by one particular run.

2.4.8 Final Evaluation

As shown in Figure 2.1 and mentioned in Section 2.3, final evaluation is only needed when the evaluation process is stochastic with noisy fitness results. In this case a final noise test is conducted on all distinct individuals in the final population in order to assess the truly "best" design solution of each evolutionary run. Each individual is subject to 100 repeated evaluations, from which the 100 fitness samples obtained are aggregated to the individual's overall fitness value according to the aggregation criterion selected by the designer. For instance, the worst fitness value could be taken as the overall fitness when robustness of solutions is emphasized. Finally the individual with the best overall fitness is selected as the final result of the evolutionary design synthesis.

Chapter 3

Multi-level Simulation and Modeling of Intelligent Vehicles

The development of advanced intelligent vehicle technologies that improve traffic safety demands multiple levels of dynamical vehicle models and human driving behavior models. These models can serve as useful tools in analytical investigations and simulations of the effects of the proposed sensor and control systems. The simulation time complexity usually increases as the model complexity and accuracy increase, hence different trade-offs between the two factors have to be made under different situations and requirements.

In general, traffic simulations can be divided into microscopic and macroscopic simulations. The macroscopic traffic flow simulations [Gartner et al. 1997] are usually concerned with global characteristics of the overall traffic flow, such as the average vehicle speed and the traffic flow density, where individual vehicle behaviors are usually not modeled. Most macroscopic models are based on the continuity equation and related to particle physics and gas kinetics [Helbing et al. 2001]. Since intelligent vehicles are considered in this thesis, all the vehicle models discussed here fall into the microscopic traffic simulation category.

Vehicle simulation models can be divided into several categories. The simplest vehicle model is the point model, where vehicles are only represented by moving points without any details of the vehicle. The embodied vehicle simulation models characterize different levels of details of the vehicle model, which usually contains a three-dimensional (3D) representation of specific vehicle modules, such as vehicle body and wheels. The kinematic embodied vehicle simulation model only describes some kinematic characteristics of the simulation, such as the vehicle position, speed, wheel speed, etc. While the dynamic embodied vehicle simulation can simulate certain dynamic scenarios, such as applying a driving/braking torque, friction force and wheel slip, body roll and pitch, or wind effects, etc. In addition, noise can be added to all different levels of vehicle simulation to simulate actuator and environmental noise effects.

On the other hand, driver models also play an important role in microscopic traffic simulation. In

fact, most traffic simulation systems are focused on driver behavior modeling, such as car-following, lane-keeping, and lane-changing, etc. Different methods are applied to try to obtain realistic human driving behavior models.

3.1 Literature Review

Various levels of simulation of traffic systems with different vehicle models and driver models have been published previously. Each simulation model has its own characteristics and specific target application background. Conversely, different models have been developed for various design requirements and research purposes. An overview of the prior work is given below.

An overview of simulation of traffic systems is given in [Pursula 1999]. Macroscopic traffic models [Helbing et al. 2001] describe the collective vehicle dynamics in terms of the spatial vehicle density and the average velocity as a function of the freeway location and time, while microscopic traffic models delineate the positions and velocities of all interacting vehicles. It is shown that a link between microscopic and macroscopic traffic models can be established with good agreement on collective vehicle dynamics [Helbing et al. 2002]. Hence the macroscopic properties of the traffic systems must be considered when developing realistic microscopic traffic models to ensure good transportability.

Vehicle and driver behavior modeling is the core of the microscopic traffic models. In the literature, there have been many research efforts devoted to the development of vehicle models and driver models for various purposes. Traditionally, vehicle and driver behavior modeling is classified into two major types of models, which are concerned with longitudinal and lateral motions of the vehicle, respectively. The longitudinal vehicle control models are concerned with the vehicle's longitudinal dynamics, while the lateral vehicle control models relate to the vehicle steering behaviors.

The vehicle models developed in the literature are mainly based on physical equations and some specific car experimental data. As a result, these models are quite complex and detail-oriented. For example, a longitudinal dynamics model of an automotive powertrain system [Hedrick et al. 1993, Cho and Hedrick 1989] involves 12 state variables: four for the engine, two for the transmission, and six for the drivetrain, plus two time delays associated with the engine. The relationships involving the state variables are tested under certain experimental conditions, and then approximated by curve-fit models.

The longitudinal driver behavior models are mainly focused on the development and analysis of car-following models, which act as the link between microscopic and macroscopic models [Gartner et al. 1997, Helbing et al. 2002]. The class of suitable basic models proposed in the literature is characterized by continuous acceleration functions depending on the velocity, the gap, and the relative velocity with respect to the preceding car, etc. Some examples include the well-

known Gipps model [Gipps 1981], the optimal velocity model (OVM) [Bando et al. 1995], the intelligent driver model (IDM) [Treiber et al. 2000], the velocity difference model [Jiang et al. 2001], the bounded rational driver model [Lubashevsky et al. 2003], and the human driver (meta-)model (HDM) [Treiber et al. 2006].

On the other hand, driver and vehicle steering system models date back to the 1950's, when the vehicle dynamics were readily characterized by differential equations of motion [Segel 1956]. Then, the driver steering control behavior was modeled by a closed-loop feedback control system for lane position regulation with a feed-forward term to anticipate road changes [McRuer et al. 1977]. Following a series of human operator models, it was suggested that driver steering control strategy during path-following can be modeled by a time-lagged optimal preview control process with driver delay and preview time parameters [MacAdam 1981]. A more complex driver steering control theoretic model was then presented [Hess and Modjtahedzadeh 1990, Modjtahedzadeh and Hess 1993] with both high and low frequency compensation modules as well as a preview module. More recent work also tried to model the driver steering control model uncertainty [Chen and Ulsoy 2001].

Finally, a comprehensive discussion of human driver modeling involving both longitudinal and lateral control models was summarized in [MacAdam 2003]. And an integrated driver model with both longitudinal and lateral motion controls was developed based on the preview-follower theory [Guo et al. 2004].

Every model mentioned above has its own specific application background and study purposes that it was developed for; subsequently different trade-off of model complexity and precision was selected for each model. Most models in the literature are quite complex and specific with parameters to be tuned for different vehicles. In this thesis, a more generic class of vehicle and driver models is desirable, therefore multiple levels of simulation models with different trade-offs of model complexity and reasonable simulation time are implemented, as described in the following sections.

3.2 Vehicle Simulation

Three different levels of vehicle simulation models are implemented for different research application requirements. The principle here is to keep the models as simple as possible with only the necessary characteristics.

The simplest vehicle models are point models, where each vehicle is simply represented by a moving point and all details of vehicle dynamics are ignored. The advantages of point models are their simplicity and fast simulation. No specific simulator is needed and basic numerical analysis can be performed efficiently. However, the disadvantage is that they only simulate ideal situations and are not easy to view graphically. To make point simulations more realistic, probabilistic models are introduced to the point models, where the motions of the points can be modeled with certain

probabilistic distributions.

To have an animated graphic interface of the simulation, a kinematic embodied simulation is introduced, where each vehicle consists of several modules, such as the vehicle body and wheels as well as on-board sensors, with certain shape defined for each module, and certain relative position and simple joints (motion constraints) defined between modules. The simulation could run either with or without noise, i.e., under ideal situations.

To simulate more realistic vehicle dynamics effects, a dynamic embodied simulation is implemented, where customized physical properties of each module can be defined and more complex joints are introduced. The simulation time complexity increases as additional features are introduced into the simulation, and the appropriate level of model complexity necessary and sufficient for a certain research purpose must be carefully chosen.

Each of these three different vehicle simulation types is described in detail below.

3.2.1 Point Models

Point models are by far the simplest vehicle models. Each vehicle is only represented by a point moving in a two-dimensional (2D) surface space. Each point has its own position, velocity, and acceleration without any other characteristics. A point's trajectory and velocity time history can be fully determined using only particle kinematics, given its acceleration history profile and initial position/velocity. The acceleration of a point is defined by the driver model, which will be discussed in Section 3.3, and may be related to the point's goal and relative position on the road as well as its position and velocity relative to other moving points.

3.2.1.1 Theoretic Models

Under theoretic conditions, the point movements are governed by fundamental kinematic equations, as shown below:

$$v(t) = v_0 + \int_0^t a(t)\,\mathrm{d}t \tag{3.1}$$

$$R(t) = R_0 + \int_0^t v(t)\,\mathrm{d}t \tag{3.2}$$

Here $a(t)$, $v(t)$, and $R(t)$ are the point's acceleration, velocity, and position time histories, respectively, with v_0 and R_0 representing its initial velocity and position, respectively. Hence the $v(t)$ and $R(t)$ of a point can be computed from the above equations given its $a(t)$, v_0, and R_0.

On the other hand, given the distance between two points, their relative velocity and acceleration, it is easy to compute when they would collide assuming certain acceleration histories $a_1(t)$ and $a_2(t)$, and when the following vehicle needs to brake to avoid a potential collision. The collision is defined

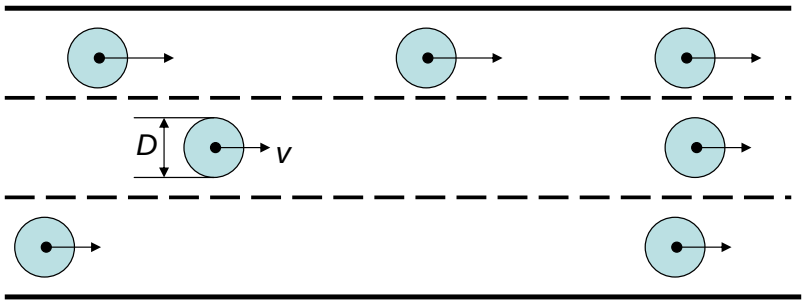


Figure 3.1: Sample Point Vehicle Models with Speed v and Collision Disk with Diameter D Moving on a Three-lane Highway

here when two points are within a certain distance D from each other. In other words, the vehicle can be considered to have a safety disk with center at the point and diameter D, as shown in Figure 3.1.

It can be noted here that even with a simple theoretic vehicle model, the driver model part that decides the acceleration profile $a(t)$ could be complex and highly stochastic in nature, which makes the whole simulation and subsequent analysis a non-trivial task.

3.2.1.2 Probabilistic Models

Variations exist in the real vehicles in that the vehicle response could be different under various conditions. For example, the engine performance could be different under different temperatures and air pressures; the transmission and drivetrain could run differently with different levels and status of the transmission fluid and lubrication oil; the brakes could behave differently under different temperature, humidity, cleanness and road surface conditions, etc. All these factors bring uncertainty to the vehicle dynamical response and should be considered in the vehicle models.

In the point model, a random term could be added into the deterministic vehicle kinematic Equation 3.1 and 3.2 to model the uncertainty of vehicle dynamics. When vehicle characteristics (e.g., vehicle speed) are measured by on-board sensors, there also exists measurement noise, which could be simulated with an additional random term. The noise probability distributions of different parameters are usually assumed to be independent of each other, and can be estimated from specific noise analysis experimental data [Brunson et al. 2002].

3.2.2 Kinematic Embodied Simulation

When more concrete models with good graphic display are desired, a kinematic embodied simulator such as Webots[1] [Michel 2004] could be used. As shown in Figure 3.2, sample traffic scenarios can be simulated in Webots, a sensor-based, 3D, embodied mobile robotics simulation software, where kinematic vehicle models with simple driver behaviors are moving on a simulated three-lane highway.

[1]Refer to `http://www.cyberbotics.com`.

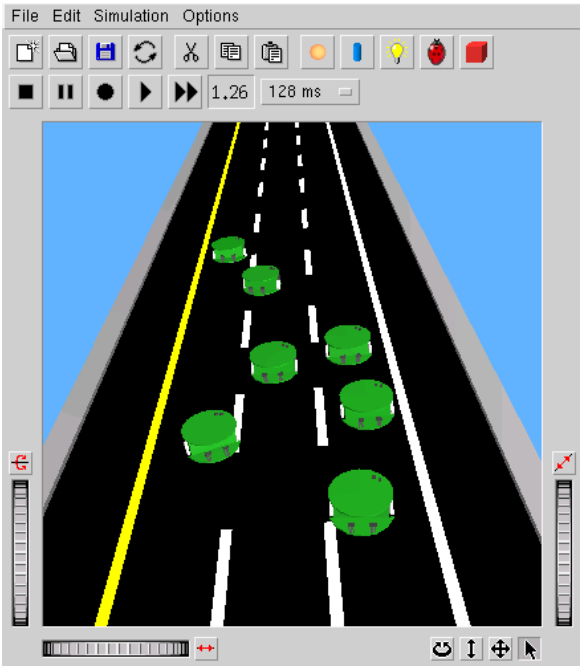


Figure 3.2: Screen Shot of Traffic Simulation in Kinematic Embodied Simulator—Webots

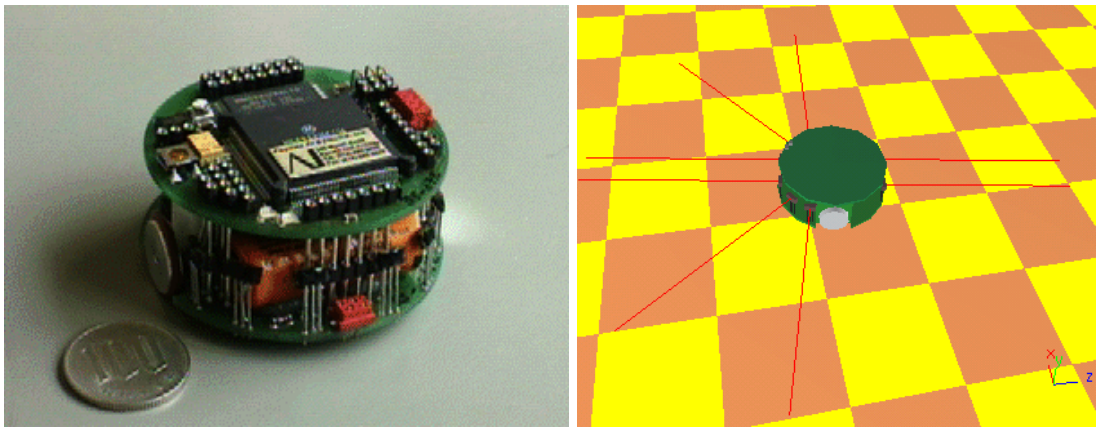
The vehicle model here is based on the Khepera[2] robot [Mondada et al. 1994], which is a miniature unicycle vehicle with a cylindrical shape measuring 5.5 cm in diameter and 3 cm in height. It is equipped with two motor wheels and eight infrared proximity sensors, as shown in Figure 3.3 (a).

The kinematic vehicle simulation model implemented in Webots contains the vehicle body and its two wheels as well as eight sensors, as shown in Figure 3.3 (b). The wheels can spin at different rotational speeds in either direction independently relative to the body, which is the only relative motion allowed between different vehicle modules, and enables the body to move freely on a 2D plane. Therefore this vehicle model has five degrees of freedom (DOF) in total, where the vehicle body has the three planar DOF (i.e., two translational and one rotational) plus two additional DOF for wheel spinning.

Each vehicle is controlled by a customized C or C++ controller program and there could also be a supervisor program managing the higher level simulation as a whole. The vehicle movement is simply controlled by specifying the left and right wheel speeds in the controller at each simulation time step. The controller can read sensor measurement values of light or distance, which can be used to compute appropriate left and right wheel speeds in order to implement the desirable robot behavior. The sensor measurements and wheel motor outputs can be simulated with realistic noise values.

Simple driver behaviors can be implemented for each vehicle model with different parameters, as will be described later in Section 3.3. Assuming the drivers are alert and react responsively, ideal

[2]Refer to `http://www.k-team.com`.



(a) (b)

Figure 3.3: Close-up of the Real Khepera Robot (a) and its Simulation Model in Webots (b) with its Distance Sensor Rays Illustrated in Solid Red Lines

sample traffic scenarios can be simulated as shown in Figure 3.2.

From this kinematic traffic simulation, the relative distances and approaching angles of all other vehicles that have appeared around the host vehicle can be recorded at each time step and accumulate to the *vehicle occurrence data*. Then a 2D vehicle occurrence probability density function (PDF), as shown in Figure 3.4, can be generated from the normalized vehicle occurrence data collected in the kinematic embodied simulation for a long enough period of time. This PDF is reflective of the accumulation of vehicle occurrences for 5000 test spans, where each test span contains 2000 simulation time steps, representing 128 seconds in real time. In addition, each vehicle is initialized with a random position and preferred speed for each test span. The number of test spans used to generate the PDF is judged to be sufficient to capture the 2D spatial distribution of vehicle occurrence probability, while averaging the temporal variations in traffic conditions.

Note that the x and y axes are fixed to the host vehicle and all other vehicles' positions are recorded relative to the host vehicle. Therefore the host vehicle is fixed to the origin of the 2D PDF, facing the positive y axis direction. The two peaks in the front and back of the host vehicle imply that the vehicles follow each other quite closely under the current car-following driver behavior. In addition, the dimensions of the 2D PDF shown are scaled to real lane width at 3.5 meters, hence the vehicles on the left and right adjacent lanes correspond to the lower peaks at $x = \pm 3.5\,\mathrm{m}$, respectively.

When only the relative approaching angle of the other vehicles is considered, the above 2D PDF can be further simplified to a 1D vehicle occurrence PDF, which can be plotted in either the Cartesian coordinates or the polar coordinates, as shown in Figure 3.5.

These vehicle occurrence PDF's capture basic characteristics of the dynamic traffic scenarios around the host vehicle more efficiently than the kinematic embodied simulation itself. For instance,

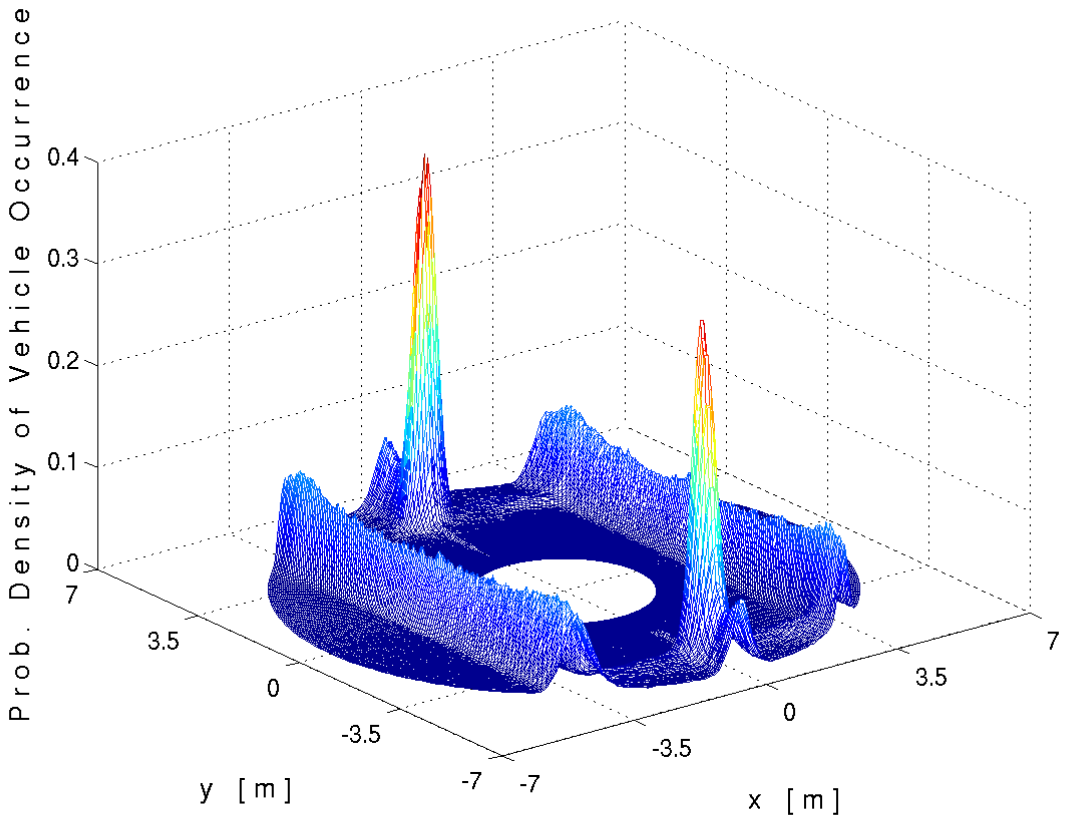
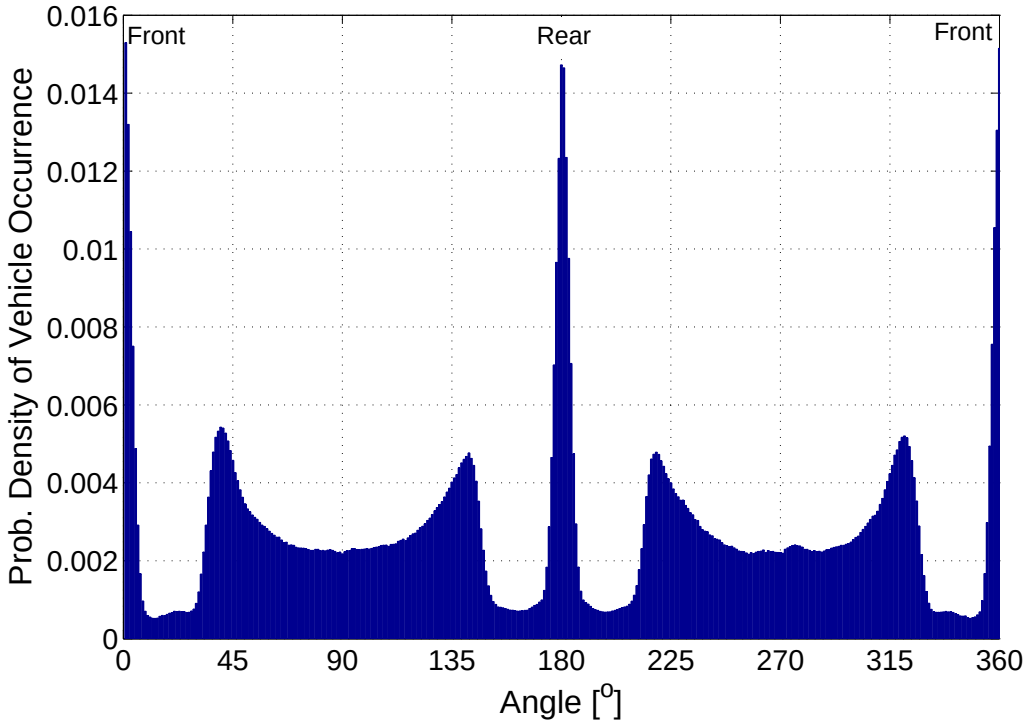
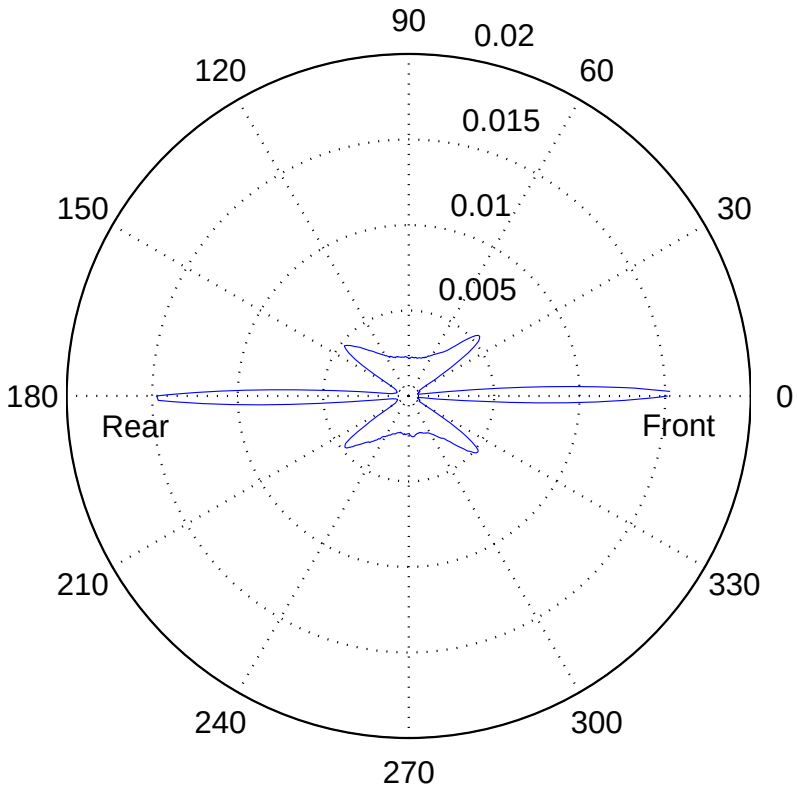


Figure 3.4: 2D Vehicle Occurrence PDF: The Host Vehicle Sits at the Origin Facing the Positive y Axis.



(a)



(b)

Figure 3.5: 1D Vehicle Occurrence PDF in Cartesian (a) and Polar (b) Coordinates

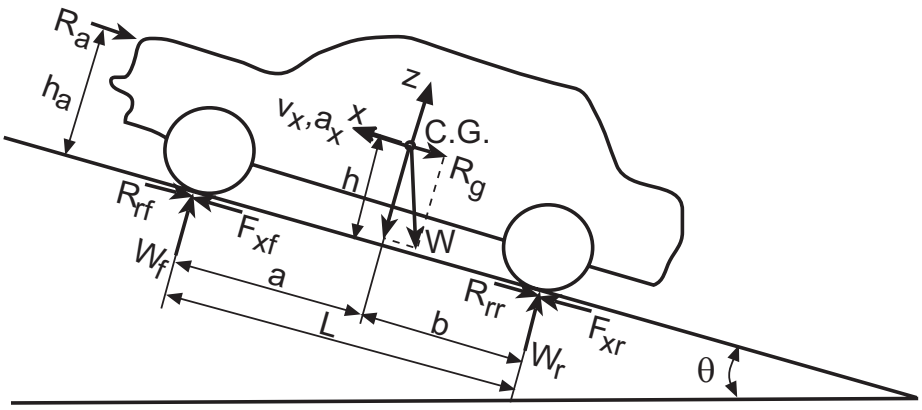


Figure 3.6: Significant Forces Acting on a Two-axle Vehicle

the vehicle occurrence probability density reflects the degree of priority for each location/direction to be covered by an on-board sensory system, i.e., when complete coverage is not achievable, some regions with higher vehicle occurrence probability should get higher priority to be covered first.[3]

More generally, the same strategy could be applied to capture some of the fundamental characteristics of the real traffic scenarios on the highway. One can imagine that the PDF would look quite differently at different times and/or places with different traffic rules and driving styles that form the driving behaviors ensemble, e.g., Rome or Paris vs. New York or Los Angeles. It would be very interesting to investigate the differences among different areas and design appropriate sensor and control systems accordingly, and perhaps to draw conclusions about the differences in driving habits and tendencies.

3.2.3 Dynamic Embodied Simulation

To make the model more realistic and relevant for the automotive industry, some essential features of vehicle dynamics are desirable for the vehicle model. It is also desirable to have a vehicle model that is based on real cars instead of a desktop robot. The principle used here is to keep the vehicle model as simple as possible while satisfying the above requirements.

3.2.3.1 Vehicle Dynamics Overview

First, important vehicle dynamics features are briefly reviewed in this section. Vehicle dynamics [Wong 2001, Zuvich 2000] is concerned with the movement of vehicles, which includes longitudinal (acceleration or braking), lateral (turning), and vertical (vibration) movements.

The major external forces acting on a two-axle vehicle are shown in Figure 3.6. W is the weight of the vehicle acting at its center of gravity (C.G.), and its normal component ($W\cos\theta$) along the

[3] In addition to vehicle occurrences, vehicle velocities and estimated threat level might also be important factors to consider.

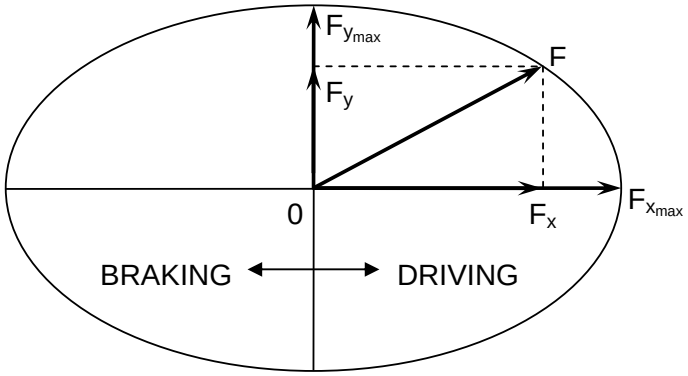


Figure 3.7: The Friction Ellipse Concept Relating the Maximum Cornering Force F_y to a Given Longitudinal Force F_x

vertical z axis is balanced by the normal supporting forces provided by the road surface through front and rear tires W_f ($\frac{b}{L}W\cos\theta$) and W_r ($\frac{a}{L}W\cos\theta$), respectively.[4]

The forces in the longitudinal direction (x axis) include the aerodynamic resistance R_a, rolling resistance from the front and rear tires R_r ($R_r \equiv R_{rf} + R_{rr}$), grade resistance R_g ($W\sin\theta$), and tractive (or braking) forces F_x ($F_x \equiv F_{xf} + F_{xr}$). The equation of motion along the longitudinal x axis of the vehicle is

$$ma_x = F_x - R_a - R_r - R_g \tag{3.3}$$

where m is equivalent mass of the vehicle and a_x is the linear acceleration or deceleration along the x axis. The maximum acceleration (deceleration) achievable by the vehicle is limited by the maximum tractive (braking) forces (F_x) available from the tire-road contact:

$$\max|F_f| = \mu \cdot W_f = \mu \cdot \frac{b}{L}W\cos\theta \tag{3.4}$$

$$\max|F_r| = \mu \cdot W_r = \mu \cdot \frac{a}{L}W\cos\theta \tag{3.5}$$

where μ is the friction coefficient between the tire and the road, which depends on many factors, including load, velocity, road surface conditions, tire pressure, etc. Refer to Section 3.2.3.4 for more details. F_x is the x component of the total friction force F ($F \equiv F_f + F_r$) available from both the front and rear wheels, which is limited by the friction ellipse, as shown in Figure 3.7. Note that there are no tractive forces available from the front tires ($F_{xf} \leq 0$) for a rear-wheel drive vehicle, and similarly $F_{xr} \leq 0$ for a front-wheel drive vehicle.

The response of automobiles to steering control [Segel 1956, Whitcomb and Milliken 1956] is quite complex. When the wheels are steered at some angle δ, they develop lateral (cornering) forces F_y that turn (yaw) the vehicle, and lateral slip angle α, which is the angle between the wheel plane

[4]Dynamic load transfer is ignored here.

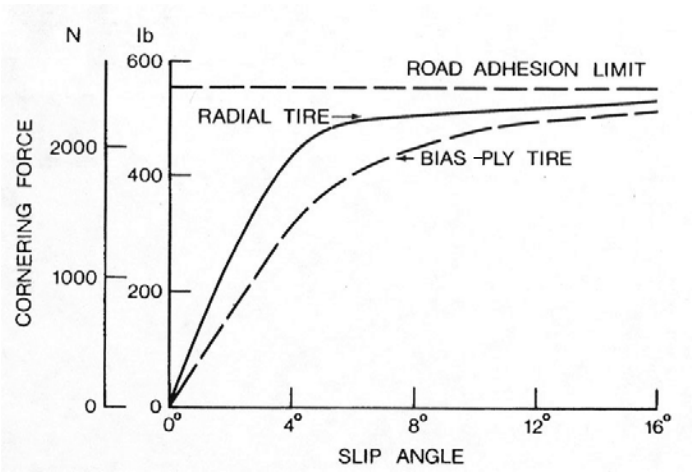


Figure 3.8: Cornering Characteristics of Car Tires

(heading) and the wheel travel direction. Figure 3.8 shows a typical plot of F_y as a function of α for a bias-ply and a radial-ply passenger car tire [van Eldik Thieme and Pacejka 1971, Wong 2001]. At small slip angles (up to 5°), the following linear relationships can be assumed:

$$F_{yf} = C_{\alpha f} \cdot \alpha_f \tag{3.6}$$

$$F_{yr} = C_{\alpha r} \cdot \alpha_r \tag{3.7}$$

where $C_{\alpha f}$ and $C_{\alpha r}$ are the cornering stiffness of front and rear tires, respectively. More complex models, such as the Magic Formula [Bakker et al. 1987, Bakker et al. 1989], could be used for more accurate simulation.

The above relationship is affected by longitudinal forces, e.g., braking or accelerating in a turn, which happen quite often. In general, a tractive or braking effort will reduce the cornering force that can be generated for a given slip angle, and the cornering force decreases gradually with an increase of the tractive or braking effort. This is due to the mobilization of the available local adhesion by the tractive or braking effort, which reduces the amount of adhesion available in the lateral direction. This is illustrated by the well-known friction ellipse concept, as shown in Figure 3.7, and one can predict the cornering force available at a specific slip angle in the presence of a tractive or braking effort, as shown in Figure 3.9 [Wong 2001].

In vehicle vertical vibration analysis, the cushioning characteristics of the suspension system and the pneumatic tires may be represented by various mathematical models. The most widely used and simplest model consists of a mass element and a linear spring in parallel with a viscous damping element, representing the fundamental mode of vibration. The spring constant and the damping coefficient can be determined from car experimental data.

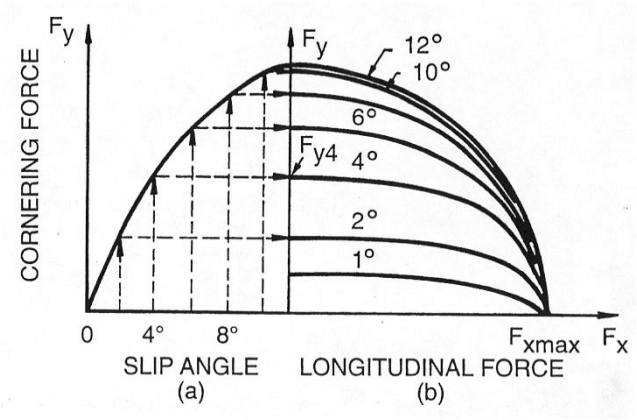


Figure 3.9: Construction of a Friction Ellipse for a Given Slip Angle

Figure 3.10: Screen Shot of the Dynamic Embodied Vehicle Model

3.2.3.2 Basic Vehicle Model

As mentioned above, simple vehicle simulation models with some realistic vehicle dynamics characteristics are desirable. A two-axle dynamic embodied vehicle model based on real passenger cars is developed for this purpose and described below.

The new vehicle model integrates all three motions (longitudinal, lateral, and vertical) of the vehicle, but it ignores the engine and transmission dynamics characteristics since they are not of major concern here. The model is developed in Webots 5 [Michel 2004] to simulate a front-wheel drive passenger car, which is based on a preliminary model developed at EPFL [Epiney 2004], as shown in Figure 3.10.

The latest version of Webots 5 provides a useful platform for developing a dynamic embodied simulation of multiple, intelligent vehicles. Webots 5 is a software package for fast prototyping and realistic simulation of customized mobile robots. It uses the Open Dynamics Engine (ODE)[5] library for realistic physics simulation. The ODE library is an open source, high performance library

[5]Refer to `http://ode.org`.

for simulating rigid body dynamics. Customized physics and vehicle dynamics properties can be implemented in Webots based on ODE. Therefore a real car-based vehicle model with realistic vehicle dynamics features can be developed in Webots with ODE.

The model shown in Figure 3.10 is composed of the vehicle body and four wheels. The vehicle body is able to move freely in all six DOF in the 3D space; while the wheels can all spin and move vertically relative to the body, plus the steering wheels can also yaw. So the whole vehicle model has 16 DOF in total ($16 = 6 + 3 \times 2 + 2 \times 2$), i.e., six DOF for the vehicle body, three DOF for each of the two steering wheels, and two DOF for each of the two non-steering wheels.

The following parameters are predefined constant parameters for a given vehicle under a certain operational condition:

- Wheel base L (m)
- Distance between C.G. and front axle center a (m)
- Distance between C.G. and rear axle center b (m)
- Equivalent total mass of vehicle m (kg)
- Cornering stiffness $C_{\alpha f}$ and $C_{\alpha r}$ (N/rad)

The vehicle model gets the following inputs from the driver controller updated in real time:

- Tractive or braking torque efforts acted on front and rear wheels T_{xf} and T_{xr} (N.m)
- Steering angle δ (rad)

The vehicle model makes corresponding movements according to the above parameters and inputs under specified environmental conditions.

Built in Webots based on ODE, this model has already incorporated basic rigid dynamics properties including typical steering dynamics response. Therefore the major task here is to establish and verify the vehicle model within the framework of existing ODE rigid body dynamics modules and to introduce special characteristics essential for the simulation of vehicle dynamics. It is also desirable to have a good graphic and animated representation of the traffic scenarios, such as the one shown in Figure 3.11, which appears realistic yet is simple enough to have a fairly fast simulation.

The next sections will introduce special vehicle dynamics features implemented especially for the dynamic embodied vehicle simulation model based on the basic model described above.

3.2.3.3 Joint Models

In a preliminary model [Epiney 2004], the driving and steering motions of the front wheels were modeled by two active *servo* nodes, respectively. One implements forward driving—longitudinal

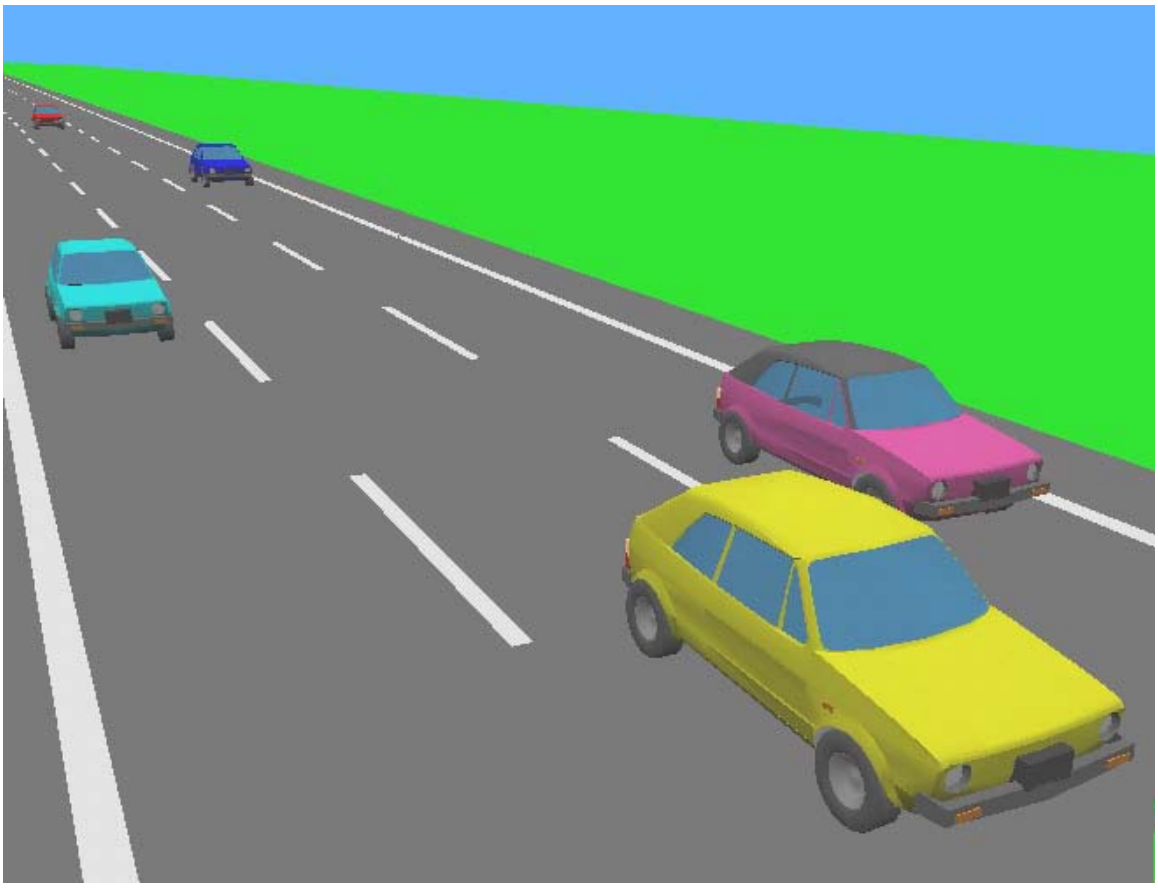


Figure 3.11: Sample Freeway Traffic Simulation with Dynamic Embodied Vehicle Models

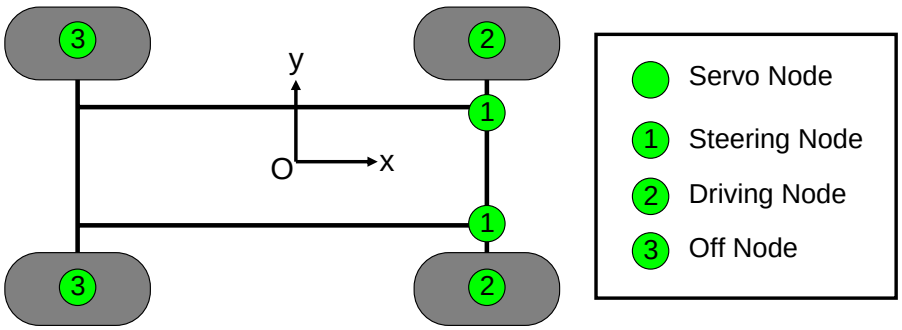
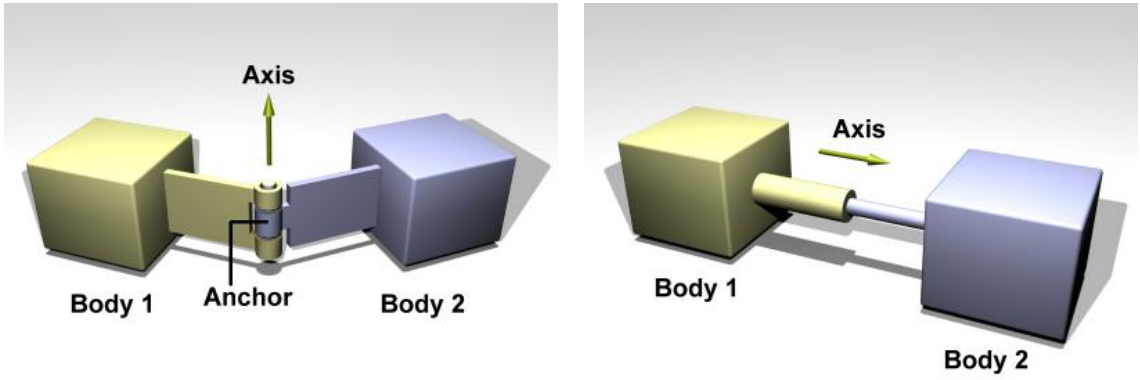


Figure 3.12: Schematic View of a Simple Two-axle Vehicle Model [Epiney 2004]



(a) (b)

Figure 3.13: The (a) Hinge and (b) Slider Joints Defined in ODE

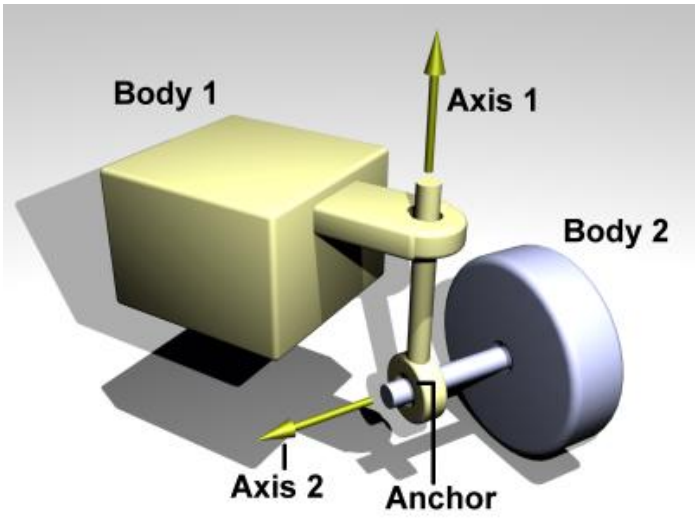
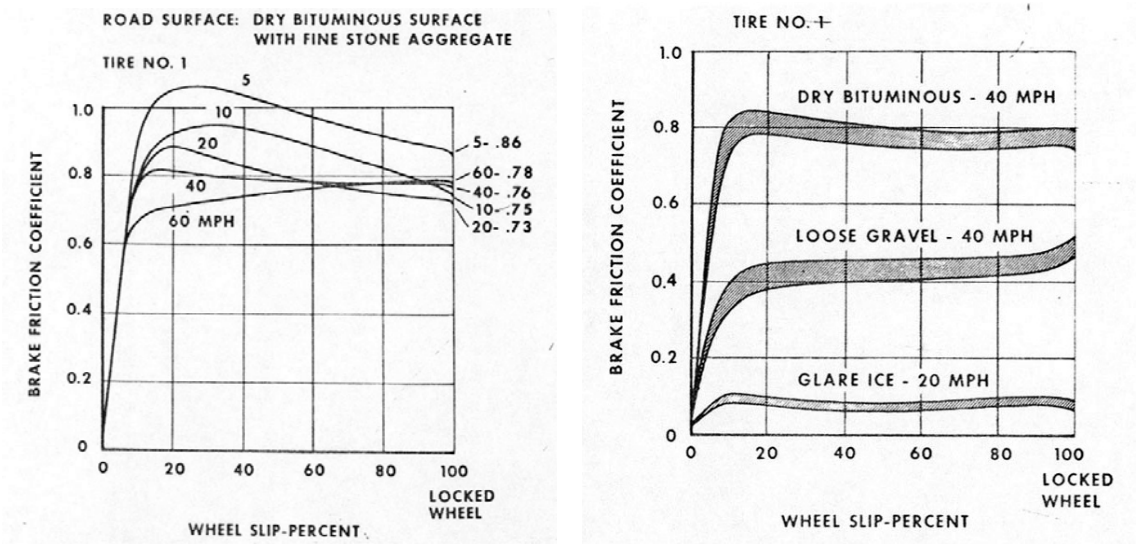


Figure 3.14: The Hinge-2 Joint Defined in ODE

movement, and the other realizes vehicle steering—cornering (yaw) movement, as shown in Figure 3.12. The rear driven wheels were simply modeled by two *off* servo nodes, allowing them to rotate freely. The servo node here was based on the *hinge* joint (as shown in Figure 3.13 (a)) defined in ODE, and it models a rotation servo motor, which tries to reach a given position and/or angular velocity goal under specified torque.

In addition, a customized *slider* joint (as shown in Figure 3.13 (b)) from the ODE can be introduced to allow relative vertical motions between the vehicle body and the wheels, simulating the car suspension system using a simple spring and damping system between the two.

An alternative way to implement all the relative motions between the vehicle body and the wheels is to use the ODE customized *hinge-2* joint (as shown in Figure 3.14), which is defined especially for car simulation. The hinge-2 joint is equivalent to two hinges connected in series, with different



(a) (b)

Figure 3.15: Variation of Friction Coefficient vs. Wheel Slip Curves on (a) Dry Asphalt and (b) Dry Asphalt, Loose Gravel, and Ice [Harned et al. 1969]

hinge axes. An example is the steering wheel of a car, which can both spin and steer along different axes. As shown in Figure 3.14, Body 1 can be the vehicle body and Body 2 the wheel, Axis 1 is the wheel steering (vertical) z axis and Axis 2 is the wheel spinning (lateral) y axis. Axis 1 can also function as a suspension axis, allowing limited relative vertical motions between the vehicle body and the wheels along Axis 1. The hinge-2 joint where Axis 1 is perpendicular to Axis 2 is equivalent to a universal joint with added suspension. Therefore all three motions (wheel steering, spinning, and vertical movements) of a steering/driving wheel can be conveniently integrated into just one joint model. For consistency, four hinge-2 joints can be applied between the vehicle body and its four wheels, respectively, where the steering motions of the rear (non-steering) wheels can be simply turned off. This is a more compact and integrated joint model especially customized for dynamic vehicle simulations.

In summary, a 16 DOF front-wheel drive vehicle model is implemented here, which allows the vehicle body to move in all six DOF in a 3D space, and each wheel to move in three DOF relative to the body.

3.2.3.4 Friction Model

As mentioned above, the friction coefficient μ between the tire and road is an important parameter limiting the maximum tractive and braking forces available from the tire-road contact, and it varies under different environmental and dynamical conditions. Figure 3.15 shows the variation of the brake friction coefficient with longitudinal wheel slip i on various surfaces and under different speeds

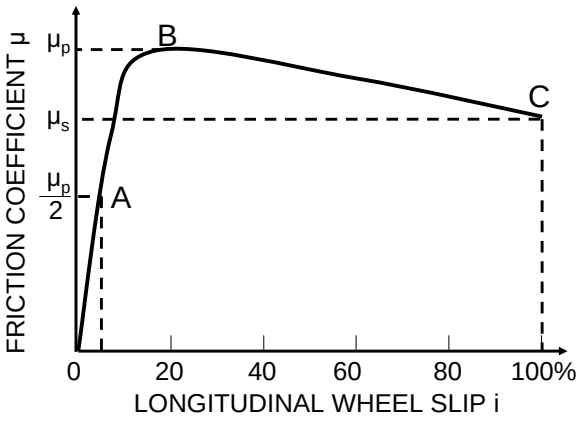


Figure 3.16: Typical Relationship between Friction Coefficient μ and Wheel Slip i

[Harned et al. 1969], where the wheel slip or skid is defined by [Wong 2001]

$$i = \begin{cases} \left(1 - \dfrac{v}{r\omega}\right) \times 100\% & v \leq r\omega, \text{ tractive slip} \\ \left(1 - \dfrac{r\omega}{v}\right) \times 100\% & v \geq r\omega, \text{ braking slip} \end{cases} \tag{3.8}$$

where v is the linear speed of the wheel center (i.e., vehicle body), ω is the angular speed of the wheel, and r is the rolling radius of the free-rolling tire.

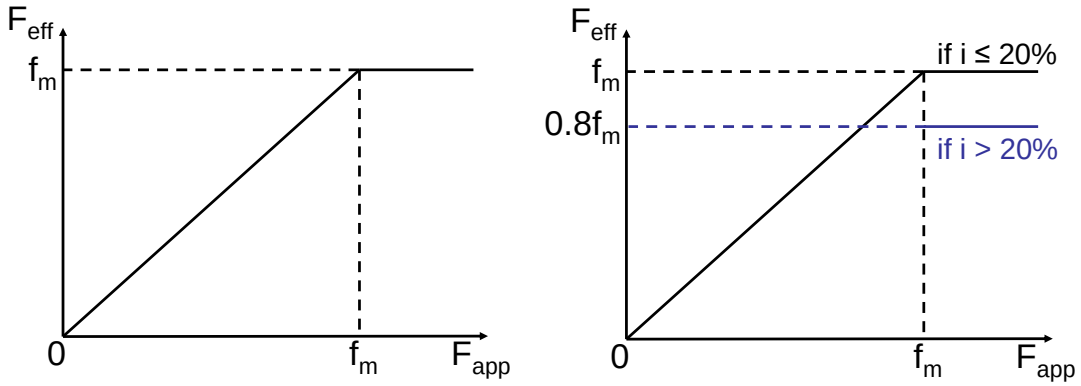
It can be observed from Figure 3.15 that the friction force available from the tire-road contact does not always increase linearly as the tractive or braking effort increases. When accelerating on a slippery or icy surface, often the tire rotates without equivalent forward movement, i.e., $v < r\omega$ and a slip results. If the vehicle does not move forward at all ($v = 0$) with tire spinning ($\omega > 0$), the slip is 100%, as defined in the first part of Equation 3.8. On the other hand, when braking on a slippery surface, sometimes the vehicle purely slides forward ($v > 0$) while the wheel is locked ($\omega = 0$); then the slip is also 100%, as defined in the second part of Equation 3.8.

In general, the tractive or braking effort first increases linearly with wheel slip because, initially, slip is mainly due to elastic deformation of the tire tread. This corresponds to section OA of the curve shown in Figure 3.16. A further increase of tractive or braking effort results in partially the tire tread sliding on the ground, and the curve becomes nonlinear, corresponding to section AB of the curve shown in Figure 3.16. Based on available experimental data, such as Figure 3.15, the peak value of friction coefficient μ_p is usually reached between 15 and 20% slip. Any further increase of slip causes the friction coefficient to fall from peak value μ_p to pure sliding value μ_s, as shown in Figure 3.16. Average peak and sliding values of the friction coefficients μ_p and μ_s on various surfaces are given in Table 3.1 [Taborek 1957].

A good approximation model of the curve shown in Figure 3.16 is the popular Pacejka "magical" model [Bakker et al. 1989], which uses nonlinear mathematical equations with many model parame-

Surface	Peak Value μ_p	Sliding Value μ_s
Asphalt and concrete (dry)	0.8–0.9	0.75
Asphalt (wet)	0.5–0.7	0.45–0.6
Concrete (wet)	0.8	0.7
Gravel	0.6	0.55
Earth road (dry)	0.68	0.65
Earth road (wet)	0.55	0.4–0.5
Snow (hard-packed)	0.2	0.15
Ice	0.1	0.07

Table 3.1: Average Values of Coefficient of Road Adhesion [Taborek 1957]



(a) (b)

Figure 3.17: The Original (a) and Modified (b) ODE Friction Models: F_{app} represents *applied* tractive or braking effort on the wheel while F_{eff} represents *effective* force available from the ground contact, with f_m representing the maximum friction limit.

ters to be tuned with real experimental data. For the purpose of this research, a simpler and more general model is desired. It can be observed from Figure 3.15 and Table 3.1 that $\mu_s \approx 0.8\mu_p$ in most cases, therefore the original oversimplified friction model implemented in ODE (shown in Figure 3.17 (a)) can be modified by a simple conditional linear piece-wise curve shown in Figure 3.17 (b). When this modified friction model is implemented, the slip ratio i is computed at each time step for each tire-road contact, and the maximum friction force limit f_m is adjusted to 80% of its original value when $i > 20\%$. For more details and examples, please refer to [Zhang 2005].

3.2.3.5 Rotational Motions

The three different rotational motions of the vehicle body are pitch, yaw, and roll, as shown in Figure 3.18, which are also simulated by the dynamic embodied vehicle model through ODE. Figure 3.19 shows the time histories of the three angles under a sample vehicle driving scenario, where the vehicle drove along alternate straight and curved roads and changed lanes regularly. First the vehicle accelerated on a straight road from static to its preferred speed (61 mph) for the first 12 s, during which only the pitch angle was a non-zero value, since the dynamic weight transfer effects

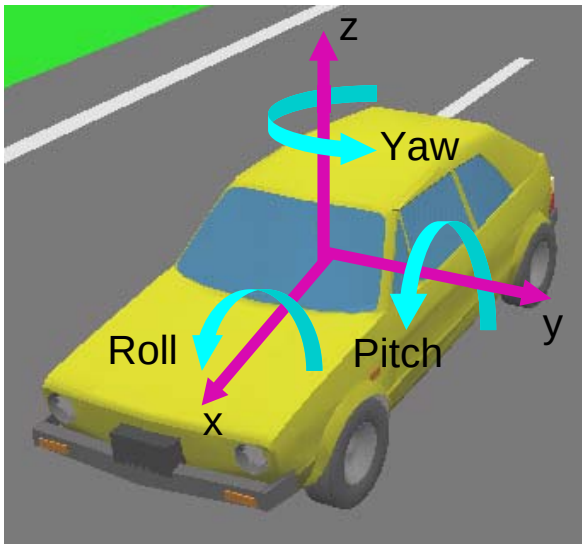


Figure 3.18: Illustration of the Three Rotational Motions: Pitch, Yaw, and Roll

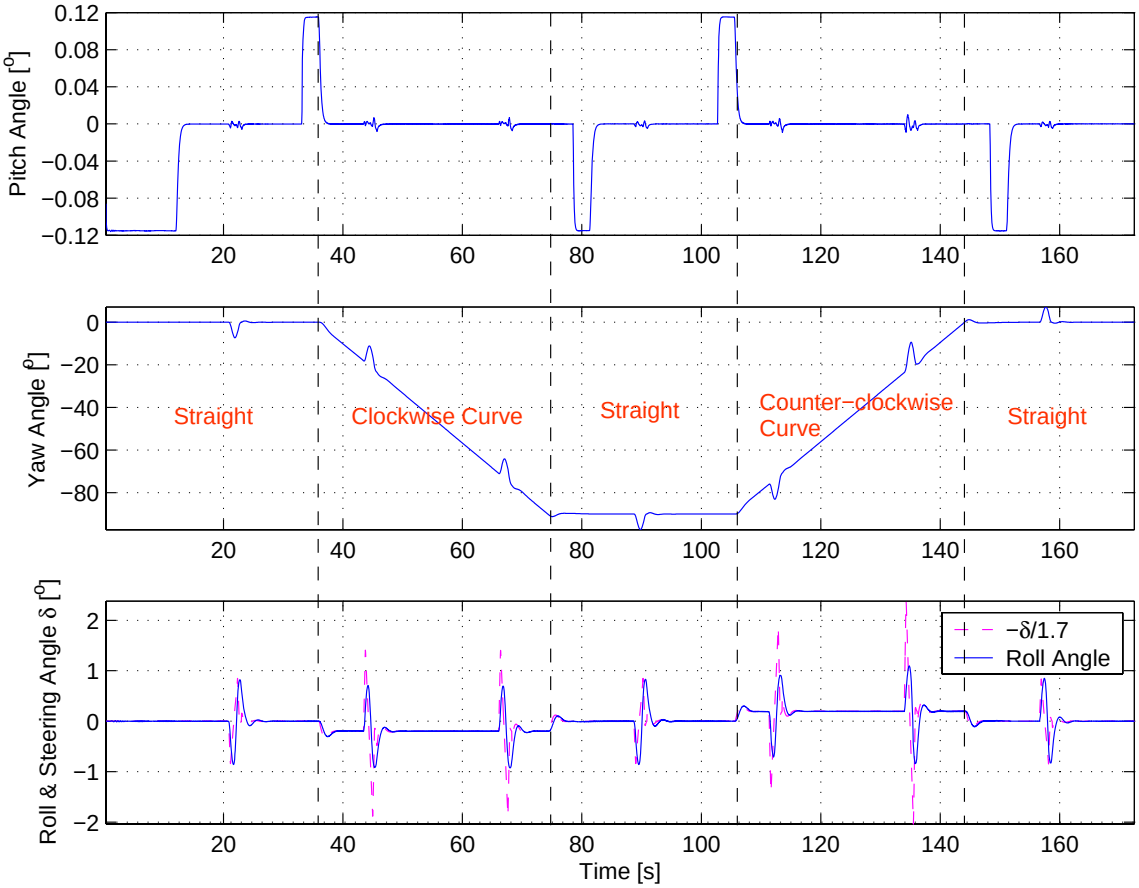


Figure 3.19: The Simulated Vehicle Pitch, Yaw, and Roll Motions with Steering Angles

due to acceleration caused the vehicle to pitch backwards. Then the vehicle entered a clockwise curve at time 37 s, before which it reduced its speed, and exited this curve at time 76 s, after which it resumed its preferred speed. It also went through a counter-clockwise curve from time 106 to 145 s, with similar speed control as before. In addition, the vehicle changed lanes at time 22, 44, 68, 89, 112, 136, and 158 s, respectively. Note that the yaw and roll angles changed dramatically due to the lane changes, while the roll angle followed the steering angle changes quite closely, especially on straight roads. Moreover, the magnitude of the pitch and roll angles depends on the elasticity and damping characteristics of the suspension system, which are adjustable vehicle model parameters to fit real car experimental data.

3.3 Driver Models

Simple yet realistic driver behavior models are also important for intelligent vehicle simulations. An ideal alert driver model should be able to drive safely under typical and normal conditions. Specifically, they should be able to maintain the vehicle in the desirable lane within lane boundaries on both straight and curved roads (lane-keeping behavior), keep a safe distance with vehicles ahead to avoid rear-end collisions even if they might brake suddenly (car-following behavior), change lanes safely as needed (lane-changing behavior), watch out for other lane-changing (cut-in) vehicles, and keep its preferred driving speed as much as possible. The three main target driver behaviors needed to be implemented are

- Car-following behavior
- Lane-keeping behavior
- Lane-changing behavior

The driver behavior models take sensor measurements updated in real time as dynamical inputs, including host vehicle speed v_H, distance to vehicle ahead (range R), relative speed to vehicle ahead (range rate RR), etc., for longitudinal motion control; lane deviation error E, vehicle heading angle and current steering angle δ, etc., for lateral motion control. Different driver models also have different preference parameters such as preferred speed v_{pref} and time headway t_h, etc.

There are different ways to implement these different driving behaviors, as described below.

3.3.1 Helbing Model

Many of the microscopic traffic simulation models proposed in the literature are focused on the car-following behavior, as reviewed in Section 3.1. The key point of car-following behavior is setting vehicle acceleration according to the current situation and driver preferences, which include the

current gap and desired gap between the host vehicle and the lead vehicle, current speed and relative speed to the lead vehicle, driver's preferred speed and driving style, etc.

Dirk Helbing presented an intelligent-driver model (IDM) [Helbing et al. 2002], which computes the target acceleration a_x as follows:

$$a_x = a_{acc}\left[1 - \left(\frac{v}{v_{pref}}\right)^{\alpha} - \left(\frac{R^*(v, RR)}{R}\right)^2\right] \tag{3.9}$$

where a_{acc} is the maximum *acceleration* limit chosen; v and v_{pref} are the current and preferred host vehicle speed, respectively; α is the acceleration exponent parameter; and R and R^* are the current and desired range (gap) between the host vehicle and the lead vehicle, respectively. Equation 3.9 is a superposition of the acceleration term $a_{acc}\,[1 - (v/v_{pref})^{\alpha}]$ on a free road, and the (braking) deceleration term $-a_{acc}(R^*/R)^2$, describing the interactions of the host vehicle with other vehicles ahead. The desired range R^* dynamically varies with current host vehicle speed v and the closing speed relative to the lead vehicle (range rate RR):

$$R^* = R_0 + R_1\sqrt{\frac{v}{v_{pref}}} + v \cdot t_h + \frac{v \cdot RR}{2\sqrt{a_{acc} \cdot a_{cft}}} \tag{3.10}$$

where R_0 and R_1 are constant distance parameters for each different driver, t_h is the preferred time headway, and a_{cft} is the comfortable *deceleration* level of the current driver. The IDM parameters used by Helbing are summarized in Table 3.2.

Parameter	v_{pref}	α	t_h	a_{acc}	a_{cft}	R_0	R_1
Value	75 mph	1	1.2 s	0.8 m/s^2	1.25 m/s^2	1 m	10 m

Table 3.2: The IDM Parameters

It can be observed from Table 3.2 that the Helbing model considers quite small values for acceleration parameters since it agrees with their macroscopic traffic model [Helbing et al. 2002], which would ignore traffic accidents because they happen so rarely in the macro scene. However, the intelligent vehicle systems aim to improve traffic safety by focusing on avoiding collisions, which need more aggressive driver behavior models.

3.3.2 Rule-based Model

A heuristic approach to implementing a driver model is to control the vehicle by a series of logical commands according to driver preferences and sensor measurements as well as vehicle model responses updated in real time. Simple driver behaviors can be easily implemented this way, and some control parameters and thresholds need to be hand-tuned or evolved to get desirable behaviors.

For example, the car-following behavior can be implemented following the logic scheme shown in Figure 3.20, where v_H and v_L are current speeds of the host vehicle and the lead vehicle, respectively,

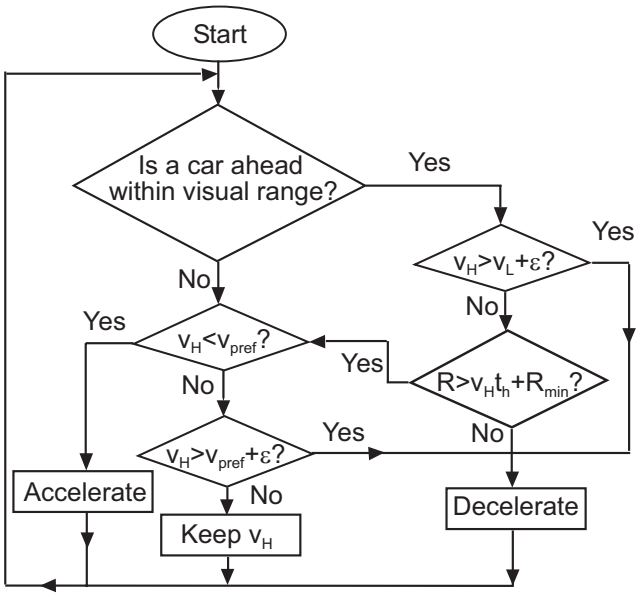


Figure 3.20: The Logic Scheme of a Simple Car-following Behavior

ε is a small positive constant speed buffer, and R_{min} is a positive constant distance parameter. The blocks "Accelerate" and "Decelerate" could be further expanded to implement more realistic car-following behaviors, which could refer to the Helbing model or be hand-tuned through trial and error. Although the logic scheme shown in Figure 3.20 seems simple, it could take considerable time and effort to fine tune the parameters to get desirable and realistic car-following behaviors.

Similarly, simple lane-keeping behavior can be implemented following the logic flow chart shown in Figure 3.21. This would apparently work for this simple behavior with appropriate parameters, which would also take much time to be tuned by hand. When the desirable behavior gets more complex, the time and effort needed to develop such driver models increase exponentially.

Lane-changing behaviors are generally triggered by a slower vehicle ahead in the same lane as the host vehicle (besides the merge and exit scenarios), and when the desired adjacent lane is available. The host vehicle is generally expected to be able to drive at a higher speed that is closer to its preferred speed after changing lanes. According to U.S. traffic rules, it is OK to change to either the left or right lane to pass a slower vehicle and it is not necessary to change back after that. Hence the logic scheme used to initiate a lane change, as shown in Figure 3.22, would check both left and right adjacent lanes for availability first, then change to an available lane if possible, and no attempt is made to change back after passing. The lane-changing behavior itself can be simply implemented using the logic scheme shown in Figure 3.23.

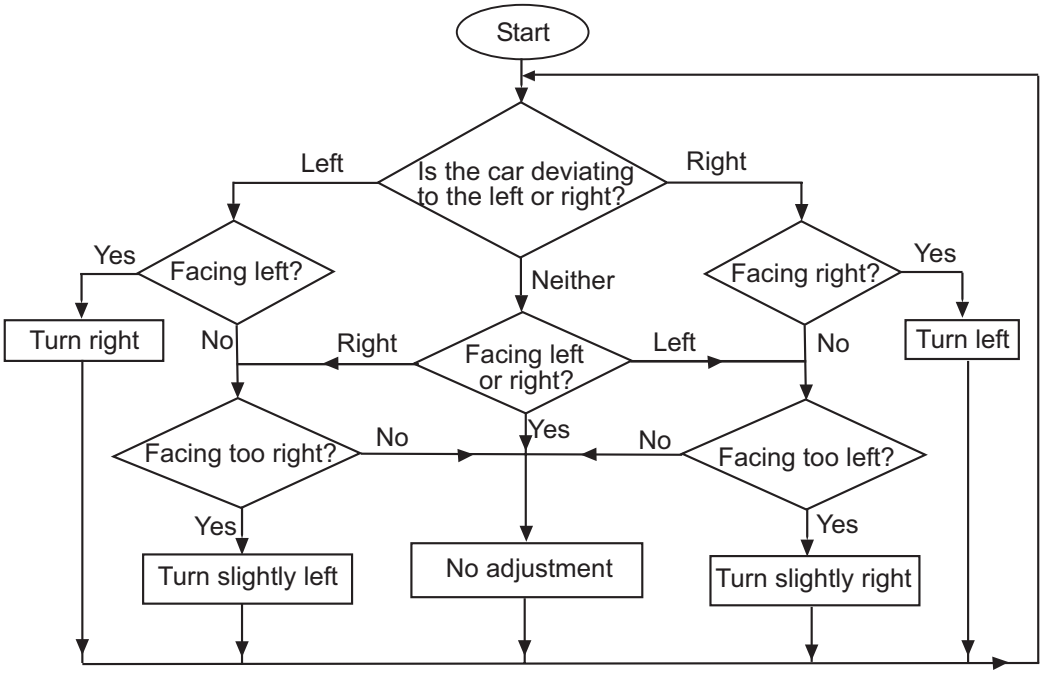


Figure 3.21: The Logic Scheme of a Simple Lane-keeping Behavior

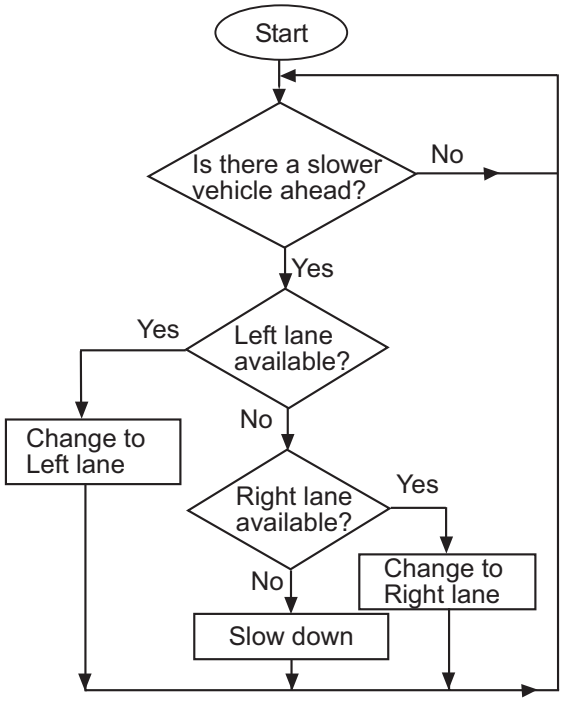


Figure 3.22: The Logic Scheme for Initiating a Lane Change

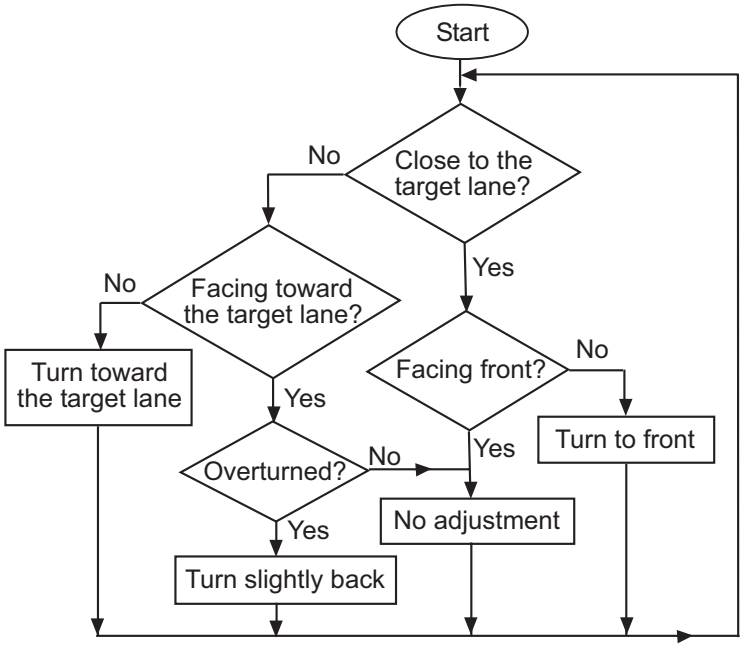


Figure 3.23: The Logic Scheme of a Simple Lane-changing Behavior

3.3.3 PID Control Model

Alternatively, the lane-keeping behavior, which follows both straight and curved roads, can be implemented using a proportional-integral-derivative (PID) compensation controller, which is the most common control methodology in classical feedback control systems [Phillips and Harbor 2000].

The control input is the lane deviation error E, which is the shortest distance from the car center to the center line of the lane. The PID controller tries to reduce this error to zero under varying driving conditions, such as different driving speeds and/or road curvatures. The output of the PID controller is the steering angle δ, and the PID compensator can be described by the following equation:

$$\delta(t) = K_P E(t) + K_I \int_0^t E(\tau)d\tau + K_D \frac{dE(t)}{dt} \tag{3.11}$$

where K_P, K_I, and K_D are PID controller parameters. The proportional term $K_P E(t)$ is the major term in the feedback control loop to compensate for the error in a timely way, but using the P controller alone usually leaves a steady state error due to changing operating conditions. The integral term $K_I \int_0^t E(\tau)d\tau$ helps reducing the steady state error to zero in the long run. The derivative term $K_D \frac{dE(t)}{dt}$ is used to anticipate a change in the system and speed up a controller's response to the change.

Figure 3.24 shows the time histories of the vehicle lane deviation error E,[6] its integral iE, and lane position Y, as well as the vehicle steering angle δ and wheel speed ω, under the same vehicle

[6] E was not updated (fixed) during lane changes.

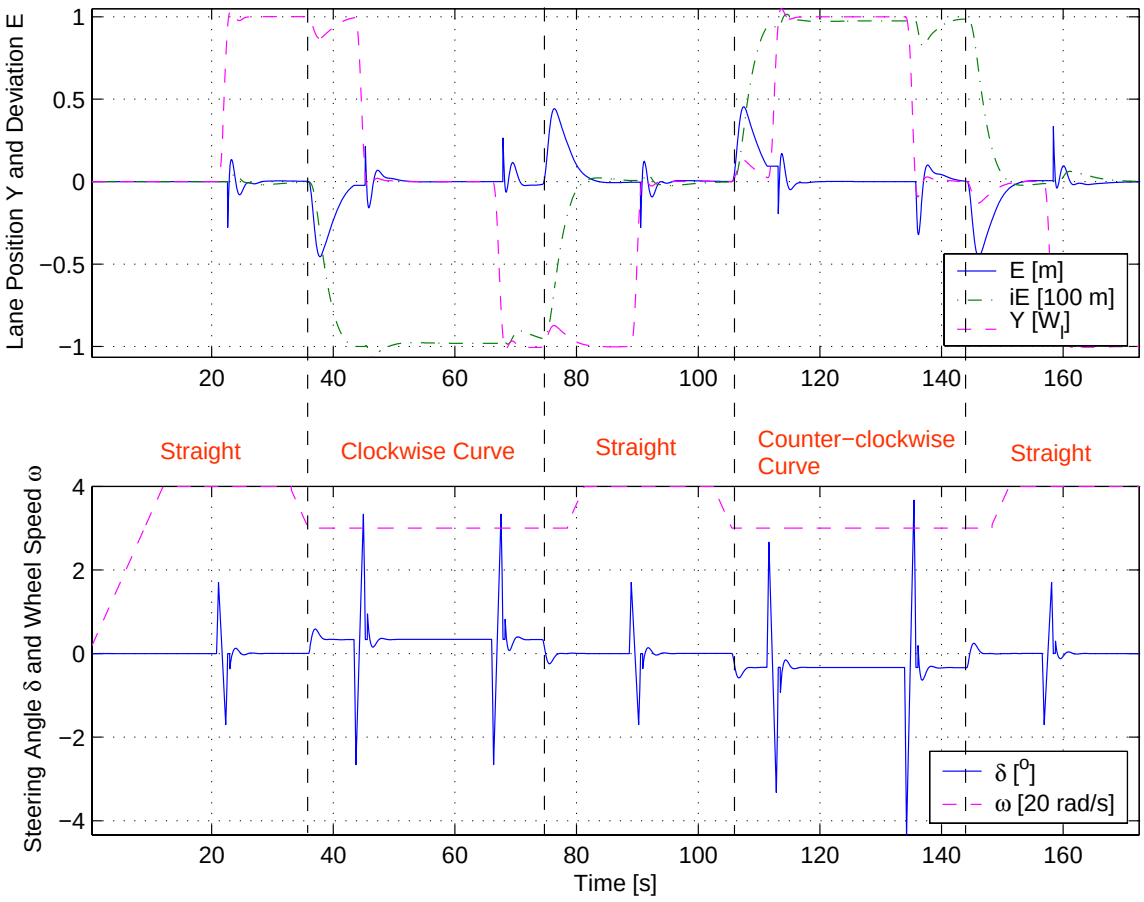


Figure 3.24: (*Top*): Time Histories of Vehicle Lane Deviation Error (E), Integral Lane Deviation Error ($iE \equiv \int_0^t E(\tau)d\tau$), and Lane Position Y in terms of Lane Widths ($W_l = 3.5\,\mathrm{m}$); (*Bottom*): Time Histories of Vehicle Steering Angle δ and Wheel Speed ω

driving scenario as described in Section 3.2.3.5 and shown in Figure 3.19.

Again it takes considerable time and experience to tune the PID controller parameters by hand for each different operational situation, e.g., different driving speeds and/or road curvatures. The different parameters tuned for each different situation can then be stored in a table or generate a fitted curve or piece-wise linear model for reference by the driver model. It turns out that the controller works best (i.e., achieves smallest lane departure error) when the driving speed is low, and the error increases with the speed, which agrees with one's expectation. For instance, the maximum lane departure error generated when a car entering a curve of radius 500 m from straight road is 0.6 m at speed 65 mph and 0.3 m at speed 38 mph.

The lane-keeping PID control algorithm can also be applied after a lane-changing algorithm to bring the vehicle center to align with the new lane center laterally and keep the vehicle in the new lane, as shown in Figure 3.24.

3.3.4 Evolved Model

Evolutionary methodologies as described in Chapter 2 can also be applied to evolve different human driver behavior models with parameters tuned automatically. An example will be presented in Section 5.4, where a smooth driving behavior is evolved.

Chapter 4

Evolutionary Design of Sensory Systems

As a first case study, the evolutionary engineering design synthesis methodology presented in Section 2.3 is applied to develop a collective sensory system for intelligent vehicles [Zhang et al. 2003a, Zhang et al. 2003b]. This case study is concerned with the configuration design of a collective on-board traffic monitoring sensory system for intelligent vehicles, addressing all of the engineering design challenges mentioned in Section 1.2. Different evaluation tests are utilized to synthesize novel design solutions efficiently. To assist the engineers in the design decision-making process, the evolutionary design synthesis methodology is applied to generate the full family of Pareto optimal design solutions, representing different engineering design trade-offs under various conditions and formulations of the design problems [Antonsson et al. 2003].

4.1 Introduction

As mentioned in Section 1.1, intelligent vehicles aim to improve traffic safety as well as fluidity by reducing collisions and collision impacts if unavoidable. To achieve this goal, an intelligent vehicle should be equipped with necessary sensorial capabilities to monitor current dynamic traffic and road conditions as well as host vehicle states in real time. Then the control system will be able to identify and assess potential threats according to sensor data, and generate appropriate warning signals and even brake automatically to avoid collisions if necessary. Therefore the first case study is focused on the design of a collective intelligent sensory system, which identifies all potential dangers around the vehicle and, for purposes of improving traffic safety and reducing crash risk, gives appropriate warnings to bring attention to the unsafe factors probably overlooked by the driver.

This is not a simple problem since an individual intelligent vehicle is expected to efficiently perceive, decide, and act in a heterogeneous group [Martinoli et al. 2002]. In other words, the key question is: How to design an intelligent vehicle that has to share the road with a group of other

vehicles that it cannot control, nor can it assume collaboration, nor can it necessarily exchange information with, e.g., standard cars? Even if it is assumed in the case study proposed here that there is no decision and/or action taken by the host vehicle based on sensory data yet (i.e., no consequences of its individual intelligent decision on the group behavior), deciding how many sensors, what characteristics they should have, and where they should be placed in order to achieve high monitoring performance and low cost, is not a trivial task, especially in a highly dynamic and noisy traffic scenario. Of course, if there exist no considerations of cost, no noise, no variations in manufacturing, and only a limited set of available choices for the sensors, a standard engineering hand-coded solution may suffice and outperform what an evolutionary algorithm may find.

Although it is not guaranteed that a global optimum from a strict mathematical point of view can always be generated, an evolutionary algorithm is able to discover some good and near-optimum solutions suitable for the engineering design use. Highly tuned systems are often sensitive to small imperfections, so engineers commonly design solutions to be slightly suboptimal to avoid such problems and increase robustness [Newman 2000].

4.2 Background

Today hundreds of sensors are used in a standard automobile, measuring linear and rotational motions (positions, distances, speeds, accelerations), pressures, temperatures, air flow, fluid levels and quality, etc., of all parts of the automobile. Automotive sensors must satisfy a difficult balance between accuracy, robustness, manufacturability, interchangeability, and low cost [Fleming 2001]. Since the 1990's there has been an increasing interest in developing new automotive sensors for intelligent transportation systems to improve traffic safety, where obstacle detection and range/speed measurements are of major importance. Several main technologies used for these purposes are

- Millimeter-wave radar
- Laser radar (or Lidar)
- Ultrasonic sensors
- Machine vision (camera) systems
- Infrared (IR) sensors

Millimeter-wave radars operate at specified government-regulated frequencies ranging between 24 GHz and 77 GHz, and they can accurately measure the range, range rate (i.e., relative speed), and azimuth angular position of detected objects under a wide range of environmental conditions (rain, snow, dirt, fog, darkness, etc.). The most popular waveform used by automotive radars is frequency-modulated continuous wave (FMCW), which can track and measure the range and range rate of as

many as 32 targets. Range is derived from the transit time of the FMCW return signals, while range rate is derived from the doppler frequency shift of the return signal. Mechanical or electronic beam scanning can be utilized with a data update rate of 5 to 50 Hz. The measuring ranges for distance and speed are customizable for short-, mid-, and long-range (up to 240 m) measurement needs with different azimuth and elevation angles [Mende et al. 2005]. They can be mounted invisibly behind other materials (e.g., bumper systems).

Laser radar, or Lidar (i.e., light + radar) emits narrow, pulsed, infrared beams at wavelengths around 850 nm. Short-duration (25 ns) but high power laser pulses are emitted over wide range of beam-scan (both horizontal and vertical) directions. Target distance is determined by transit times of individual pulses. Speed information is derived from range information. Beam scanning of automotive lidars can be accomplished by mechanically scanning systems, electrically switched beam systems, or electro-mechanically driven mirror-scan mechanisms. Distances up to about 250 m can be measured with resolution of a few cm and update rates of 10 to 50 Hz. Lidar sensors feature high accuracy, wide angular coverage, and precise target location, but they need to be cleaned and their performance is diminished in bad weather situations (e.g., heavy rain, snow, dirt, etc.).

Automotive ultrasound sensors are mainly used for parking aid applications because they offer wide-area, near-distance beam coverage with relatively low cost. Current automotive ultrasound sensors cover distances from about 10 cm up to 2.5 m with resolution of a few cm. These have been in use for over 20 years; they are designed for low speed maneuvering and are not suitable for high speed driving. Ultrasound sensors need to be visibly mounted at the vehicle, which might be inconvenient for vehicle body design.

Machine vision systems use one or more cameras to monitor the current traffic situations. Advanced image processing software is usually required to provide further information, such as lane recognition, object (e.g., traffic signs) classification, and forward path prediction/identification, etc. They are useful in the lane departure warning systems, where a vehicle's position relative to roadway lane markers is monitored and a warning is issued when the vehicle drifts out of the lane. They can also help collision avoidance systems to identify whether an object detected is in the path of the host vehicle.

Pyrometers are passive infrared sensors evaluating temperature differences between objects and their environment through measurement of infrared radiation energy. They are often used in night vision systems and can help detect and classify pedestrians, deer, other cars, etc.

In summary, all technologies have certain advantages. On the other hand, fusion of sensing information acquired by different sensing technologies offers the highest potential to further increase the overall sensing performance and reliability by complementing each other's advantages. Currently, fusion of radar and camera information seems to be the most promising candidate [Hoess et al. 2004].

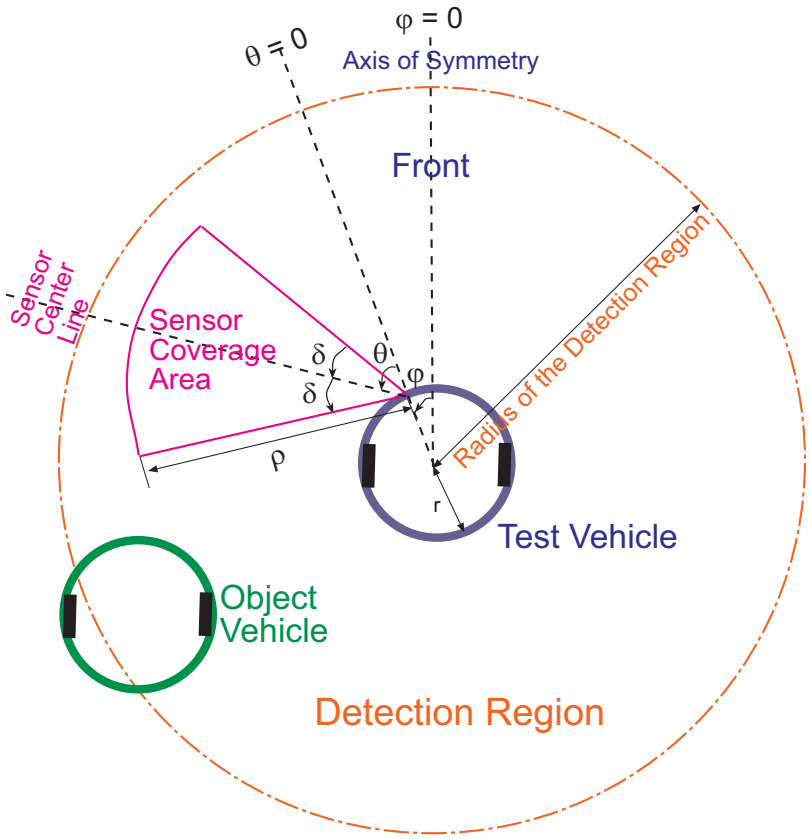


Figure 4.1: Sensor Parameters and the Target Detection Region

4.3 Problem Definition

The case study investigated in this chapter is a simple problem in a complex (dynamic and noisy) environment. The goal is to determine the optimal configuration (such as number, type, and placement) of distance sensors (e.g., radar or lidar sensors) on an intelligent vehicle, in order to monitor the traffic (i.e., other vehicles) in a preestablished desired detection region around the host vehicle in realistic traffic scenarios. The vehicle model used here is the kinematic embodied model, as described in Section 3.2.2, and the detection region chosen is circular around the host vehicle, as shown in Figure 4.1. An object vehicle is considered detected by the collective sensory system if the vehicle's body has overlap with at least one sensor's scanning area or ray.

4.3.1 Sensor Parameters

Sensors are mounted on the periphery of the vehicles, as shown in Figure 4.1. The type and placement parameters, as well as the number of sensors, are the design variables to be determined and optimized according to the designer's preferences on various performance measures and the trade-off strategies chosen, which will be described in detail in Section 4.3.3. Except for the number of sensors, which

may take any positive integer values, all other design variables are encoded as discretized real numbers selected from predefined finite feasible ranges. The placement parameters of each sensor are characterized by two angles: the position angle φ (the angle between the front direction of the vehicle and the radius pointing to the sensor's mount) and the orientation angle θ (the angle between the radius pointing to the sensor's mount and the center line of the sensor's scanning area), as shown in Figure 4.1. The type of each sensor is specified by its range ρ and cone of view δ. Therefore, each sensor is characterized by four design variables, and the number of design variables for a collective sensory system with n sensors is $4n$.

Each sensor also has a cost factor that depends on its range ρ and cone of view δ.[1] Typically the sensors with wider cones of view and longer ranges have a higher cost. This relationship can be determined from real sensor data or sensor models. A simple linear relationship is assumed in this case study:

$$cost_i = c_1\rho_i + c_2\delta_i + c_3 \tag{4.1}$$

$$Total\_cost = \sum_{i=1}^{n} cost_i \tag{4.2}$$

where $cost_i$, ρ_i, and δ_i are the ith sensor's cost, range, and cone of view, respectively; c_1, c_2, and c_3 are constant coefficients; n and $Total\_cost$ are, respectively, the number and the total cost of all sensors used in the current sensory system. Note that $cost_i$, ρ_i, and δ_i are all positive real numbers except that δ_i is also allowed to equal zero when the ith sensor is a line sensor.

As an important competing factor in the engineering design synthesis, the designer's preference for cost will be defined and incorporated into the fitness function, which will be introduced later in Section 4.3.3.

This seemingly simple case study problem reflects all of the engineering design challenges mentioned in Section 1.2. First, the optimal number of sensors is unknown, hence the number of design parameters as well as the complexity of the design solution is also open and increases with the number of sensors in the solution. Second, improving the coverage of the detection region and at the same time keeping a reasonable total cost for the whole sensory system are the two main design objectives here, whose relative importance lies in the aggregated fuzzy fitness function, described in Section 4.3.3, that leads to a trade-off between the two. Moreover, the evaluation process of candidate design solutions may be stochastic or deterministic, depending on the evaluation test used, which will be described in Section 4.3.2.

[1] For a real sensor, besides its range and cone of view, the sensor cost may also depend on several other factors, such as accuracy, scanning frequency, and power, etc., which are not included in this case study for simplicity.

4.3.2 Evaluation Tests

To understand the role of noise in shaping the evolved solutions, and to find out the best and most efficient evaluation tool for this kind of design synthesis, six different types of evaluation tests are implemented [Zhang et al. 2003a]: static, 1D/2D quasi-static, 1D/2D full coverage, and a kinematic embodied test, as shown in Figures 4.2 as well as Figure 3.2. Static and full coverage (FC) tests are deterministic tests while quasi-static (QS) and embodied tests are stochastic tests, where different evaluation results (fitness values) are obtained by repeating the same evaluation test multiple times for a given solution.

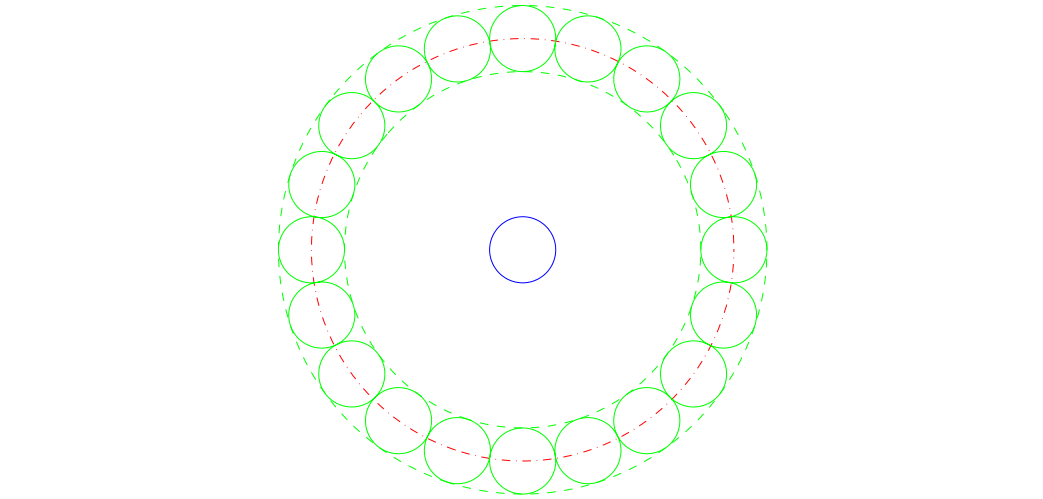
As mentioned in Section 3.2.2, sample traffic scenarios based on the kinematic embodied vehicle models are simulated in Webots, where candidate sensory configuration design solutions are embedded on the test vehicle to detect and monitor other object vehicles that enter the detection region of the test vehicle. Each evaluation span here contains 2000 simulation time steps, representing 128 seconds in real time. The noise involved in this type of evaluation test, based on the kinematic embodied simulation, includes random initial conditions (e.g., position and preferred speed) for each vehicle at each evaluation span, sensor and actuator noise (e.g., wheel slip) introduced to the kinematic embodied vehicle model, and variations in driver behavior.

From the kinematic embodied simulation, 1D and 2D vehicle occurrence probability density functions can be generated as described in Section 3.2.2 and shown in Figure 3.4 and 3.5. These PDF's represent an accumulation of vehicle occurrences for 5000 evaluation spans, averaging the temporal variations involved in the kinematic embodied traffic simulation, and capture basic characteristics of the simulated traffic scenarios. Based on these PDF's, less computationally expensive and more abstracted traffic simulation tests, such as the quasi-static and full coverage tests, can be defined in terms of the simple point model as described in Section 3.2.1.

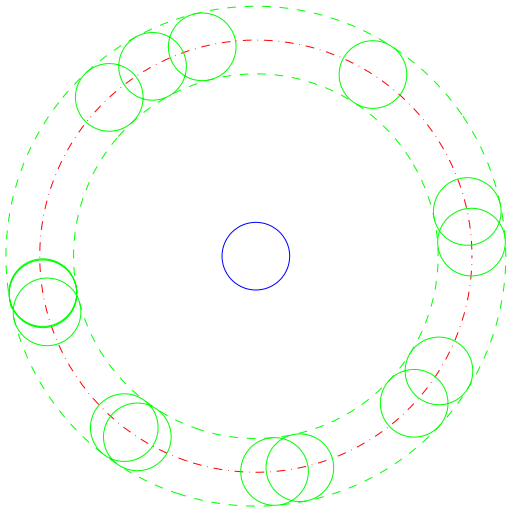
In the quasi-static tests, as shown in Figure 4.2 (b)(c), the test vehicle lies at the center statically, and object vehicles are generated randomly according to the PDF's on a ring (1D) or in the area (2D) within the detection region. While in the full coverage tests, the object vehicles are placed systematically along the ring (1D) or the area (2D) within the detection region, as shown in Figure 4.2 (d)(e), where the PDF's are used to weigh the detection of the object vehicle at each position in order to estimate the coverage achieved by the current sensory solution in the underlying traffic scenario represented by the PDF's, as explained in Section 4.3.3. In the static test shown in Figure 4.2 (a), 20 static object vehicles are distributed evenly on the same 1D ring, representing a simple control experiment, which is not related to a traffic scenario at all.

The six types of evaluation tests simulate the traffic scenarios with different levels of abstraction and significantly different simulation time costs, as shown in Figure 4.3 and Table 4.1.[2] In general, more realistic simulations are relatively more computationally expensive. To be more efficient, a

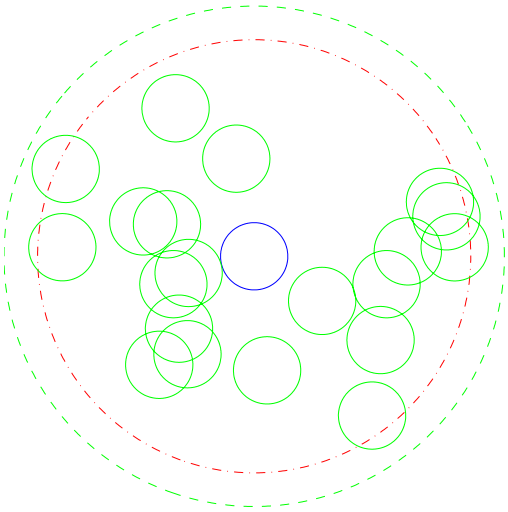
[2]The experiments are conducted on computers with 1.5 GHz AMD CPU.



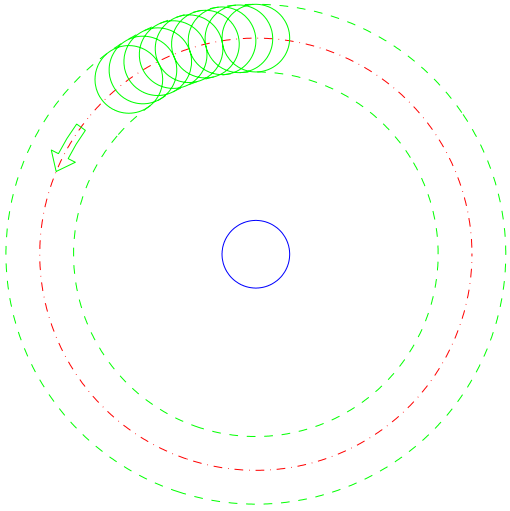
(a) Static



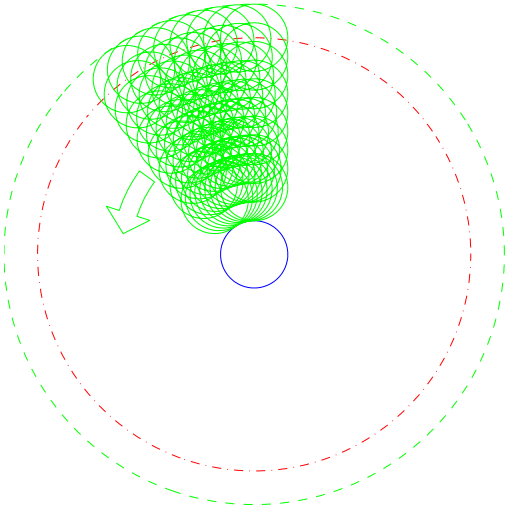
(b) 1D Quasi-static



(c) 2D Quasi-static



(d) 1D Full Coverage



(e) 2D Full Coverage

Figure 4.2: Graphical Representation of Different Types of Evaluation Tests Based on Point Model

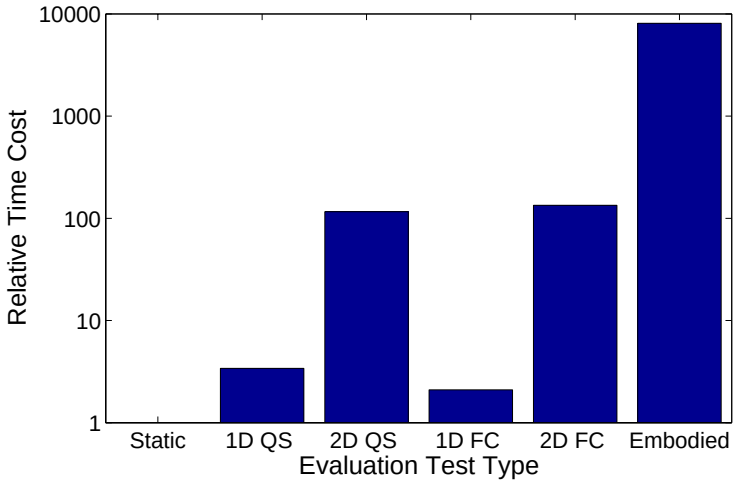


Figure 4.3: Approximate Relative Time Costs of Different Evaluation Tests Plotted on a Log Scale

Evaluation Test	**Time for 2500 Evaluations**	**Relative Cost**
Static	4.6 seconds	1.0
1D Quasi-static	15.6 seconds	3.4
2D Quasi-static	8.9 minutes	116
1D Full Coverage	9.7 seconds	2.1
2D Full Coverage	10.3 minutes	134
Embodied (Kinematic)	10.4 hours	8100

Table 4.1: Approximate Time Costs of Different Evaluation Tests

new type of evaluation test could be a *hierarchical* test that combines some of these basic types of evaluation tests. In the hierarchical test, an abstracted simulation test is performed first as a prescreening test prior to a more realistic and computationally more expensive test, which then in turn also serves as a pretest of an even more realistic and expensive test, and so on. Therefore, only if an individual solution performs well enough in a pretest, would it have a chance to be evaluated under a more realistic test at the next level up in the hierarchy. In this way more computational time will be invested on more promising solutions, and poor solutions can be recognized and eliminated quickly with little simulation time cost. Although no results are presented in this thesis based on a series of hierarchical tests, this approach will be applied in future work when appropriate, and is expected to be plausibly more time-efficient, especially for cases with several evaluation tools of different levels of abstraction and time costs available.

4.3.3 Fitness Function

As mentioned above, the "goodness" of design solutions is evaluated by fitness functions defined by the designers according to the desired design goals. In this case study, the goal is to achieve the best

coverage of the detection region of the test vehicle while at the same time maintaining a reasonable cost for the whole sensory system. Other performance criteria such as the resolution and reaction time of the sensory system are ignored in this case study for simplicity.

First, the *Coverage* under various evaluation tests is computed as follows:

$$Coverage = \sum_{i=1}^{V} k_i \cdot \mathrm{PDF}(\alpha_i, r_i) \tag{4.3}$$

where V is the number of vehicles effectively appearing within the detection region during the evaluation span; k_i is 1 if the ith object vehicle is detected, or 0 if it is not; α_i and r_i are, respectively, the approaching angle and distance of the ith object vehicle relative to the test vehicle. The PDF indicates the weight of importance at each particular position α_i and r_i. For full coverage tests, the PDF is generated from the vehicle occurrence data, as those shown in Figure 3.4 and 3.5; while for all other tests, the PDF is simply a constant $(1/V)$ for any α_i or r_i and $V > 0$. Therefore the $Coverage \in [0, 1]$ represents the effective percentage of object vehicles detected by the collective sensory system.

4.3.3.1 Preferences

As mentioned above, the fuzzy fitness function is based on the designer's overall preference on each candidate design solution, which depends on the individual fuzzy preferences on all relevant performance criteria achieved by the candidate solution. All design preferences here are expressed using fuzzy sets and take real values between 0 (totally unacceptable) and 1 (completely acceptable).

In this case study, the fuzzy preference functions, $\mu_{coverage}$ and μ_{cost}, are defined for the two competing factors, *Coverage* and *Total_cost*, respectively. The preference for coverage ($\mu_{coverage}$) is simply defined as a power function of *Coverage* with power β, showing the designer's preference for better coverage; while the preference for cost (μ_{cost}) is chosen to decrease linearly from 1 to 0 when *Total_cost* increases from A to B ($0 \leq A < B$):

$$\mu_{coverage} = Coverage^{\beta} \qquad \beta \in \mathcal{N} \tag{4.4}$$

$$\mu_{cost} = \begin{cases} 0 & Total\_cost \geq B \\ 1 & Total\_cost \leq A \\ \dfrac{B - Total\_cost}{B - A} & \text{otherwise} \end{cases} \tag{4.5}$$

where $\beta = 2$, $A = 2$, and $B = 20$ for the current case study, as shown in Figure 4.4. Note that $Coverage \in [0, 1]$ and $Total\_cost \geq 0$, so the preference functions only need to be defined for the appropriate ranges. These simple preference functions are chosen for convenience in this case study, the same methodology can be easily applied with more complicated preference definitions.

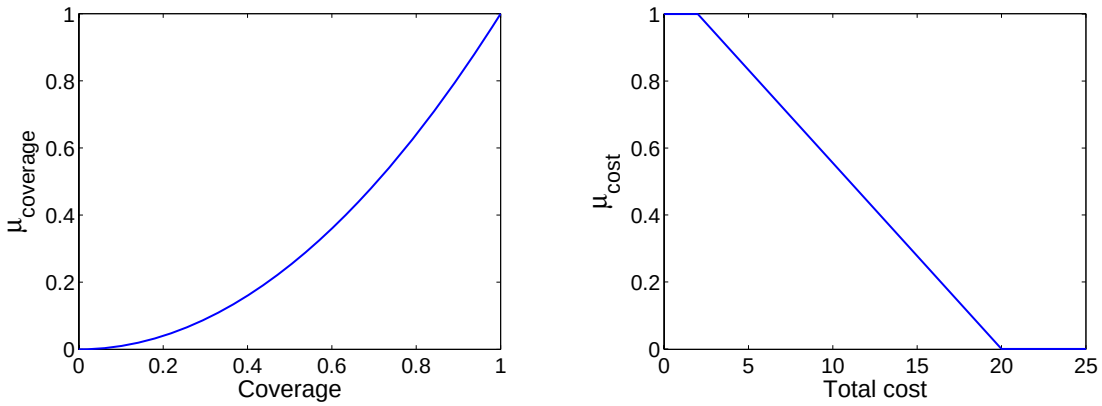


Figure 4.4: Designer's Fuzzy Preferences for *Coverage* and *Total_cost*

4.3.3.2 Aggregation

A common way to construct the fitness function with multiple criteria is to assign importance weights to each criterion, and then aggregate the weighted preferences into an overall preference. The best design will have the highest overall preference. All current multi-criteria decision makings ultimately rely on the aggregation of disparate preferences with *aggregation functions*. The axioms that an aggregation function should obey to insure rational design decision makings were presented in [Otto and Antonsson 1991]. It was also shown [Scott and Antonsson 1998] that there is a family of aggregation function operators $\mathcal{P}_s$ that spans an entire range of possible operators between *min* and *max*, of which the set $\{\mathcal{P}_s | s \leq 0\}$ has included all operators that satisfy the design axioms. The class of functional equations [Aczél 1966], known as quasi-linear weighted means, is given by

$$\mathcal{P}_s(\mu_1, \mu_2; w_1, w_2) = \left(\frac{w_1 \, {\mu_1}^s + w_2 \, {\mu_2}^s}{w_1 + w_2} \right)^{\frac{1}{s}} \tag{4.6}$$

Here, μ_1 and μ_2 are individual preferences on desired performance criteria. The parameter s establishes the *degree of compensation*, or the *trade-off strategy* adopted by the designer. Higher values of s indicate a greater willingness to allow high individual preference on one performance criterion to compensate for the other lower one. The parameters w_1 and w_2 are importance weights, and their ratio $w = \dfrac{w_2}{w_1}$ is sufficient to characterize the relative importance of the two performance criteria. The definition above is only for two attributes, but it is straightforward to be extended to cases involving more criteria.

It was also shown [Scott and Antonsson 1998] that

$$\mathcal{P}_{-\infty} = \lim_{s \to -\infty} \mathcal{P}_s = \min(\mu_1, \mu_2) \tag{4.7}$$

$$\mathcal{P}_0 = \lim_{s \to 0} \mathcal{P}_s = ({\mu_1}^{w_1} \, {\mu_2}^{w_2})^{\frac{1}{w_1 + w_2}} \tag{4.8}$$

$$\mathcal{P}_1 = \lim_{s\to 1} \mathcal{P}_s = \frac{w_1\,\mu_1 + w_2\,\mu_2}{w_1 + w_2} \tag{4.9}$$

$$\mathcal{P}_\infty = \lim_{s\to +\infty} \mathcal{P}_s = \max(\mu_1,\, \mu_2) \tag{4.10}$$

Therefore the *min* operator $\mathcal{P}_{-\infty}$ indicates no compensation at all among various criteria and the weighted geometric mean $\mathcal{P}_0$ represents the highest degree of compensation in design-appropriate (i.e., $s \leq 0$) aggregation functions. Note that the commonly used weighted sum $\mathcal{P}_1$ is just one special instance (where $s = 1$) of this whole family of aggregation functions, which, as well as the *max* operator, turns out to be an inappropriate aggregation function for rational engineering design [Otto and Antonsson 1991, Scott and Antonsson 1998]. It was also shown that any Pareto optimal point can be reached by a choice of some combination of the weight ratio w and design-appropriate trade-off strategy s.

Based on the above results, the fuzzy fitness function used in this case study is defined to be the overall preference aggregated as a generalized weighted mean of the individual preferences, and is given by

$$Fitness(w, s) = \left(\frac{\mu_{cost}^{\,s} + w \cdot \mu_{coverage}^{\,s}}{1 + w} \right)^{\frac{1}{s}} \tag{4.11}$$

where

$$w \equiv \frac{w_{coverage}}{w_{cost}} > 0 \text{ and } s \in [-\infty, 0].$$

The design goal here is to maximize the fitness of the sensory configuration designs by maximizing both design preferences, which, according to the preference curves shown in Figure 4.4, is equivalent to increasing the coverage of the detection zone while at the same time reducing the total cost of sensors. To get better coverage of the detection region, more sensors with wider cones of view and/or longer ranges are needed, which tends to increase the total cost of the sensing system. While the achievable coverage of the detection region depends, to an important degree, on the number and types of the sensors. So a trade-off has to be made between the two, and the key point is how to choose the weight ratio w and trade-off strategy s that lead to a desirable trade-off between the coverage and system cost under specific design requirements. Hence it is important not to arbitrarily limit the range of Pareto optimal points that can be selected by choosing a predetermined trade-off strategy. A method for establishing w and s for a given problem was presented in [Scott and Antonsson 2000].

Different Pareto optimal solutions for this case study can be easily obtained by setting the pair of weight ratio w and trade-off strategy s in Equation 4.11 and letting the evolutionary algorithm automatically synthesize solutions according to each fitness function efficiently [Antonsson et al. 2003]. Sample results are presented in Section 4.5.2. Then the design engineer will be able to learn what level of performance can be achieved under certain preference settings, along with the corresponding cost of the sensing system, even in an early stage of design. This will help guide the design decision to a desirable trade-off between various competing design objectives.

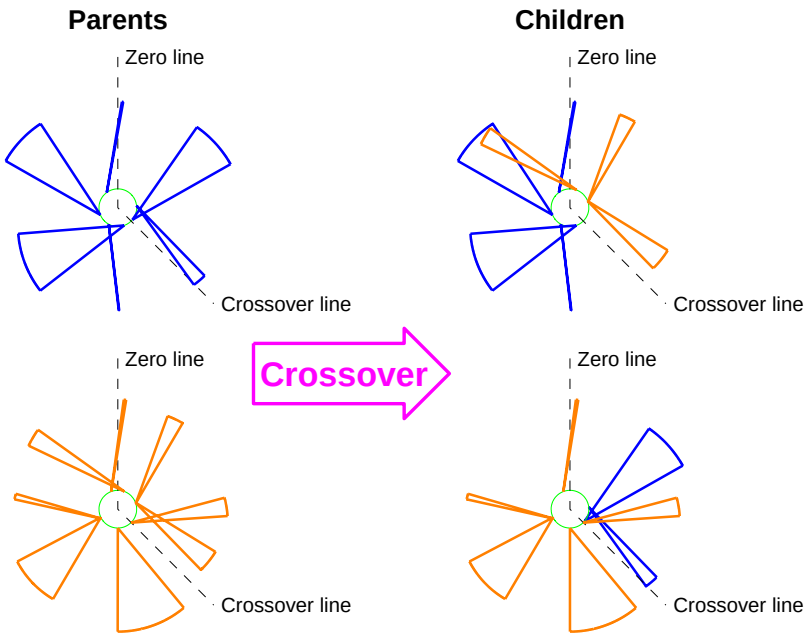


Figure 4.5: Illustration of the One-point Crossover Scheme for Two Sensory Systems with Different Numbers of Sensors: The sectors (or lines) represent the sensor scanning areas (or rays).

4.4 Evolutionary Experiments

The evolutionary design synthesis method presented in Section 2.3 is applied to generate the "best" sensor configurations under different conditions, i.e., different evaluation tests and different fitness functions with different values of w and s, which reflect the designer's different emphasis assigned to the two competing factors, $Coverage$ and $Total\_cost$, and how much higher preference values compensate for lower ones.

The evolutionary algorithm here uses a parent selection based on the roulette wheel scheme, an elitist generation selection, a one-point crossover, and a uniform mutation with insertion and deletion of individual sensors from the sensory system, as described in Section 2.4. Especially the one-point crossover can be further specified in the context of this particular sensory configuration case study, since each sensor can be considered as a module in the design variable vector that encodes the collective sensory system. The crossover line can be randomly chosen along the position angle φ (as shown in Figure 4.1) from its range $[0, 2\pi)$, i.e., randomly selecting a point along the periphery ring where the sensors are mounted and connecting with the vehicle center. Then the sensors of the parents between the crossover line and the zero line ($\varphi = 0$) are swapped in the crossover operation, as shown in Figure 4.5.

Table 4.2 summarizes the parameter values used in the evolutionary algorithm. The probabilities of genetic operators are fixed during evolutions and are defined for each design solution.

Parameter	Value
Population size	50
Selection scaling factor	2
$p_{crossover}$	0.2
$p_{mutation}$	0.182
$p_{insertion}$	0.05
$p_{deletion}$	0.05

Table 4.2: Evolutionary Algorithm Parameters

4.5 Results and Discussion

Systematic experiments are performed to verify the effectiveness of the evolutionary design synthesis method in terms of this sensor configuration design case study problem. First, different evaluation tests are used and their respective evolved solutions are cross-tested, and the most computationally efficient evaluation test is identified. Then different values are chosen for the parameters in the fitness function to generate different Pareto optimum design solutions.

4.5.1 Comparison of Different Evaluation Tests

Evolutionary runs based on the static, 1D quasi-static, and 1D full coverage tests are repeated 20 times using different random number generator seeds and terminated after 200 generations for each run; 2D quasi-static and 2D full coverage evolutionary runs are repeated 10 times and stopped after 200 generations; kinematic embodied evolutionary runs are repeated 5 times and stopped after 100 generations each. These values are selected upon consideration of the relative computational costs of the different levels of simulation (refer to Figure 4.3 and Table 4.1). The 1D/2D quasi-static and full coverage evaluation tests use the 1D/2D PDF's shown in Figure 3.5 and 3.4, respectively, and the traffic conditions in the kinematic embodied simulations are the same as those used to generate the PDF's.

The number of sensors is either evolved (variable) or preestablished (in this case, six sensors). For the evolutionary runs with a variable number of sensors, the initial population is randomly generated with sensory systems with a randomly chosen number of sensors from 1 to 20.

Note that the quasi-static and kinematic embodied tests are stochastic evaluation tests, so each candidate solution's fitness value during evolutions is based on not only one time performance but an aggregation of multiple *re-evaluations*, as mentioned in Section 2.4.5, where the number of evaluations depends on the number of generations a solution has survived. The *minimum* aggregation criterion is used here to search for robust solutions, and it has been verified that this scheme outperforms standard evolutionary algorithms when dealing with noisy fitness functions [Pugh et al. 2005].

Figure 4.6 (*Top*) shows a comparison of the performances achieved by the best design solutions

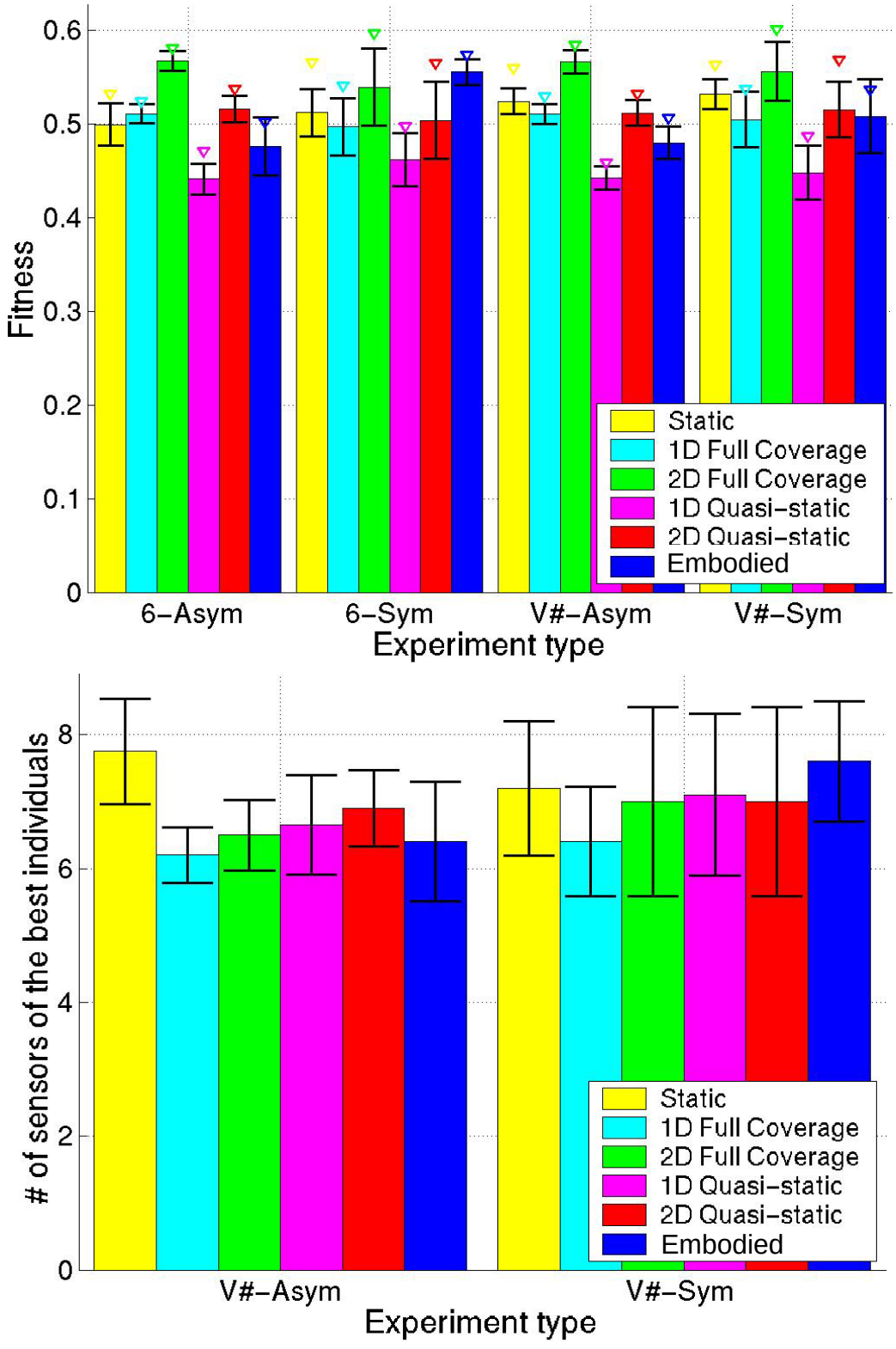


Figure 4.6: (*Top*): Performances of the Best Design Solutions Evolved under Different Conditions and Evaluation Tests with each Final Noise Test Conducted under the Same Evaluation Test Used in Evolution, respectively; (*Bottom*): Numbers of Sensors Used by the Best Design Solutions Evolved with Variable Number of Sensors

at the last generations of the evolutionary runs with different evaluation tests and under different conditions, i.e., forced symmetry (Sym) on the sensory configuration or not (Asym), fixed (6) or variable number (V#) of sensors. The performances shown are statistics of the best final fitness values obtained from the final noise tests conducted with the same evaluation tests used in the evolutions, respectively. Note that for the deterministic evaluation tests (i.e., static and full coverage tests), the final test is just a single evaluation test; while for stochastic evaluation tests (i.e., quasi-static and embodied tests), a final noise test contains 100 repeated evaluations for each individual and again the *worst* result over the 100 evaluation results is taken to be the final fitness value to search for the most robust solution.

In the histograms shown in Figure 4.6 and 4.7, the height of a column represents the average value, while error bars and triangular marks, respectively, correspond to the standard deviations and the maximum values over the best design solutions evolved with repeated evolutionary runs under the same type of evolutionary experiment.

Figure 4.6 (*Bottom*) shows the numbers of sensors used by the best design solutions evolved with variable number of sensors, which provide some hints of the optimal number of sensors needed under the specified conditions. This is also the reason why six is chosen to be the number of sensors for the cases with a fixed number of sensors.

Figure 4.7 shows the performances of the best design solutions evolved under different conditions and evaluation tests cross-checked under the *same* final evaluation test, i.e., the 2D full coverage test (*Top*) and the kinematic embodied test (*Bottom*), respectively.

It is interesting to notice that the two cross tests show similar trends for qualitative comparisons among the best results from different evolutions, except that the embodied test results are characterized with more variation. This is expected since the PDF used by the 2D full coverage test is generated from data accumulated over much longer embodied simulation time (5000 evaluation spans) than the embodied final noise test (100 evaluation spans), which in turn reduces the noise effect. The two tests are intrinsically quite different: one being deterministic with just a single evaluation test and the other being stochastic with 100 evaluation tests of expensive embodied simulations; hence the 2D full coverage test is a much more efficient test that could replace the embodied test itself in the final noise tests.

In addition, as expected, the static test is the simplest but has the worst performances in the cross tests shown in Figure 4.7 due to lack of traffic information, although a comparable level of performance is achieved in its "native" final test as shown in Figure 4.6. On the other hand, significant differences are barely observed among the best results achieved by the embodied evolution and other types of evolutions. In other words, the performances of the evolutionary design synthesis under the expensive embodied evaluation and those cheaper ones based on the traffic PDF's are very close under general design conditions. The results from 2D full coverage and quasi-static

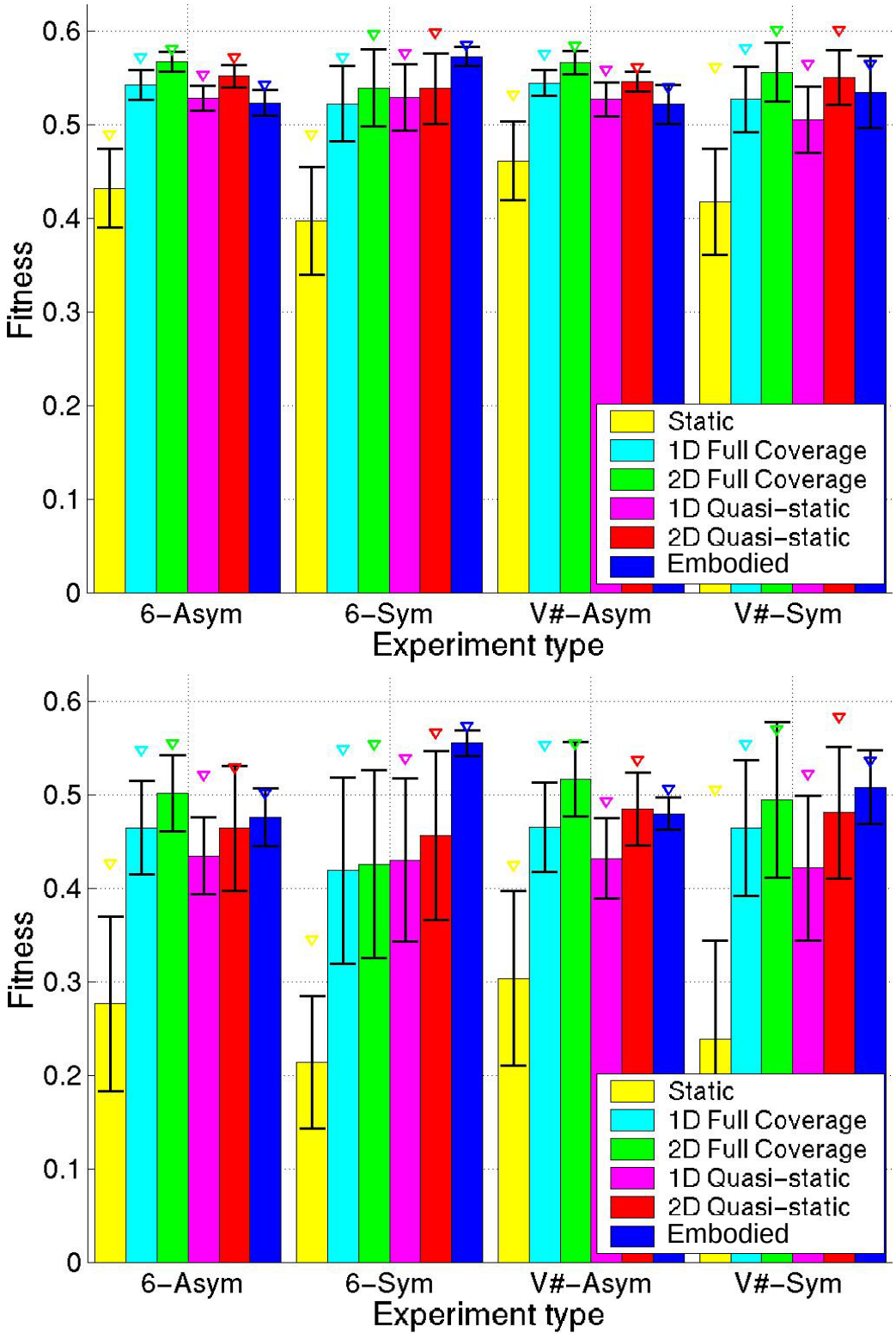


Figure 4.7: Performances of the Best Design Solutions Evolved under Different Conditions and Evaluation Tests with the Final Noise Tests Conducted under the 2D Full Coverage Evaluation Test (*Top*) and the Kinematic Embodied Evaluation Test (*Bottom*), respectively

evolutions are almost always comparable to, if not better than, those from the embodied evolutions, which suggests that the computationally expensive embodied test could be replaced by simpler and significantly faster evaluation tests during evolutions without compromising the performance of the design solutions synthesized. Moreover, the 1D/2D full coverage and quasi-static evolutions have achieved almost interchangeable performances in most cases, with the 2D cases slightly better than the 1D cases, although the PDF's are used quite differently in the two types of evaluation tests, as explained in Section 4.3.3.

Furthermore, Figure 4.6 and 4.7 also show that enforcing symmetry does not necessarily improve the performance achieved at the end of evolutions. Enforcing symmetry (and therefore reducing the search space to half) usually only shortens convergence time but does not lead to major improvement in performance of the evolved results, since 100 or 200 generations is long enough to synthesize good design solutions in asymmetric cases.

Moreover, major difference is not observed between the performances achieved in the fixed six-sensor cases and the corresponding variable number of sensors cases. Hence the exact optimal number of sensors need not be known beforehand, which can be left to the evolutionary algorithm to discover. Finally, it is also observed that a variety of good design solutions with different numbers of sensors can achieve nearly the same level of performance, providing multiple alternative solutions.

4.5.2 Evolving Engineering Design Trade-offs

In this section, the automatic design synthesis method described above is applied to generate the best sensor configurations with different values of the weight ratio w and trade-off strategy s in the fitness function (Equation 4.11), which reflect the designer's different emphasis assigned to the two competing factors, $Coverage$ and $Total\_cost$, and how much higher preference values compensate for lower ones.

Since the evolutions under 2D full coverage and quasi-static tests can generate design solutions of equivalent, if not better, quality as those under the embodied test, as shown in Section 4.5.1, results exclusively gathered with a 2D full coverage test are presented in this section for simplicity and computational efficiency, because the 2D full coverage test is a deterministic implementation about 60 times faster than the embodied simulation, as shown in Table 4.1.

The evolutionary runs are conducted with the 2D full coverage evaluation test based on the 2D traffic PDF shown in Figure 3.4. For simplicity, the sensor configurations are forced to have "modified" left-right symmetry in the evolutions, i.e., the sensors are generally left-right symmetric except that those lying close to the symmetry axis are mirrored to the opposite end, as illustrated in Figure 4.8, 4.9, and 4.10, conforming to the traffic PDF used. For each different experiment, evolutionary runs are repeated 10 times with different random number generator seeds and terminated after 200 generations for each run. Each initial population is randomly generated with sensory

systems with a randomly chosen number of sensors from 2 to 20, and the final optimal number of sensors is determined by the evolutionary algorithm.

Figure 4.8, 4.9 and 4.10 show the results obtained from the evolutionary design synthesis experiments under three different fitness function settings, respectively. The top plots show the evolution processes of the mean of the population $Fitness$, as well as the two individual preferences, $\mu_{coverage}$ and μ_{cost}, over 200 generations; while the bottom plots show the corresponding best sensor configurations evolved under each specific condition with their respective values of $Coverage$ and $Total\_cost$.

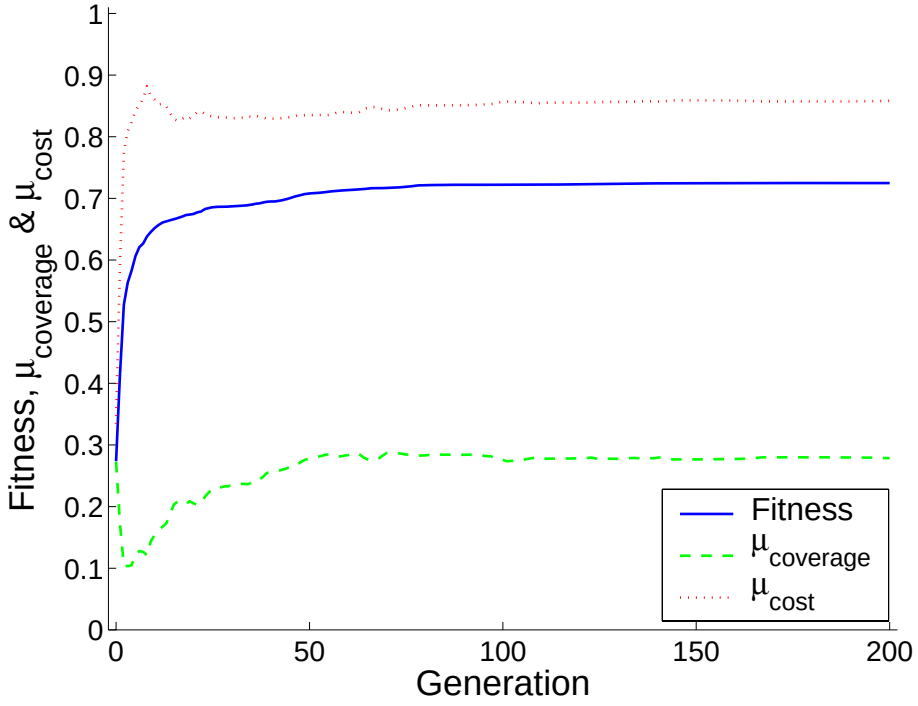
Figure 4.8 shows the evolutionary result of an experiment with the weight ratio $w = \frac{3}{17}$ and the degree of compensation $s = 0$ in Equation 4.11, which indicates that reducing cost is considered to be relatively more important than increasing coverage, and that the higher individual preference (μ_{cost}) can compensate for the lower one ($\mu_{coverage}$). As a result, a simple and inexpensive sensory system of four sensors with low cost and low coverage is selected to cover only the regions with apparently high vehicle occurrence probability, as shown in Figure 3.4.

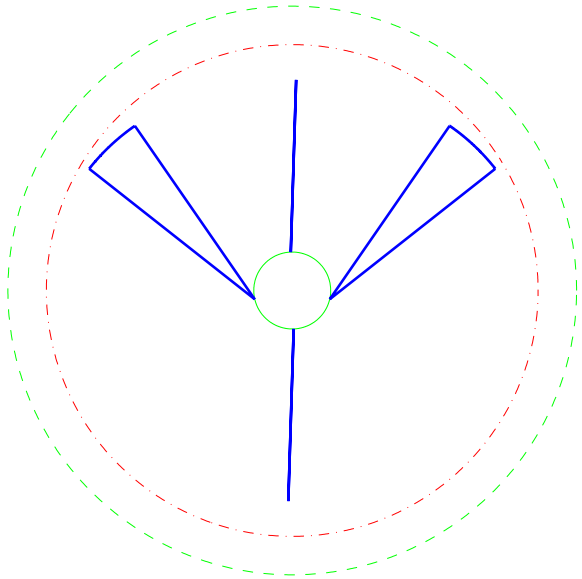
On the contrary, Figure 4.10 shows the result of an experiment with the weight ratio $w = 4$ and the degree of compensation $s = 0$ in Equation 4.11, which means that the designer emphasizes on obtaining better coverage rather than reducing cost, and that the same trade-off strategy is adopted with opposite effects, i.e., the higher individual preference ($\mu_{coverage}$) can compensate for the lower one (μ_{cost}). Consequently, a rather complex and expensive sensory system with eight sensors is evolved to cover most areas of the detection region with a coverage rate of 98%.

Finally, Figure 4.9 shows the result of a special case with the degree of compensation $s = -\infty$, which means that the minimum of the two individual preferences is taken to be the overall preference regardless of their relative weights, i.e., a non-compensating trade-off strategy is adopted. A sensory system of medium cost and coverage is selected by the evolutionary algorithm in this case.

As expected, the evolutionary engineering design synthesis methodology selects considerably different design solutions under different choices of fitness function parameters. More experiments based on different combinations of w and s are performed and the set of the final best trade-offs reached by the evolutionary algorithm constitutes an approximate feasible Pareto optimal frontier for this design problem, which is shown in Figure 4.11. The top graph illustrates the Pareto frontier by plotting the $Coverage$ versus $Total\_cost$ of the best sensory configurations evolved with different fitness function parameters, while the bottom graph shows the same Pareto frontier depicted in terms of the fuzzy preferences $\mu_{coverage}$ and μ_{cost}. Each data point represents the best result of one particular type of evolutionary experiments under a given combination of w and s in Equation 4.11.

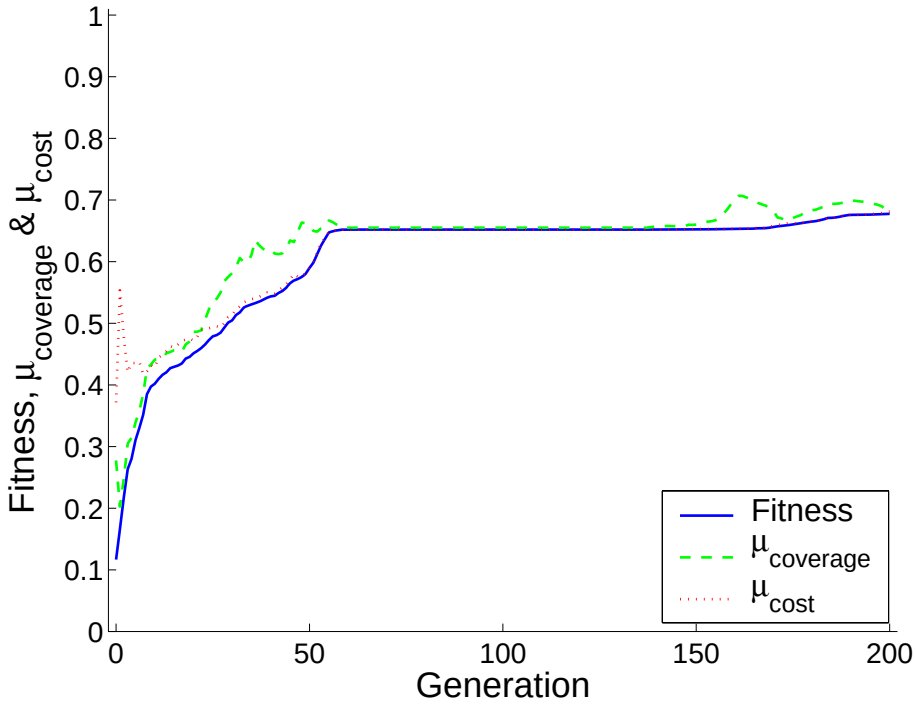
The top graph of Figure 4.11 quantitatively outlines the general extent of the achievable coverage at various levels of cost: The coverage increases as the cost increases, and the rate of coverage increase lessens when the coverage approaches its upper bound 1, which agrees with common sense. It is desirable to maximize both performance measures, i.e., reach the utopia point at the upper-right

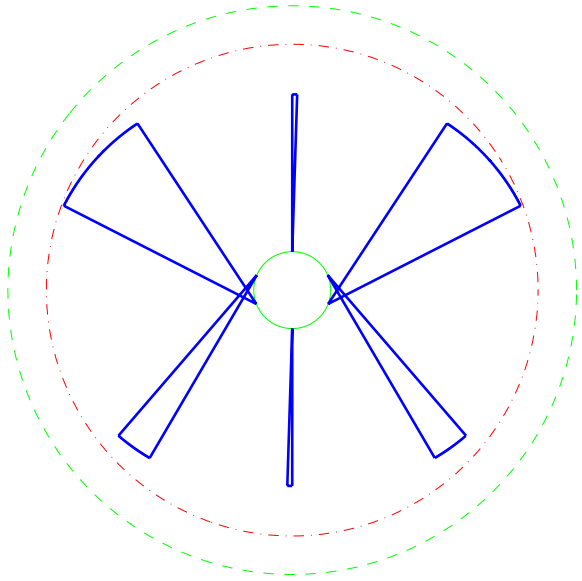




$Coverage = 53\%, Total\_cost = 4.6$

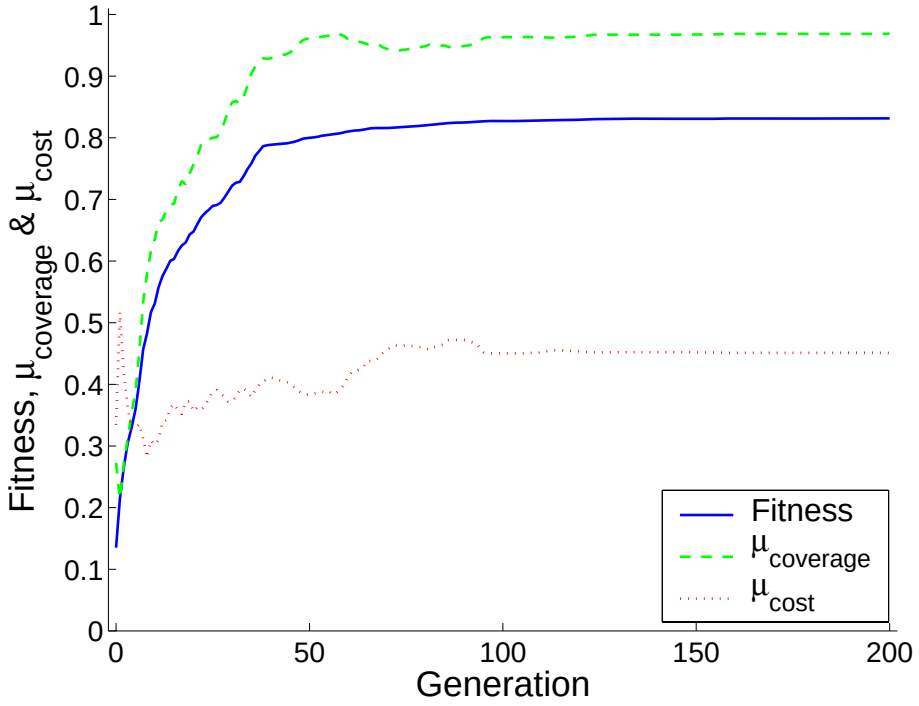
Figure 4.8: Evolution Process of the Population Mean Fitness and Preferences (*Top*) and the Best Sensor Configuration Evolved (*Bottom*) with $s = 0$ and $w = \frac{3}{17}$

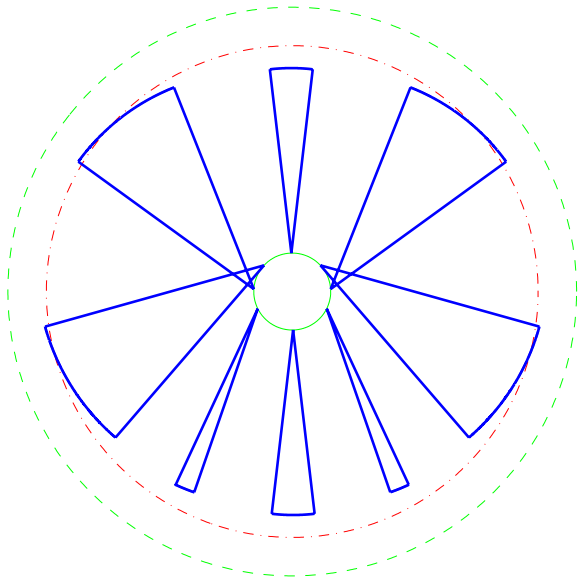




$Coverage = 82\%, Total\_cost = 7.7$

Figure 4.9: Evolution Process of the Population Mean Fitness and Preferences (*Top*) and the Best Sensor Configuration Evolved (*Bottom*) with $s = -\infty$ and arbitrary w (i.e., $Fitness = \min(\mu_{coverage}, \mu_{cost})$)





$Coverage = 98\%, Total\_cost = 11.9$

Figure 4.10: Evolution Process of the Population Mean Fitness and Preferences (*Top*) and the Best Sensor Configuration Evolved (*Bottom*) with $s = 0$ and $w = 4$

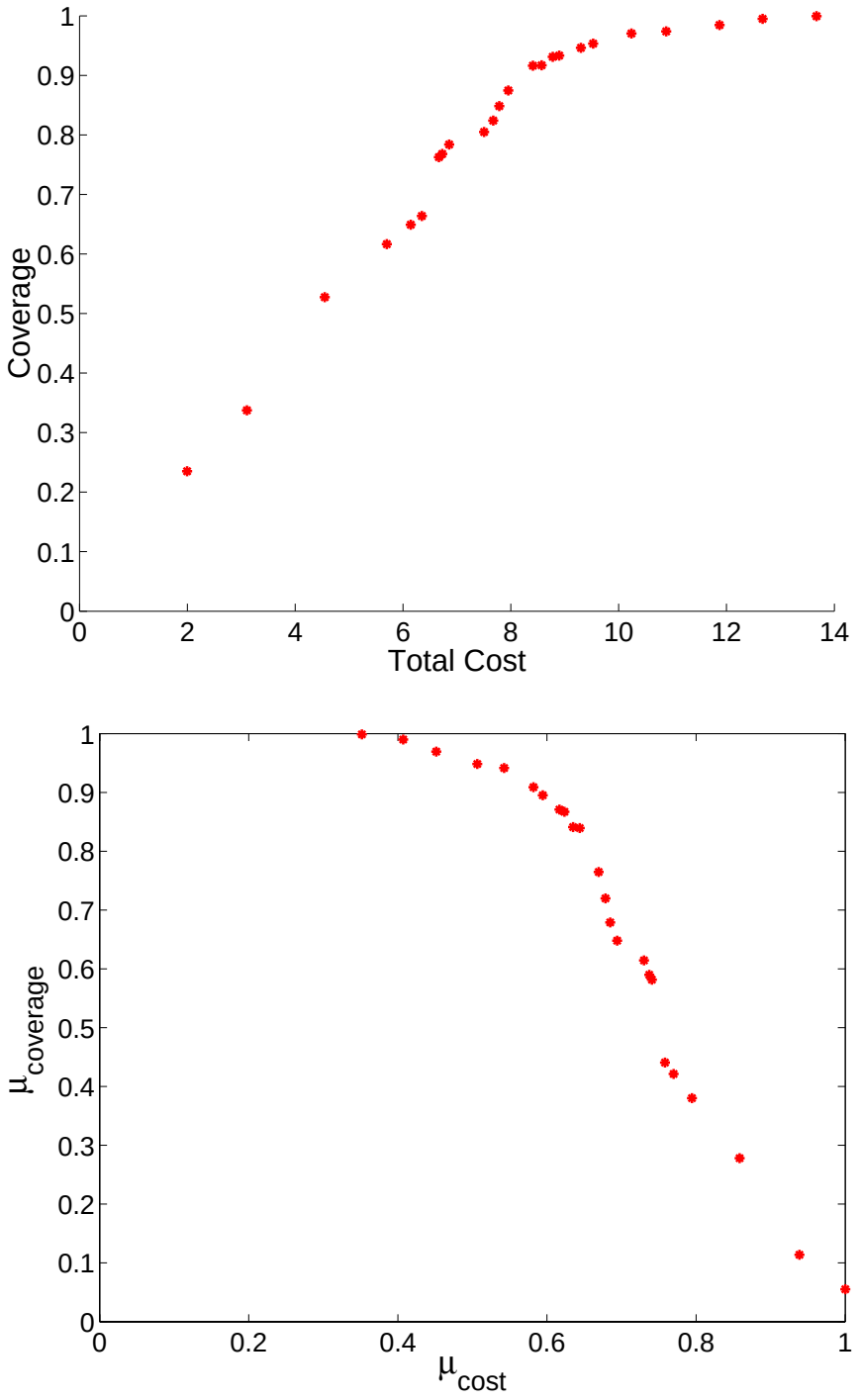


Figure 4.11: Evolved Pareto Frontier for the Design Trade-offs Present in the Case Study

corner of the bottom graph of Figure 4.11, which is, however, impossible to be achieved. Therefore a trade-off has to be quantitatively established with an appropriate ratio of the relative importance (w) and degree of compensation (s) between the two performance measures.

This result can be helpful for engineers in the design decision-making process. With the automatic design synthesis method proposed, these results can be obtained with minimum human engineering effort and a modest computational cost.[3] Although the best solutions found by the algorithm do not necessarily represent the optimal solutions under the specified conditions, they can quickly provide the design engineers with a general idea of various novel, promising configurations in the early stage of design.

Finally, the other parameters in the fitness function (Equation 4.1, 4.4, 4.5, and 4.11) and the evolutionary algorithm (Table 4.2) can also be changed by the design engineers to investigate their respective influences on the final results evolved and try to advance toward the desired design goals.

4.6 Conclusion

A first case study on effective design synthesis of sensory configurations for intelligent vehicles under various design requirements is presented, with special emphasis on addressing the modern engineering design challenges mentioned in Section 1.2. The engineering design synthesis method presented in Section 2.3 is applied to the case study problem and novel design solutions are synthesized automatically by the evolutionary algorithm.

The candidate design solutions are evaluated under several different levels of deterministic and stochastic simulations of traffic scenarios in the evolutionary process. The results indicate that noisy and time-consuming, but more realistic embodied simulations can be apparently replaced by more abstract and computationally more efficient evaluation tests without compromising the quality of the final evolved results in terms of this case study problem.

Different engineering design trade-offs are automatically synthesized utilizing fuzzy fitness functions with different importance weighting ratios and trade-off strategies selected for multiple performance measures, which can provide useful information to assist design engineers in the design decision-making process.

The experimental results presented in this chapter show that the proposed evolutionary design synthesis method can be efficiently applied to deal with the engineering design challenges appropriately, and that it appears to be a promising approach for more complex engineering design synthesis problems.

[3] An evolution of 200 generations with the 2D full coverage test and a population of 50 individuals requires about 22 minutes of computational time on a computer with 1.5 GHz AMD CPU.

Chapter 5

Evolution of Neural Controllers

In this chapter, the evolutionary design synthesis methodology presented in Section 2.3 is applied to evolve neural controllers for a special class of intelligent vehicles, i.e., miniature autonomous mobile robots. Both feed-forward and recurrent neural networks can be evolved with fixed or variable network topologies. The efficacy of the evolutionary methodology is demonstrated again in the framework of two case studies on collective robotic inspection of regular structures as well as simple driver behavior modeling, respectively, where the vehicles (robots) are only equipped with limited local on-board sensing and actuating capabilities.

The neural controllers generated during evolutions are evaluated in a sensor-based kinematic embodied simulation environment with realistic noise, as introduced in Section 3.2.2. If the embodied simulator is faithful enough for the target hardware platform, evolved controllers can be easily transfered to real robots [Miglino et al. 1995]. Homogeneity of the robot team is enforced here to limit the search space, achieve scalability, and bypass the credit assignment problem typically arising in distributed systems consisting of individuals using only local information [Hayes et al. 2003, Versino and Gambardella 1997].

The performances of the evolved neural controllers are compared with that of a hand-coded rule-based controller in terms of the inspection case study under the same conditions [Zhang et al. 2006]. It will be shown that the evolutionary algorithm appears powerful and promising for automatic synthesis of novel neural controllers, requiring little prior domain knowledge or neural network structural information. The evolved solutions can serve as good starting points for further study and optimization by human engineers.

5.1 Background

Miniature autonomous mobile robots share important characteristics with simple biological systems: robustness, simplicity, small size, flexibility, and modularity. Each individual is rather simple with limited local sensing and actuating capabilities, while as a group they can accomplish diffi-

cult global tasks in dynamic environments, without any external guidance or centralized control [Bonabeau et al. 1999].

Design and control of such an intelligent vehicle (robot) swarm are difficult mainly because their group behavior is an emergent property of their mutual interaction and their interaction with the environment. The robot swarm becomes a distributed dynamical system due to independent parallel actions of different individuals [Martinoli 1999]. Since the robots only have partial perceptions based on crude and noisy sensors, limited computational capabilities and energy budget, managing the robots to solve a global task under such constraints presents significant technical challenges. This is especially true because human intelligence is specialized in individuals and centralized control, instead of the collective intelligence shown in nature.

Evolutionary robotics [Nolfi and Floreano 2000] is a new and promising technique for automatic design synthesis of control strategies for autonomous robots in a distributed control system, especially for miniature robots. Inspired by nature, evolutionary robotics makes use of tools such as neural networks and evolutionary computation algorithms.

Inspired by biological neural networks, Artificial Neural Networks (ANN) have been a powerful computational tool widely applied in science and engineering [Hertz et al. 1991]. They are often used to implement robot controllers because of their light computational requirements and nonlinear basic elements, properties that allow for real-time control and, potentially, modular implementation of complex perception-to-action functions. ANN can be designed and trained using various methods, including those based on evolutionary computation [Yao 1999, Nolfi and Parisi 2002]. As opposed to optimization of behavior-based controllers, the key feature of ANN evolution is that, the genotypical searching space is less constrained by ANN models and the resulting phenotypical solution directly shapes the robot behavior as a whole.

Evolutionary computation algorithms, as reviewed in Chapter 2, have gained considerable popularity as effective tools for searching vast, complex, deceptive, and multi-modal search spaces with little domain-specific knowledge. In recent years, they have found natural applications in the automatic synthesis of artificial neural network controllers for intelligent agents [Patel et al. 2001]. Evolutionary algorithms allow co-evolution of the network architectures as well as the connection weights within task-specific design constraints. As stochastic optimization methods, evolutionary algorithms are good at working in noisy environments to search for robust solutions, and can easily adapt to collective robotic tasks.

5.2 Evolution of Artificial Neural Networks

Based on the same general evolutionary optimization loop presented in Section 2.3 and applied in Chapter 4, the basic evolutionary loop used for automatic neural network controller synthesis

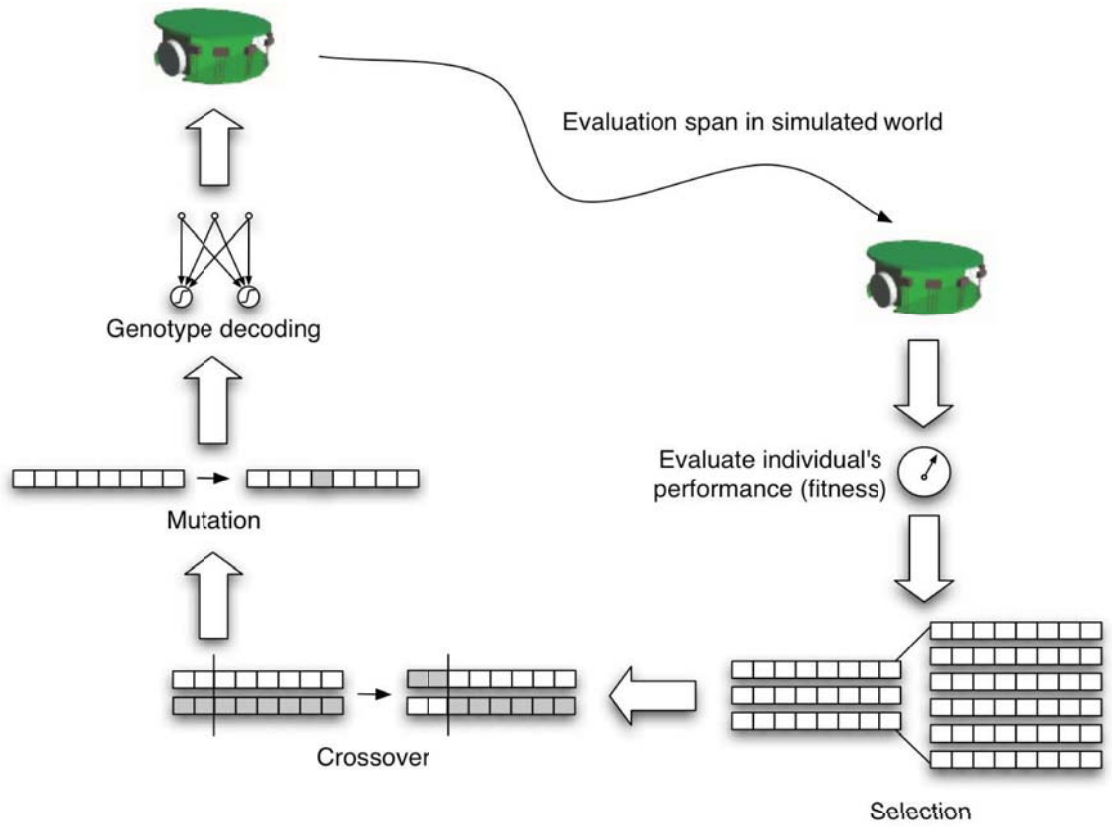


Figure 5.1: Evolutionary Run for Automatic Robotic Neural Network Controller Synthesis

is shown in Figure 5.1. Evolutions of both feed-forward and recurrent neural network controllers are performed, using real-valued vectors to encode synaptic weights with variable ANN structures, traditional roulette wheel parent selection with fitness scaling, elitist generation selection, and both crossover and mutation genetic operations. Only synaptic weights are evolved if the ANN topology is predefined, otherwise the network structure and synaptic weights are simultaneously evolved.

5.2.1 Encoding

The ANN synaptic weights are directly encoded as a sequential vector of real numbers. The vector length is fixed if the network structure is *a priori* determined, where a fully connected ANN is usually assumed and the fixed vector length can be computed as follows:

$$n_c = \begin{cases} (1+n_i)n_o & \text{if } n_h = 0 \text{ \& feed-forward} \\ (1+n_i)n_h + (1+n_h)n_o & \text{if } n_h > 0 \text{ \& feed-forward} \\ (1+n_i+n_o)n_o & \text{if } n_h = 0 \text{ \& recurrent} \\ (1+n_i+n_h)n_h + (1+n_h+n_o)n_o & \text{if } n_h > 0 \text{ \& recurrent} \end{cases} \tag{5.1}$$

where n_c, n_i, n_h, n_o and "1" represent the numbers of fully connected weights, inputs, hidden neurons, outputs, and biases, respectively.

When the ANN structure is also evolved, n_h becomes a design variable to be optimized. So the design vector length must also be variable to accommodate the variable ANN structure and evolve solutions of suitable complexity. To give the algorithm more freedom to search for the appropriate network structures, no restrictions are imposed on the number of permissible connections and the variable vector length is computed as follows:

$$n_c = \begin{cases} (1+n_i)n_h + (1+n_i+n_h)n_o & \text{if feed-forward} \\ (1+n_i+n_h+n_o)(n_h+n_o) & \text{if recurrent} \end{cases} \tag{5.2}$$

where n_c represents the *maximum possible* number of connections, but not all of them must be active. A non-zero real value in the genotype vector represents the weight value of an active connection, while zero values represent inactive (non-existent) connections. Note that the $n_h = 0$ cases of Equation 5.1 are also included in Equation 5.2.

5.2.2 Initialization

The population is randomly initialized at the beginning of an evolutionary run. For fixed network structure cases, all the genotype vectors are of the same length with synaptic weight values randomly drawn from $[-1, 1]$. For variable network structure cases, first n_h is randomly selected from 1 to 10[1] for each individual, then the genotype vector length is computed by Equation 5.2, and each real number in the genotype vector is set to 0 (inactive) or a random value between -1 and 1 (active) with probability 50%.

The networks initialized this way might contain some useless hidden neurons that do not contribute to the outputs at all. To improve the algorithm efficiency, they are identified and removed from the network by a simple routine after initialization to make the network more concise and relevant.

5.2.3 Genetic Operations

As mentioned above, crossover and mutation are both used in the evolutionary design synthesis process here.

In fixed ANN structure cases, standard crossover operators such as one-point or uniform crossover can be directly applied to two real vectors of equal length. In variable ANN structure cases, the crossover must operate on two vectors of different lengths, which represent two distinct network structures.

[1] There is, however, no upper limit for n_h during the evolution.

It is known that evolutions relying on crossover do not perform well in ANN topology optimization because of their intrinsic disruption feature and the permutation problem [Yao 1999]. To protect possible modules in the network, crossover points can not be arbitrarily chosen along the whole genotype vector. Following the scheme mentioned in Section 2.4.3 and illustrated in Figure 2.2, the genotype vector of ANN can be grouped into modules or blocks according to the connections with hidden neurons. For example, "Module 0" could represent all the connections directly from inputs to outputs (i.e., not connecting any hidden neurons), "Module 1" could consist of connections to and from hidden neuron #1, "Module 2" correspond to those of hidden neuron #2, and so on. The one-point crossover implemented here only allows interchange of corresponding modules between the parents. In practice, the algorithm tries to match the sequence of the hidden neuron modules as much as possible before crossover to reduce influence of the permutation problem. Then, from all possible crossover points, as shown in Figure 2.2, a random crossover point is selected for both parents and their modules below the crossover point are exchanged to create two new offspring networks. For the example shown in Figure 2.2, two parents networks of 6 and 3 hidden neurons produce two new networks of 4 and 5 hidden neurons, respectively.

Mutation is also a powerful tool for creating new network structures as well as modifying synaptic weights. In fixed structure cases, only Gaussian mutation is used to change values of the synaptic weights by a small amount randomly drawn from a Gaussian (normal) distribution. In variable structure cases, two extra types of mutations are introduced. First, a hidden neuron could be added to or removed from the current network configuration. Second, a connection between any two neurons could be turned on or off by switching between non-zero and zero values. When a hidden neuron is added or a connection is switched on, the synaptic weight values are initialized as described in Section 5.2.2.

The crossover and mutation operations could also introduce useless hidden neurons, which are identified and removed from the network before evaluation. In addition, identical individual networks might be generated as a result of these genetic operations. If identical copies are allowed to exist in the population, the power of crossover and pool diversity are reduced, which might cause pre-convergence of the evolution. Therefore each new individual is compared to all other individuals in the population, and any redundant copies are removed before the evaluation step.

5.3 Case Study 1: Collective Robotic Inspection

This case study is concerned with the automatic synthesis of ANN-based robotic control algorithms to enable autonomous inspection of regular structures using a homogeneous robot swarm. The goal is to design relatively simple "local" control strategies for individual robots such that their emergent group behavior can accomplish the complex global task of cooperative inspection. Sensor uncer-

tainty and vehicle position uncertainty should be taken into account when planning the individual movements that carry out the integral motion plan, i.e., the collective effect of the multi-vehicle platform as a whole.

5.3.1 Application Background

Autonomous robots find a wide variety of applications in the real world to release humans from various chores, especially when the working environment is hazardous or not easily accessible by humans. For instance, inspection of human-occupied space transportation systems and platforms is currently heavily labor-intensive, costly and time-consuming. Another example could be the inspection of propulsion systems (such as jet turbines), which usually requires full engine break-down and human experts' visual inspection using borescopes, a process which is both time-consuming and cost-intensive [Martin and Stewart 2000].

Therefore it is desirable to have the inspection task performed autonomously by a swarm of miniature mobile robots in these situations [Wong and Litt 2004], where robustness and fault tolerance are expected to be achieved through redundancy in task handling. In addition, the relatively simple agent controllers demand minimal computational capability, which in turn allows for greater miniaturization of the robotic agents. This idea is intellectually appealing and it could find broader applications for general inspection of engineered or natural structures.

5.3.2 Problem Formulation

A simple 2D scenario is considered, where the objects to be inspected have regular cylindrical shapes with a diameter of 20 cm and they are separated from each other by a distance of about 60 cm, as shown in Figure 5.2. The kinematic vehicle model used here is again based on the miniature Khepera robot with a diameter of 5.5 cm, as introduced in Section 3.2.2 and shown in Figure 3.3. A continuous repetitive world without boundaries can be simulated by placing the robots to the corresponding opposite side when they move out of one of the four sides (i.e., wrap-around world).

It is assumed that completely circumnavigating an object is a good emulation of the scanning-for-flaws maneuver. Thus the collective performance measure for this case study depends on the ratio of the inspected object surfaces over a pre-specified time span to all that needed to be inspected in the world. The maximum performance "1" can be achieved by complete coverage, i.e., fully inspecting all distinct objects in the world, within the time limit. Note that only 12 distinct objects are present in the repetitive world shown in Figure 5.2 under the wrap-around condition.

The robots are equipped with eight distance sensors with extended virtual sensor range of 10 cm, as shown in Figure 3.3 (b). The sensors are assumed to be line sensors characterized by a linear response with the distance: The closer the sensed object the higher the value. Sensor values are

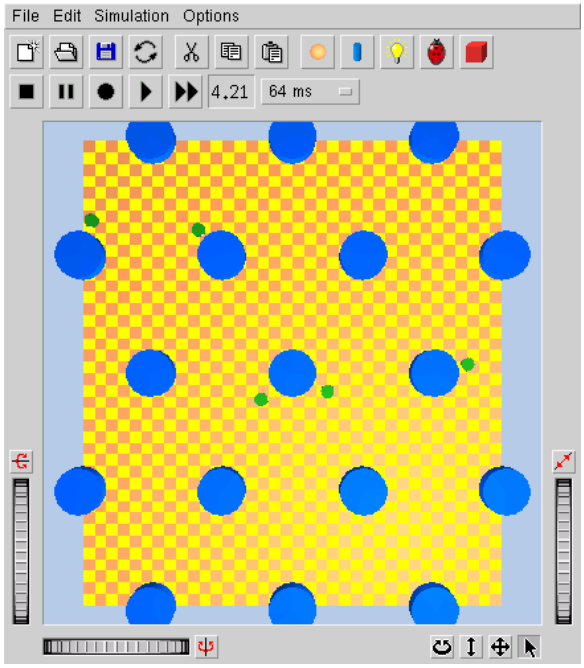


Figure 5.2: Top View of the Structure Inspection Experiment Setup in the Kinematic Embodied Simulator (Webots): The bigger blue disks represent the cylindrical objects to be inspected while the smaller green dots are the miniature robots.

integers ranging from $[0, 1023]$ with ± 10 white noise, and random integer values drawn from $[0, 10]$ are returned when nothing is sensed. These crude and noisy sensor measurements, normalized to $[0, 1]$, serve as the only inputs to the ANN controller to be evolved. Information received from teammates could also be added in the multi-agent scenario in the future.

Both hidden (if any) and output neurons use sigmoid output functions producing outputs in the range of $[0, 1]$. The ANN has two outputs mapping to the two wheel speeds[2] of the robots, taking integer values from $[-20, 20]$, with each speed unit representing 8 mm/s in the real world. Wheel slippage is also simulated with $\pm 10\%$ white noise on wheel speeds at each simulation step.

The geometric dimensions in this case study are chosen such that the robot can detect at most one object structure (besides any teammates) at any time, and there is a good chance that it might detect nothing at all. The inspection task requires the robot to approach an object structure, inspect it with minimal redundancy and maximum speed, leave it, and search for other objects. In collective scenario the robot also needs to distinguish its mobile teammates from static objects to be inspected and avoid teammates. Although one could implement a heuristic hand-coded rule-based controller (see Section 5.3.3), this is nevertheless a non-trivial task for a neural controller reading eight crude distance sensors and controlling two imprecise motor wheels directly. Indeed, the controller must not only evolve basic reactive behaviors such as object search, object inspection (i.e., follow the

[2]Either the left and right speeds, or the forward and rotation speeds.

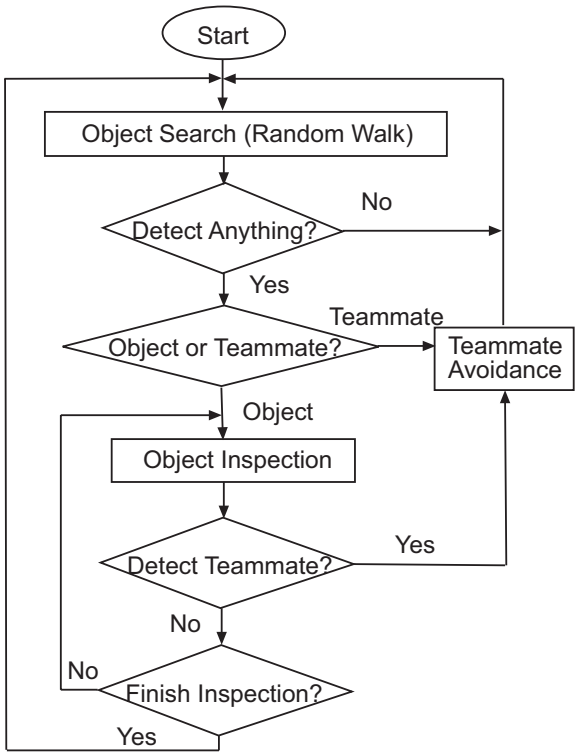


Figure 5.3: The Logic Scheme of the Hand-coded Rule-based Controller for Structure Inspection

contour of the object), and teammate avoidance, but also correctly *sequence* them: for instance, switching from object search to inspection when an object is found, and searching for new objects after finishing inspecting an object.

5.3.3 Hand-coded Controller

In order to create a baseline of the achievable level of performance in this specific case with a traditional engineering design method based on human intelligence, a simple hand-coded controller based on logical rules is implemented for the same task as described in last section. It exploits exactly the same sensor inputs and controls the same motor outputs as the evolved ANN controller solutions, and can be used to evaluate and compare with the evolved ones. As shown in Figure 5.3, this basic hand-coded controller is only based on the robot's crude distance sensor inputs and some internal timers. There is one key parameter that controls how long the robot keeps inspecting an object. This parameter depends on the circumference of the objects to be inspected[3] and is manually tuned by systematic search in order to get the best performance. Except for that, the robot has no other information about the objects (e.g., locations) and no memory of the past, and there is no communication between teammates.

Although it is rather straightforward to implement such a simple hand-coded controller for the

[3]It is assumed here that all objects to be inspected have the same shape and size, as shown in Figure 5.2.

ANN Type Description	Symbol	Pool Size
feed-forward without any hidden neurons	ffnh	50
recurrent without any hidden neurons	rcnh	50
feed-forward with n_h^* hidden neurons	ffvh	100
recurrent with n_h^* hidden neurons	rcvh	150

*Note that $n_h \geq 0$ is a variable number.

Table 5.1: Different ANN Types Considered in the Evolutionary Design Synthesis

specific scenario defined in Section 5.3.2, it is not obvious how to complete the same task with a structural ANN controller. Moreover, it becomes more intractable, even infeasible, to implement such a hand-coded controller for more complex (e.g., inspection of 3D, irregular structures) scenarios without raising the complexity and requirement for each agent significantly, where evolutionary algorithms might be more appealing to find a suitable trade-off.

On the other hand, more complex behavior-based controllers have been developed especially for the turbine blade inspection case study [Correll and Martinoli 2004, Correll and Martinoli 2005], which explore the special geometric properties of the turbine blades as well as information received from teammates, etc. In addition, a theoretic coverage algorithm based on graph representation was presented for complete inspection of various convex objects in a 2D test environment [Easton and Burdick 2005], where complete object geometry and location information with perfect omnidirectional sensor models are assumed. It is expected that these controllers would achieve better performance under specific conditions, but they also have higher requirements on each robot (e.g., more memory, better sensors, and extra communication modules), need more known information of the environment to be inspected, and utilize more complex and more specialized hardware and control software.

5.3.4 Results

The evolutionary design synthesis method presented in Section 5.2 is applied to evolve ANN controllers under different configuration settings: feed-forward or recurrent networks with or without a variable number of hidden neurons, as shown in Table 5.1. The population sizes of the evolutions depend on the dimension of the genotype vector to be optimized. The first two types of neural networks are of simplest fixed topologies with shortest vector lengths and smaller pool sizes. The latter two types of neural networks are of variable structures with longer vector lengths on average and larger pool sizes.

For each type of ANN controller synthesis, a series of evolutionary experiments are conducted with different output speed maps and different coefficients in the sigmoid neuron output function. For each evolutionary experiment, 5 evolutionary runs with different random seeds are executed,

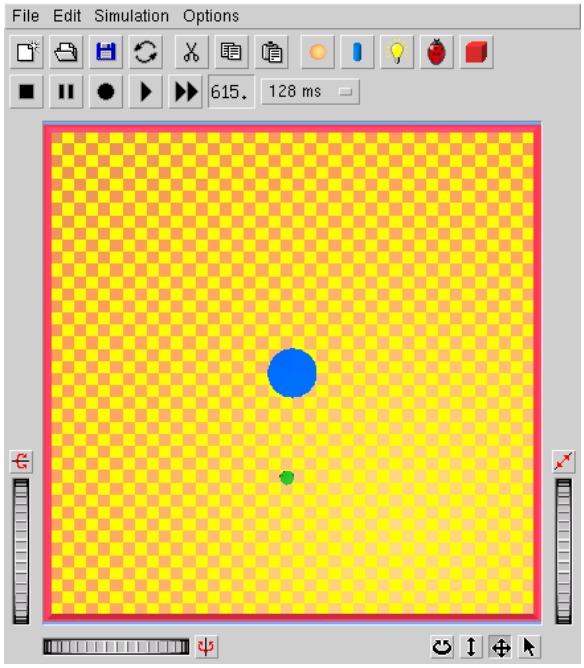


Figure 5.4: Screen Shot of the SRSO Scenario

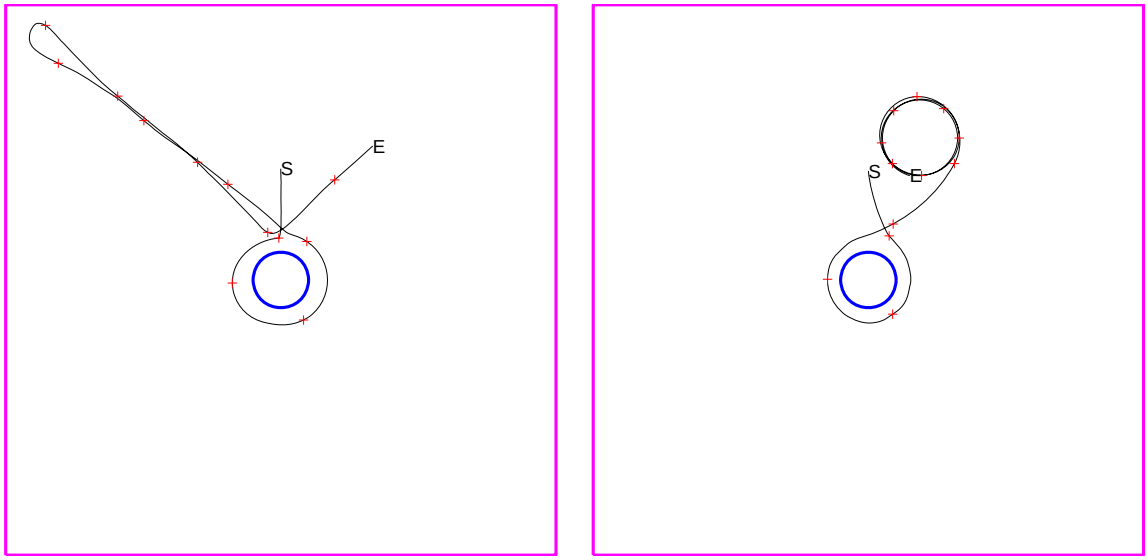
each lasting for 200 generations.

As mentioned above, the fitness evaluation of candidate controller solutions in the kinematic embodied simulation environment is characterized with noise, due to sensor and actuator noise (e.g., wheel slip) introduced as well as random initial conditions for each evaluation span. However, only one single evaluation is performed for each individual during the evolutions to save computational time, because the evaluations here are significantly more computationally expensive than the genetic operations and the variation of fitness value of a given solution is small compare to population fitness variance. In this case single evaluation (i.e., smallest sample size) appears to be a computationally effective strategy [Fitzpatrick and Grefenstette 1988, Miller and Goldberg 1996]. Again a final noise test of 100 evaluations is performed for each ANN controller in the final population of each evolutionary run as well as the hand-coded controller to get fair performance comparisons of different controllers. Different aggregation criteria (minimum, geometric mean, average, etc.) can be used as a measure to estimate the overall performance of each controller from its 100 fitness sample values.

In the following sections, the inspection task is approached by three systematic steps: the single robot single object scenario (Section 5.3.4.1), the single robot multiple objects scenario (Section 5.3.4.2), and the multiple robots multiple objects scenario (Section 5.3.4.3).

5.3.4.1 Single Robot Single Object (SRSO) Scenario

The global problem of collective robotic inspection of multiple objects can be decomposed to the microscopic interaction between a single robot and a single object, as shown in Figure 5.4. Here the



(a) (b)

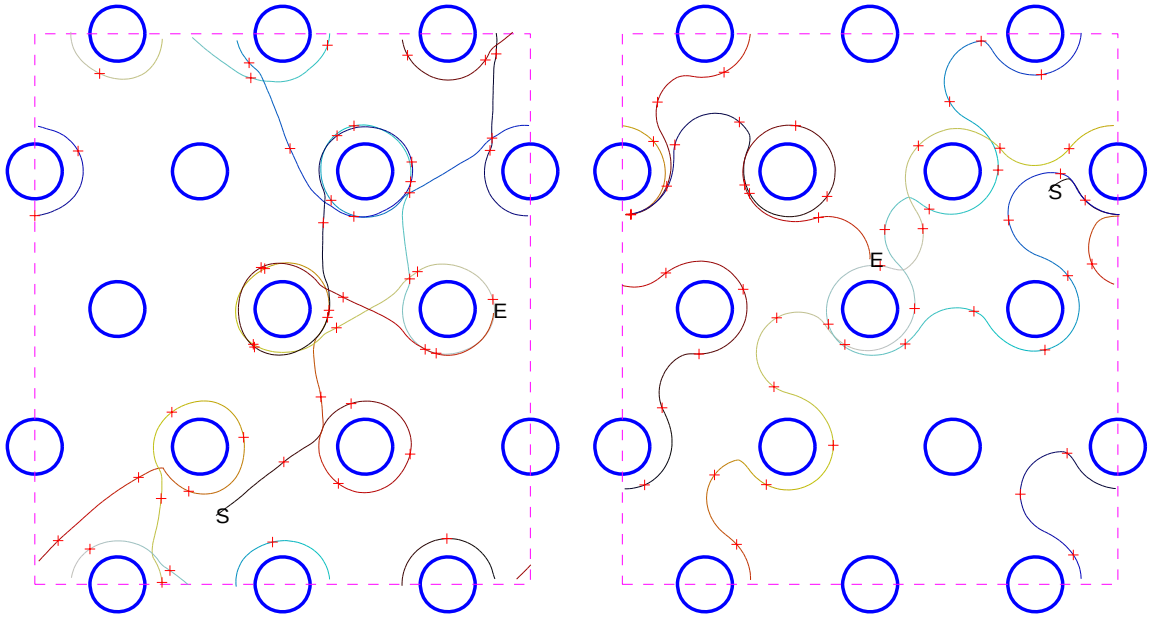
Figure 5.5: Sample Robot Trajectories of the SRSO Scenario for 500 Time Steps (32 s) Using the Hand-coded Rule-based Controller (a) and the Best Evolved ANN Controller (b): "S" represents the constant starting point and "E" the ending points, with "+" symbols placed along the trajectories every 40 time steps (2.56 s).

goal of the robot is to make one and *only one* full circle of the object as soon as possible without any collisions. A short evaluation span of 500 time steps[4] is chosen here. The robot always starts at the same initial position and orientation (facing the object) for all evaluation spans here to reduce noise effects. Walls are included in this scenario to facilitate the development of object avoidance behavior of the ANN controllers during evolutions. Walls can be distinguished from the circular-shaped object by their different sensory value patterns.

Figure 5.5 shows the sample robot trajectories with the hand-coded controller and the best ANN controller evolved, respectively. It is interesting to note the distinct behaviors of the two robots under different controllers. The robot with the hand-coded rule-based controller clearly follows the logic shown in Figure 5.3: It goes directly to the object, inspects it by walking around it, then leaves and starts a random walk. The robot with the evolved ANN controller hangs around in circles after it finishes the inspection task.

For this scenario, neural controllers that have access to an additional timer input have achieved better results than those do not, and are comparable to the hand-coded controller, which also uses timers. This implies that timing is a key factor here probably due to lack of spatial clues in the world. It was shown that recurrent neural networks can be trained to generate this additional timing signal as needed [Gers et al. 2002].

[4]Each time step simulates 64 ms in real time.



(a) (b)

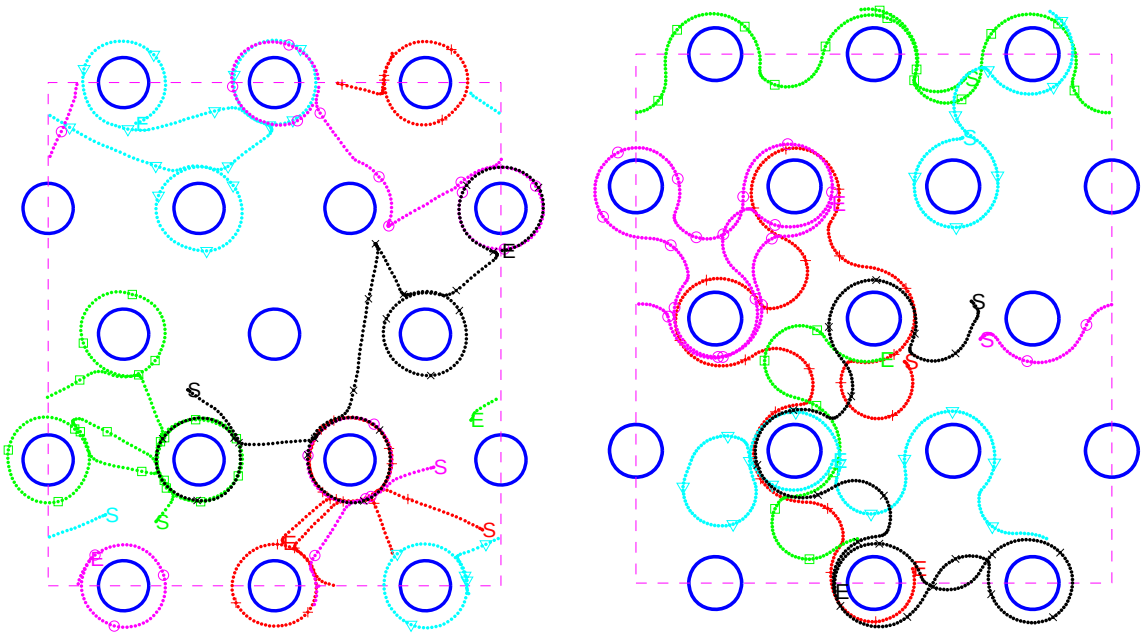
Figure 5.6: Sample Robot Trajectories of the SRMO Scenario for 2000 Time Steps (128 s) Using the Hand-coded Rule-Based Controller (a) and the Evolved ANN Controller (b). The dashed lines delimit the wrap-around boundaries. "S" represents the random initial starting points and "E" the ending points. The trajectories are shown in gradually changing colors with "+" symbols placed along the trajectories every 40 time steps (2.56 s).

5.3.4.2 Single Robot Multiple Objects (SRMO) Scenario

A single robot is set to explore the multi-object scenario shown in Figure 5.2 in the SRMO scenario. Again the goal here is to inspect (circle around) as much as possible the 12 distinct circular-shaped objects in the world. No walls are simulated in this scenario since wrap-around is applied to simulate a continuous repetitive world, similar to an unfolded ball. The evaluation span is 2000 time steps here for each ANN candidate controller during evolutions. The robot starts from a random initial position and orientation for each evaluation.

Figure 5.6 shows the sample robot trajectories with the hand-coded and evolved controllers for the SRMO scenario. The difference is quite obvious. The robot with the hand-coded controller always tries to make a full circle of each object it finds, then walks away in rather straight lines to search for "new" objects.[5] On the other hand, different evolved ANN controllers demonstrate a variety of robot behaviors. The one shown in Figure 5.6 (b) always walks in alternate curves: counterclockwise and closely curved when inspecting an object while clockwise and less curved otherwise. Most engineered solutions would probably apply the strategy of the hand-coded controller. It is surprising to discover that the evolved ANN-based strategy can work equally well here. In addition, the evolved controllers

[5]Note that the robot has no clue to figure out whether a newly discovered object has been inspected before or not according to the simple logic shown in Figure 5.3.



(a) (b)

Figure 5.7: Sample Robot Trajectories of the MRMO Scenario for 800 Time Steps (51.2 s) Using the Hand-coded Rule-based Controller (a) and the Evolved ANN Controller (b). The dashed lines delimit the wrap-around boundaries, but some trajectories beyond the boundaries are kept to enhance the display. "S" represents the random initial starting points and "E" the ending points. Different robots' trajectories are shown in different colors with different markers placed along the trajectories every 40 time steps (2.56 s).

no longer need additional temporal input here to be comparable with the performance of the hand-coded controller, which still depends on its timers.

However, it appears more difficult for the evolved controllers to achieve complete coverage of all objects. Because it often leaves (drifts away from) an object before fully inspecting it, it would generally take longer to fully inspect all objects in the world than the hand-coded one. Some possible reasons might be that the robot could hardly achieve complete coverage within the specified evaluation span, and there is no pressure in the evolution to emphasize complete coverage.

5.3.4.3 Multiple Robots Multiple Objects (MRMO) Scenario

A homogeneous team of five robots is employed to collectively inspect the 12 distinct circular objects in the simulated repetitive world, as shown in Figure 5.2. The goal as well as the wrap-around and random initial conditions are the same as those in Section 5.3.4.2, with an evaluation span of 800 time steps.

Figure 5.7 shows the sample robot trajectories with the hand-coded and evolved controllers for the MRMO scenario. Both behaviors seem to follow the respective strategies discussed in Section 5.3.4.2. The robots could stick to each other (i.e., non-perfect teammate avoidance) occasionally over a large

number of evaluations using either the hand-coded or the evolved controllers, owing to the limited sensor capability of the robots. However, it is noteworthy for the evolution to develop the obstacle (including teammates) avoidance behavior, especially when it is not explicitly defined in the fitness function, which only requires the robots to maximize their collective coverage over the evaluation span.

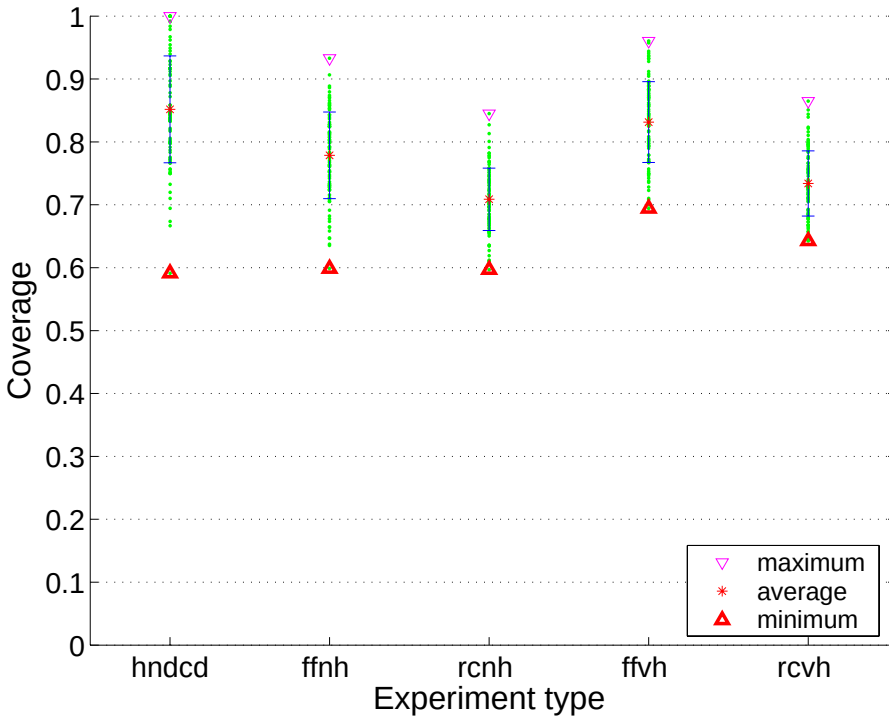
Figure 5.8 shows the performances of the hand-coded and best controllers evolved with different ANN types (as shown in Table 5.1) for the MRMO scenario. It is shown that the evolved controllers, especially those with a variable ANN topology, seem to have achieved comparable performances as the hand-coded controller in terms of average performance, and even slightly beat the hand-coded controller in terms of worst performance, appearing more robust to noise. It is remarkable for the evolutionary algorithms to automatically discover robust solutions from only single noisy evaluation of each candidate solution during the evolutions, which verifies its ability to work in noisy environments.

However, the hand-coded controller exhibits a better chance (>1%) of achieving complete coverage than the evolved ones (<1%) within the specified evaluation span. Generally longer evaluations would be needed for the evolved controllers to achieve complete coverage with certain confidence level. As discussed above, this might be due to the fact that there is no pressure in the fitness function to favor complete coverage, which could be further investigated in the future.

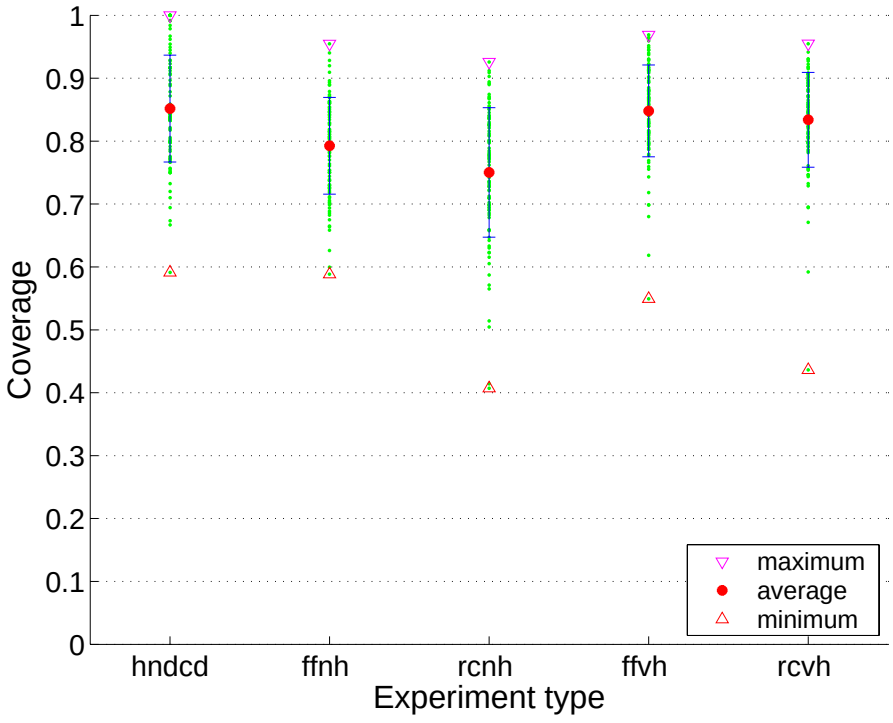
It is also observed that controllers evolved with variable ANN topologies can generally achieve better results than those with fixed ANN topologies. This demonstrates the power of evolutionary algorithms to synthesize appropriate ANN topologies for a given problem, and evolve the necessary synaptic weights simultaneously. For example, the trajectory shown in Figure 5.6 (b) is generated with a controller evolved with *ffvh*, while Figure 5.7 (b) is *rcvh*, both using variable ANN topologies. On the other hand, no significant performance differences are observed between the feed-forward and recurrent ANN's when all other conditions are the same.

Although the best evolved control strategies have achieved the same level of performance as each other as well as the hand-coded controller, their underlying ANN topologies are completely different, including both feed-forward and recurrent ANN's with a number of hidden neurons from two to six. In other words, even the evolutions might converge to a family of similar behavioral strategies for a given problem, they could be implemented by a variety of ANN's with vastly different topologies, which can provide human engineers with diverse alternative candidate solutions that might be difficult to conceive from human intelligence.

It is also noted that the evolutionary algorithm is able to adapt the controller solutions according to the collective or single robot scenarios. As a result the controllers evolved in collective scenarios can achieve better results in collective scenarios than those evolved in single robot scenarios, and vice versa.



(a)



(b)

Figure 5.8: Coverage Values Achieved by the Hand-coded Rule-based Controller (hndcd) and the Best Controllers Evolved with Different ANN Architectures (refer to the symbols defined in Table 5.1) and Selected according to (a) Minimum and (b) Average Performance for the MRMO Scenario. Each column shows the coverage values (*the green dots*) obtained by one controller during the 100 evaluations in its final noise test and the error bars indicate the standard deviation.

5.4 Case Study 2: Driver Behavior Modeling

As mentioned in Section 3.3.4, simple driver behaviors based on ANN could also be evolved with the same design synthesis method presented in Section 5.2. An initial attempt is made to evolve the lane-keeping driver behavior with the simple kinematic vehicle model described in Section 3.2.2. The evolved "lane-keeping" control strategy shows a smooth driving behavior and a good ability to adapt to new environments [Lutz 2005].

The driver model here is based on simple neural networks, whose weights are automatically tuned by the evolutionary algorithms. A simple feed-forward neural network with two outputs, two inputs, and a bias is chosen. Six synaptic weights, represented by real numbers in $[-1, 1]$, directly connect the inputs and bias to the outputs. The inputs of the neural network are the vehicle lateral position in lane (alignment) and the vehicle heading angle, which are updated and normalized to $[0, 1]$ by the simulation software at each time step, with a constant bias input of $+1$. The two outputs can be directly decoded into the vehicle's left and right speeds, or its forward and rotation speeds. A nonlinear sigmoid output function is used for both output neurons with output range $[0, 1]$. Scaling factors for the two neuron output functions are either tuned manually or evolved together with the six synaptic weights.

The candidate driver behavior models are tested in the kinematic embodied simulation environment, as shown in Figure 3.2. Individuals are evolved on two different road types: a strictly *straight road* and a *curved road*, which has alternate curved and straight segments. Evolved controllers that show good results in one world should be able to perform just as well on a different course. To achieve a more general control strategy for different situations, the curved road is designed using curved segments with different radii and straight segments with different lengths, as shown in Figure 5.9. The fitness function is defined in terms of the distance traveled during the evaluation span, the maximum lane deviation error and lateral acceleration, etc.

It is observed that individuals evolved on the curved road perform in most cases better than the ones evolved on the straight road. A curved road can be interpreted as a straight road with more noise at certain places. Therefore individuals evolved on the curved road are expected to perform better in noisy environments with a good ability to adapt to new environments.

Figure 5.10 shows the inputs and outputs time histories of a sample NN driver model evolved on the curved road shown in Figure 5.9. The left vertical axis represents the normalized NN input values, i.e., the vehicle heading angle and the lateral alignment, whereas the right axis designates the output speed values. The horizontal axis represents the time in time steps. In the first curve segment the robot turns left: The right speed increases and the left speed decreases. The difference between the left and right speeds (i.e., the yaw speed) is bigger on the last two curve segments with a smaller radius than the first two curve segments, while the average speed of the left and right

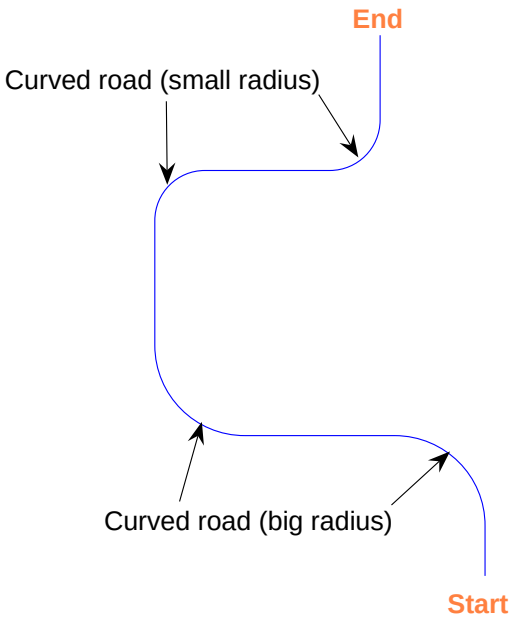


Figure 5.9: Curved Road Shape for Driver Model Evaluation

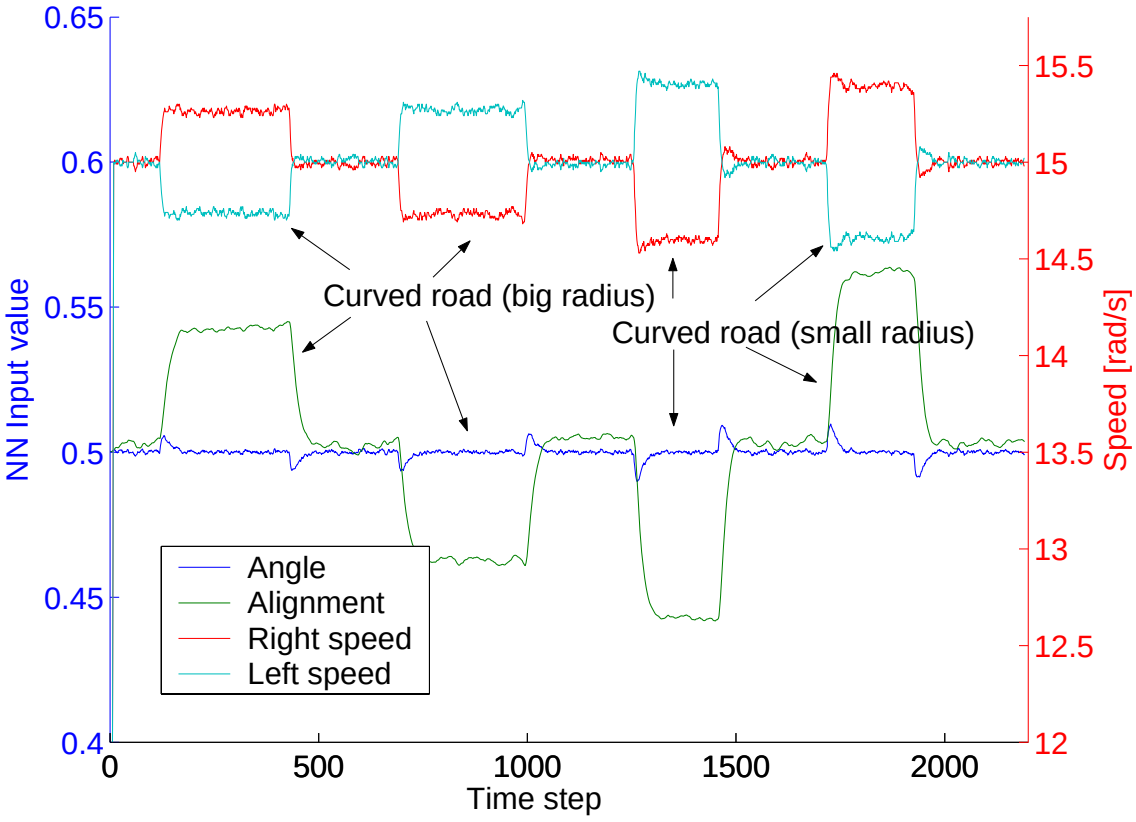


Figure 5.10: NN Inputs and Outputs of a Sample Driver Model Evolved on the Curved Road Shown in Figure 5.9

speeds (i.e., the forward speed) always keeps the maximum value allowed. This is exactly the kind of behavior promoted by the fitness function, which maximizes the distance traveled and minimizes the alignment error.

The vehicle heading angle stays almost constant except for the start and the end of the curves where the angle changes dramatically and then gets back gradually. While a constant non-zero alignment error is observed on the curves because the robot drifts toward the outside of the curves. This "drift" behavior is typical of proportional (P) controller, and it is noted that the evolved NN driver model is very close to a simple P controller. While a PI or PID controller as described in Section 3.3.3 would have performed better with zero alignment error but it is beyond the capability of the specified simplest neural network model. Alternatively, the same evolutionary algorithm used here could also be applied to the PID controller model as well as the rule-based models described in Section 3.3 to tune their parameters automatically under various situations.

5.5 Conclusion

The evolutionary algorithm is applied to automatically synthesize neural network controllers for autonomous robots in a noisy simulation environment. The evolutionary design synthesis methodology is validated again in the framework of two case studies concerned with collective robotic inspection of 2D regular structures as well as driver behavior modeling. It is demonstrated that both the NN topologies and parameters can be effectively tuned by the evolutionary algorithm, and that the best evolved NN controllers can achieve excellent and robust performances comparable to that of a manually tuned hand-coded rule-based controller with a variety of NN representations, providing multiple good candidate solutions for human engineers. In addition, the evolutionary design synthesis method also appears to be able to deal with the noise in fitness evaluation efficiently and adapt to the collective task nature well in terms of the collective robotic inspection case study.

In the future, the same methodology can be applied to more complex and realistic problems such as collective robotic inspection of 3D irregular space structures and/or jet propulsion systems as well as development of more complex and realistic driver behavior models. Implementation and verification of evolved controllers with real robots would also be meaningful.

Chapter 6

Collision Avoidance System

Collision avoidance systems (CAS) are an emerging automotive safety technology that assists drivers in avoiding potential collisions. The information sources of the collision avoidance systems come from multiple on-board sensors. The range, range rate, and angular information of other vehicles and/or objects around the host vehicle can be measured by radar, lidar, and/or cameras in real time, as mentioned in Section 4.2. Other regular on-board sensors measure host vehicle speed, acceleration, steering angle, yaw rate, etc. Collision avoidance systems process all the information in real time to keep track of the most current vehicle-to-vehicle kinematic conditions. When a potential collision threat is identified by the system, appropriate warnings are issued to the driver to facilitate collision avoidance. If the driver fails to react in time to the warnings to avoid the imminent collision, an overriding system takes over control to avoid or mitigate the collision in an emergency situation. Therefore collision avoidance systems could assist drivers in two ways, warning and/or overriding, according to the dynamic situation.

In developing a collision warning system (CWS), two important parameters involving driver behavior have to be considered. One parameter is the time it takes for the driver to respond to the crash alert and begin braking, i.e., driver reaction time, and the second parameter is the driver deceleration (or braking) behavior in response to this alert across a wide variety of initial vehicle-to-vehicle kinematic conditions. An overriding system has the advantage of being less sensitive to human factors, hence it is more promising in terms of achieving better and robust system performance. However, it is also the most intrusive way to assist the driver, so its timing has to be carefully designed to get driver acceptance.

In addition, both warning and overriding systems are subject to some objective hardware limits and environmental factors, such as the maximum traction available from the ground-tire contact and brake efficiency, etc. A traction sensor could be used to obtain an estimate of the current road traction conditions, as reviewed in [Li et al. 2006].

6.1 Previous Work

A lot of research has been done on collision warning systems, which are the first resort to assist drivers in collision avoidance. The key is to ensure that warnings are issued to drivers at the *appropriate* time, i.e., just in time for the driver to react and avoid the collision while not too early or too frequent to become a nuisance or distraction to the driver. Different measures were defined to characterize the emergency level of various dynamic situations, and different sets of human-vehicle experiments were carried out to calibrate these measures to human performances and reactions, based on which different warning criteria were developed to assist the human drivers.

6.1.1 Measures Defined

First, as mentioned above, quantitative measures need to be defined to characterize the emergency level of various dynamic situations. The measures defined in the literature include time-based, distance-based and deceleration-based measures.

One frequently used time-based measure is the *time-to-collision* or *time-to-contact* (TTC), which refers to the time it would take for a collision to occur at the prevailing speeds, distances, and trajectories associated with the host vehicle and the closest lead vehicle [van der Horst 1990]. In particular, the minimum TTC value (TTC_{min}) indicates how imminent a potential or actual collision has been during the process of approaching. More specifically, three different measures based on TTC were further investigated in [Kiefer et al. 2003]. The *TTC1* measure follows the TTC definition above, which is mathematically defined as the range R (i.e., the bumper to bumper distance between the two vehicles) divided by the closing speed between the two vehicles, or $-R/RR$, where RR is the range rate. Note that the vehicle speeds are assumed to remain constant here, i.e., the accelerations of both vehicles are ignored and assumed to be zero during the $TTC1$ calculation. The *inverse TTC1* measure is simply defined as the inverse of $TTC1$, or $-RR/R$. The *TTC2* measure is defined as the time it would take the host and lead vehicles to collide assuming the prevailing vehicle speeds and acceleration/deceleration values (i.e., at the current "constant" rate of speeding/slowing), and if either vehicle comes to a stop, it would remain stopped thereafter. Hence the difference between $TTC1$ and $TTC2$ is that the latter takes into account the acceleration information of both vehicles while the former does not.

Another related time-based measure is the *time headway* (t_h), which is calculated as the range between the two vehicles divided by the following host vehicle speed, or R/v_H [Fuller 1981]. Time headway is important because it specifies how much time the following vehicle has to react in case the lead vehicle suddenly brakes at the maximum deceleration level.

One important deceleration-based measure is the *required deceleration* (a_{req}) measure, which is defined as the constant deceleration level required for the host vehicle to avoid a potential collision

Type	Definition
$TTC1$	projected time to collision assuming prevailing speeds and distance $(-R/RR)$
$1/TTC1$	inverse of $TTC1$ $(-RR/R)$
$TTC2$	projected time to collision assuming prevailing speeds, accelerations, and distance
t_h	time headway between the host and lead vehicles (R/v_H)
a_{req}	constant deceleration level required to avoid a rear-end collision
D_{min}	projected minimum distance during collision avoidance process

Table 6.1: Various Threat Assessment Measures Defined in the Literature

with the vehicle ahead [Kiefer et al. 1999]. This measure is calculated under the same assumptions as the $TTC2$ measure above. In comparison, the *actual deceleration* (a_{act}) measure is defined as the constant deceleration level required to yield the actual stopping distance observed during an experiment or real scenario. The difference between the two measures is due to the safety margins adopted by individual drivers in avoiding the collisions during the practices.

One distance-based measure is the *projected minimum distance* (D_{min}) between the host and lead vehicles during the approaching/avoiding process [Brunson et al. 2002]. It is calculated using the prevailing range and vehicle speeds, and the assumption that the lead vehicle would keep the current acceleration level until it comes to a stop, while the host vehicle starts to brake at the maximum deceleration level constantly after an assumed driver reaction time, during which it keeps its original acceleration level. An alert is issued when the projected D_{min} is within a distance threshold D_{thresh}. Based on this warning criterion, the corresponding warning range (R_w) can be calculated, and a warning is issued if the actual range R is within R_w [Burgett et al. 1998]. Another related measure is the *projected time to* D_{min}, which indicates the imminence or urgency of the situation [Polychronopoulos et al. 2004].

Table 6.1 summarizes the various threat assessment measures reviewed above. Each of them characterizes the current dynamic situation in one way and can be used as basis for threat assessment and warning/overriding criteria. However, there does not exist a clear quantitative relationship between these measures and the threat level, as well as the best timing for warning and overriding actions.

6.1.2 Driver Reaction Time

Driver reaction time is an important parameter and plays a major role in the success of the collision warning systems. In this thesis the driver reaction time includes the human mental processing time in response to a signal or stimulus, the movement time for the driver's foot to switch from gas to brake petal, and the brake system delay.

Many research experiments have been performed to measure human driver reaction times to different stimuli under various situations. There have been comprehensive reviews on driver reaction times reported in the literature [Olson 1989, Sens et al. 1989, Green 2000]. It was noticed that the

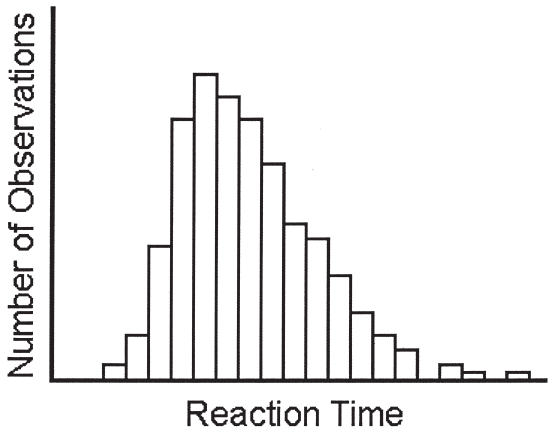


Figure 6.1: Hypothetical Reaction Time Distribution [Green 2000]

driver reaction time data reported were almost always skewed toward longer values, as shown in Figure 6.1. As a result, the lognormal probability distribution can be used as an approximate statistical distribution model for driver reaction time t_r with parameters μ and σ^2 [Taoka 1989, Brunson et al. 2002]. In other words, the logarithm of the driver reaction time ($\ln t_r$) is normally distributed with the same parameters μ and σ^2.

Of the various experiments conducted on driver reaction times, two kinds of situations are given particular attention in this thesis. One is normal driver reaction times toward unexpected natural driving scenarios, such as the onset of the brake lights of the lead vehicle or yellow traffic lights. The other is the driver reactions in response to some unexpected artificial signals, such as a red icon appearing in front of the driver or specific auditory signals, which could be considered as potential warning signals.

From the results reported in the various literature, the best estimate for natural driver brake reaction time to common but uncertain signals (e.g., lead vehicle brake lights or yellow traffic lights) lies between 1.14 and 1.38 seconds [Gazis et al. 1960, Sivak et al. 1982, Chang et al. 1985, Sivak et al. 1981]. Standard deviations of results vary widely across studies, but 0.6 s seems to be a good estimate. Hence the lognormal distribution model with parameters $\mu = 1.13$ s and $\sigma = 0.46$ s would approximately represent the natural human driver reaction time t_r with mean 1.25 s and standard deviation 0.6 s.

On the other hand, the experiments on driver reaction times in response to the sudden appearance of a red square reported mean values of 0.96 s on easy straight roads and 1.3 s on curvy routes, resulting approximately 1.13 s on average for all driving conditions [Alm and Nilsson 1994]. The driver brake reaction time in response to some completely unexpected auditory signals was estimated to be 0.9 s or longer in 50% of all sudden accident situations, and about 1.2 s on the 75th percentile [Johansson and Rumar 1971]. Finally, driver reaction times under different types of dual-modality

Type	mean	std	median(μ)	σ	75%	85%	90%
Natural (no alerts)	1.25	0.6	1.13	0.46	1.53	1.81	2.02
Visual alert	1.13	0.52	1.03	0.44	1.38	1.62	1.80
Auditory alert	0.99	0.44	0.9	0.43	1.2	1.40	1.55
Visual + Auditory alert	0.90	0.34	0.84	0.37	1.08	1.23	1.35

Table 6.2: Estimates of Unexpected Driver Reaction Time in Seconds

(i.e., both visual and auditory) crash alerts were extensively investigated in a series of experiments on potential forward collision warning (FCW) systems [Kiefer et al. 1999], where the shortest reaction times with the least variance were recorded under surprise, unexpected conditions. It was further verified that brake reaction times were faster (0.90 s versus 1.15 s on average) with FCW alerts [Kiefer et al. 2005a].

Table 6.2 gives a summary of driver reaction times in response to different types of unexpected stimuli, characterized by the lognormal probability density model with parameters μ and σ. Note that the parameter μ is also the median, i.e., the 50th percentile value, and the parameter σ is the dispersion parameter. The mean and standard deviation (std) values as well as the 75th, 85th, and 90th percentile values are also listed in the table.

6.1.3 Collision Warning Systems

The Crash Avoidance Metrics Partnership (CAMP) was established to accelerate the research in advanced automotive collision avoidance systems to improve traffic safety. In [Kiefer et al. 1999], CAMP developed basic elements of FCW systems, which provide alerts intended to assist drivers in avoiding or mitigating rear-end crashes. Crash alert timing and crash alert modality (auditory, visual and/or haptic) requirements as well as driver reaction time and braking behavior were studied by conducting a series of closed-course human factors studies using a "surrogate target" methodology, where drivers were asked to perform last-second braking maneuvers while approaching a slowing or stopped vehicle (surrogate target). Drivers were instructed to use either "normal" or "hard" braking to avoid a crash. Drivers' reaction times to a variety of stimuli under surprise and alerted conditions were also measured and combined with knowledge of drivers' braking behavior to develop the FCW alert model. This timing criterion intends to provide an alert after most attentive drivers would have started a "normal" last-second braking maneuver, yet soon enough for most inattentive drivers to still avoid a crash using last-second "hard" braking. This approach tries to minimize the number of nuisance alerts while maintaining high FCW effectiveness under tested conditions. Based on the required deceleration (a_{req}) measure, this model is significantly different from traditional models that are based on TTC or t_h.

In [Kiefer et al. 2003], a follow-on study extended the previous CAMP human factors work addressing FCW timing requirements by gathering not only "last-second" braking maneuver data, but

also data from "last-second" steering (or lane-changing) maneuvers. Drivers performed "normal" or "hard" last-second braking and steering maneuvers under a wide variety of vehicle-to-vehicle kinematic scenarios. It was observed that the mean last-second steering onsets tended to occur later (i.e., were more aggressive) than the mean last-second hard braking onsets when the closing speed was high.

Two last-second braking onset timing models, the *required deceleration model* and the *inverse TTC model*, were developed using the last-second maneuver database established from both CAMP studies mentioned above. Using the linear regression approach, the required deceleration model estimates a_{req} continuously in real time and uses it to decide if the driver is in a hard braking onset scenario. In contrast, the logistic regression statistical modeling technique is used in the inverse TTC model to predict the probability the driver is in a normal or hard braking scenario based on an inverse TTC threshold that decreases linearly with speed [Kiefer et al. 2005b].

In both CAMP studies, the *braking onset range*, estimated based on the above models, along with the assumed *delay time range*, is used to calculate the total *warning range* (R_w) as the crash alert warning criterion [Kiefer et al. 1999]. The delay time range is calculated based on the projected change in range during the driver reaction time interval assuming prevailing speeds and deceleration levels of the lead and host vehicles.

The effectiveness of the CAMP FCW timing approach described above was further tested under the *surprise trial methodology* and *visual occlusion* techniques [Kiefer et al. 2005a], which intended to simulate a "surprised" distracted driver, who had been intentionally distracted by look-down tasks or visual occlusion until the onset of an FCW alert, immediately following which the driver had to quickly decide upon and execute a crash avoidance maneuver. Results indicate that under the CAMP FCW alert timing conditions, drivers were able to execute an unassisted, successful braking maneuver for over 85% of the trials across the approach conditions examined, while the unsuccessful trial rates almost doubled when no alert was presented for the look-down trials, which may be due to long alert onset–look up delays[1] (the time between the alert onset and when the eyes "landed" on the forward view) for some drivers. The average alert onset–look up delay time was 1,505 ms for unsuccessful trials, while it was 566 ms for successful trials. In addition, it was observed that there was generally no age or gender effects under the FCW alert conditions across all various experimental approaches, suggesting that a "one size fits all" FCW alert timing approach may be feasible.

In a related research [Curry et al. 2005], a subset of the previous closed-course CAMP experiments with a surrogate target was replicated on the National Advanced Driving Simulator (NADS) facility for comparison and validation purposes. It was concluded that the test scenarios should emphasize high lead vehicle decelerations and high closing speeds (particularly when the lead vehicle

[1]Note that this is the additional delay time besides the assumed driver brake reaction time when the driver was looking down during the distraction.

is stationary), and attention should be focused on the interpretation of last-second hard braking or hard steering onset behavior. It was observed that the NADS data generally showed good agreement with the closed-course data under the above conditions. When there was disagreement, it was usually the case that the NADS drivers reacted more cautiously, initiating braking or steering earlier than their closed-course counterparts.

Along another line of research on CWS, the National Highway Traffic Safety Administration (NHTSA) developed an experimentally based rear-end collision warning algorithm and sponsored analysis of its performance [Brunson et al. 2002]. Integrated along with a General Motor (GM)–developed algorithm, the NHTSA alert algorithm processes data received from a vehicle-mounted radar and other sensors to alert drivers to potentially dangerous situations and the need to take evasive actions. The decision to issue an alert is based on the projected minimum distance (D_{min}) calculated at each time interval, assuming constant lead vehicle deceleration, a driver reaction time estimate, the maximum host vehicle deceleration level, and measured parameters characterizing the current host vehicle and vehicle-to-vehicle dynamic situations.

Two sets of theoretical analyses were performed on the NHTSA alert algorithm. The first analysis examined the performance under the assumption of perfect input data. The second analysis examined the effects of measurement noise and driver variability on the performance of the alert algorithm in terms of probability of false alarm (P_{FA}) versus probability of miss (P_{miss}). The results indicated that the driver response variability (braking level and reaction time) had a much greater impact on algorithm performance than the vehicle dynamics measurement errors.

Verification testing was also conducted with the NHTSA alert algorithm installed in a test vehicle equipped with a prototype CWS. It was noted that the algorithm performance depended on the ability of the radar system to report valid targets on curves and at longer ranges. Algorithm performance was most affected when the host vehicle was traveling at higher speeds. For instance, sometimes the radar detection range is even shorter than the imminent warning range. In addition, data quality and resolution also affect the algorithm performance, especially the resolution of relative acceleration (a_R) was the principal source of error. Moreover, it was shown through simulation results that the probability of collision was closely related to the probability distribution of driver reaction time t_r.

A special situation was further investigated where two vehicles were initially traveling at the same speed in the same lane when the lead vehicle suddenly braked [Burgett et al. 1998]. Following the same logic as above, a warning range R_w as well as its corresponding range rate RR_w could be computed assuming constant lead vehicle deceleration level a_L, driver reaction time t_r, initial speed v_0 and time headway t_h, and assumed maximum host vehicle deceleration $a_{H_{max}}$. Then a family of warning curves can be generated by plotting (R_w, RR_w) pairs on a R-RR plot that are parametric in a_L for each combination of initial conditions (v_0, t_h), as shown in Figure 6.2. It was then claimed

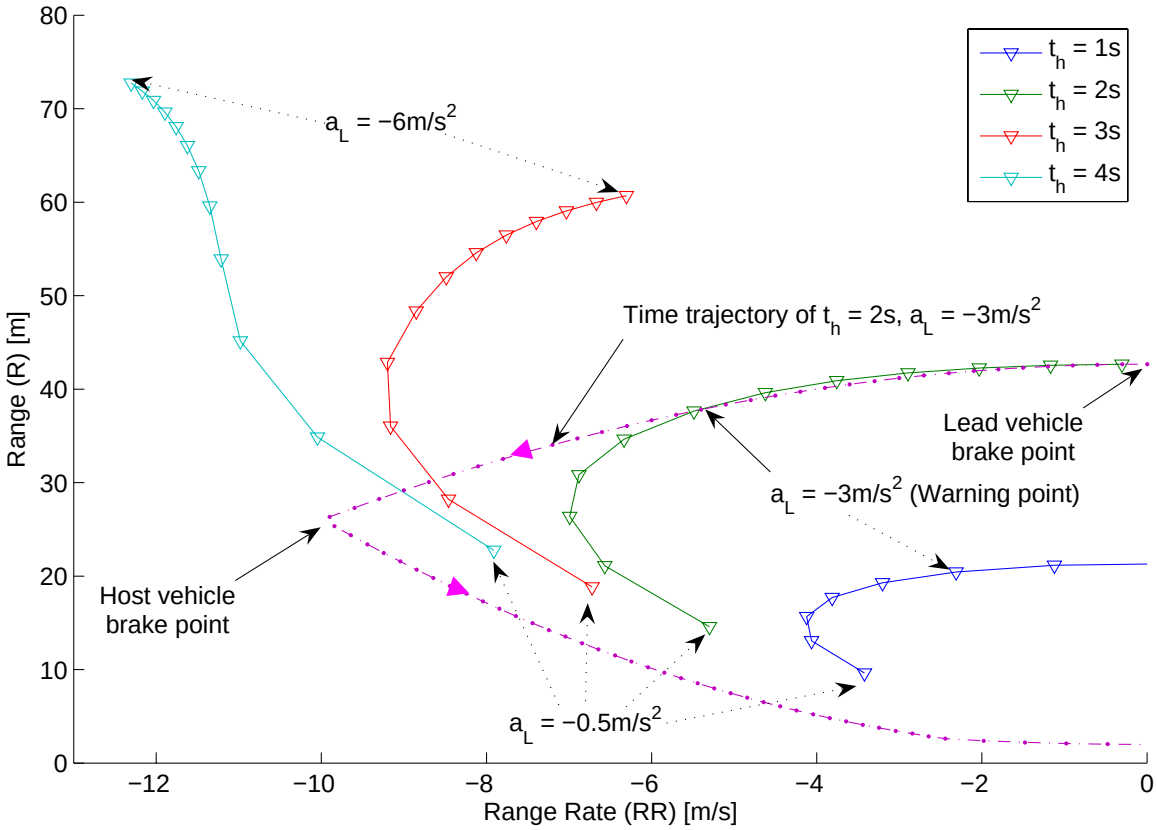


Figure 6.2: Warning Curves (solid lines) Parametric in a_L ($v_0 = 48\,\text{mph}$, $t_r = 1.5\,\text{s}$, $a_{H_{max}} = -5\,\text{m/s}^2$) with a Sample Time Trajectory (dash dot lines) of $t_h = 2\,\text{s}$ and $a_L = -3\,\text{m/s}^2$

that these warning curves could be used as an efficient warning criterion *without* the estimation of a_L. While it is desirable to eliminate the estimation of a_L for the warning criteria, this would not work under common conditions. An example is shown in Figure 6.2, where the warning curves for $v_0 = 48\,\text{mph}$ and several t_h values are plotted, along with a time trajectory for the case $t_h = 2\,\text{s}$ and $a_L = -3\,\text{m/s}^2$. It can be observed that the sample time trajectory is very close to the warning curve of $t_h = 2\,\text{s}$ between the lead vehicle brake point and the warning point, hence it is not clear when to issue the warning, even omitting the sensor noise in measuring range and range rate. In the example shown in the paper [Burgett et al. 1998], these warning criteria would work when t_h is rather long (e.g., $t_h = 5\,\text{s}$), which is, however, not the imminent situation concerned by the CWS/CAS.

6.2 Warning and Overriding Algorithms

Various warning and overriding algorithms have been developed and investigated in the literature [Lee and Peng 2005]. Most of these compute a warning range (R_w) based on the current kinematic situation, and a warning is issued if the current range R is less than R_w. Some of the algorithms calculate an overriding range (R_o), and automatic braking (overriding) is applied if R is within R_o.

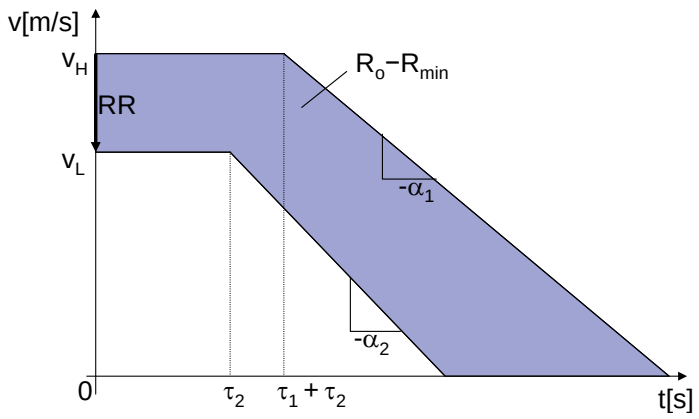


Figure 6.3: Interpretation of the Mazda Overriding Algorithm

6.2.1 Mazda Algorithm

The Mazda overriding algorithm [Doi et al. 1994] considers a hypothetic worst case, as shown in Figure 6.3. First, it assumes that initially both the host vehicle and the lead vehicle maintain constant speeds v_H and v_L, respectively. Then the lead vehicle starts to brake after time τ_2 at deceleration level $-\alpha_2$, while the host vehicle starts to brake after an additional time τ_1 at deceleration level $-\alpha_1$, which continues until both vehicles come to a full stop. The overriding range R_o is computed as the minimum range needed at time 0 to allow the above scenario to happen without collisions:

$$R_o = v_H \cdot \tau_1 - RR \cdot \tau_2 + \frac{v_H^2}{2\alpha_1} - \frac{v_L^2}{2\alpha_2} + R_{min} \tag{6.1}$$

where RR is the range rate, i.e., $RR \equiv v_L - v_H$, and R_{min} is a constant distance headway offset. The shaded area in Figure 6.3 is the required safety range buffer between the two vehicles should the hypothetic scenario described above happen. The following parameters were used: $\alpha_1 = 6\,\mathrm{m/s^2}$, $\alpha_2 = 8\,\mathrm{m/s^2}$, $\tau_1 = 0.1\,\mathrm{s}$, $\tau_2 = 0.6\,\mathrm{s}$, $R_{min} = 5\,\mathrm{m}$. The system provides a warning when the actual range R approaches R_o, i.e., $R_w = R_o + \epsilon$, where ϵ is a positive constant parameter. The system applies automatic braking to try to avoid or mitigate collisions if R is within R_o.

6.2.2 Honda Algorithm

The Honda algorithm [Fujita et al. 1995] uses the following warning criterion:

$$R_w = -2.2 \cdot RR + 6.2 \tag{6.2}$$

which is based on the $TTC1$ measure, as defined in Section 6.1.1, with a constant distance headway offset of 6.2 m. Warning is issued when the $TTC1$, after offset adjustment, is below 2.2 s.

The Honda overriding algorithm also considers a hypothetical scenario, as shown in Figure 6.4. It

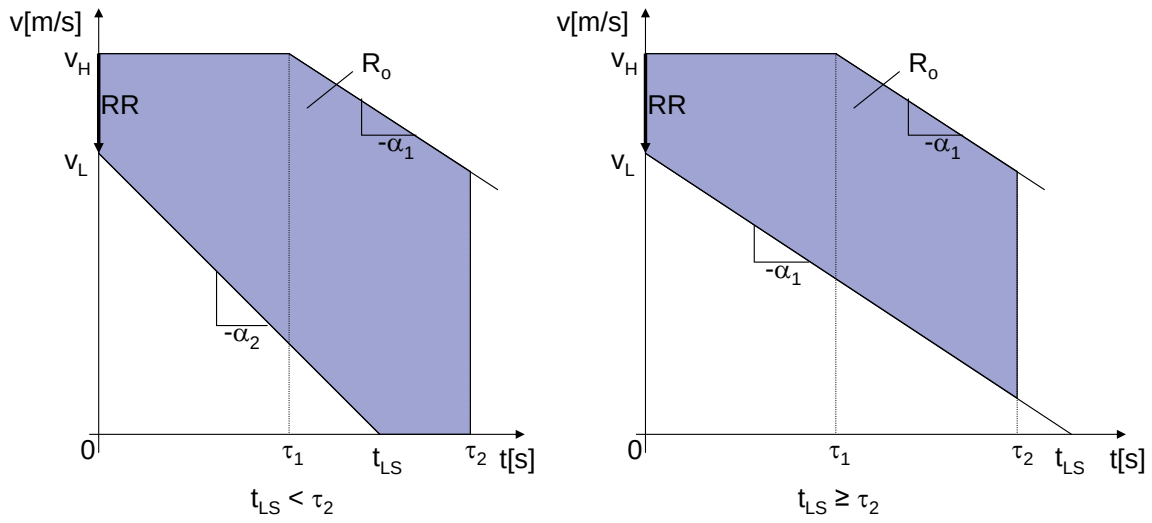


Figure 6.4: Interpretation of the Honda Overriding Algorithm

consists of two parts, depending on whether the lead vehicle is expected to stop within the considered time range τ_2. It is assumed that the lead vehicle brakes constantly at deceleration level $-\alpha_2$ (if the estimated lead vehicle stopping time $t_{LS} \equiv v_L/\alpha_2 < \tau_2$) or $-\alpha_1$ (if $t_{LS} \geq \tau_2$), while the host vehicle starts to brake after reaction time τ_1 at deceleration level $-\alpha_1$. The safety range R_o is estimated as the minimum range buffer needed to avoid collisions until τ_2 at both situations, which is represented by the shaded areas in Figure 6.4 and computed as follows:

$$R_o = \begin{cases} v_H \cdot \tau_2 - \frac{1}{2}\alpha_1(\tau_2 - \tau_1)^2 - \frac{v_L^2}{2\alpha_2} & t_{LS} < \tau_2 \\ -RR \cdot \tau_2 + \alpha_1\tau_1\tau_2 - \frac{1}{2}\alpha_1\tau_1^2 & t_{LS} \geq \tau_2 \end{cases} \tag{6.3}$$

The following parameters were used: $\alpha_1 = 7.8\,\mathrm{m/s^2}$, $\alpha_2 = 7.8\,\mathrm{m/s^2}$, $\tau_1 = 0.5\,\mathrm{s}$, $\tau_2 = 1.5\,\mathrm{s}$. Automatic braking is applied to assist collision avoidance if the current range R is within R_o.

6.2.3 Berkeley Algorithm

The Berkeley algorithm [Seiler et al. 1998] proposes a conservative R_w to provide a wide range of visual feedbacks (cautionary warnings) to the driver, and a non-conservative R_o to reduce undesirable effects of overriding to normal driving operations. As shown in Figure 6.5, it is assumed that the lead vehicle brakes at the maximum constant deceleration level $-\alpha$, while the host vehicle starts to brake after reaction time τ at the same deceleration level. The warning range R_w is estimated as the minimum range buffer needed to avoid collisions until both vehicles come to a full stop, while the overriding range R_o only considers the range buffer needed from time 0 to τ.

$$R_w = \frac{v_H^2 - v_L^2}{2\alpha} + v_H \cdot \tau + R_{min} \tag{6.4}$$

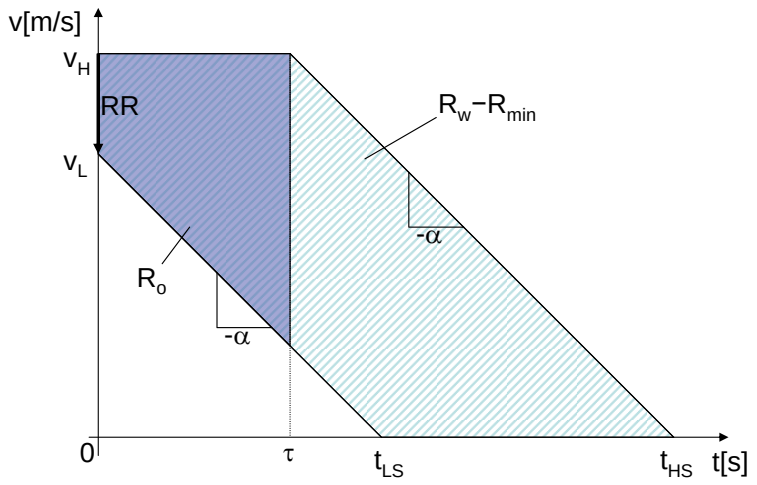


Figure 6.5: Interpretation of the Berkeley Warning/Overriding Algorithm

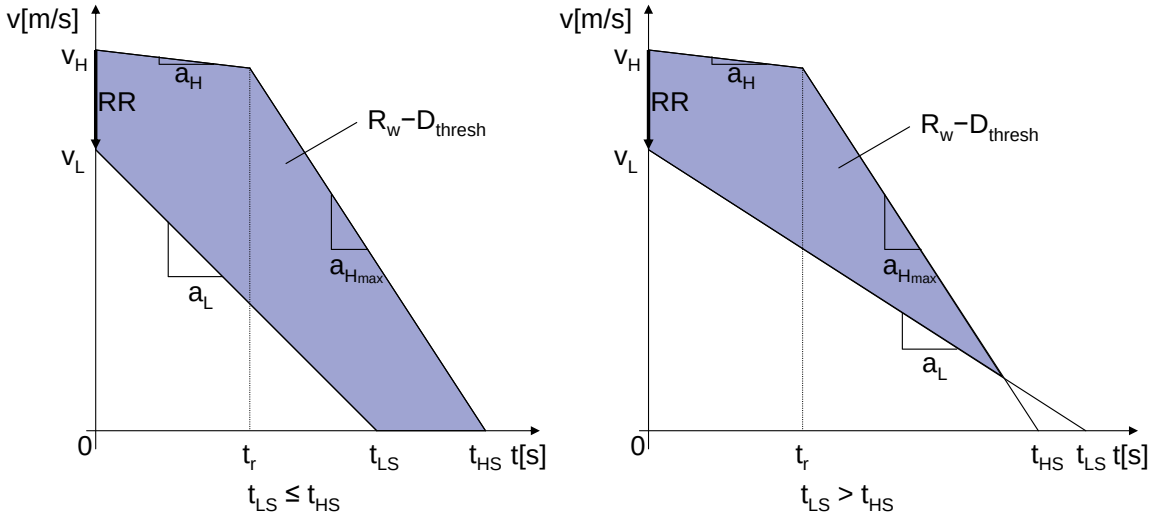


Figure 6.6: Interpretation of the NHTSA Alert Algorithm

$$R_o \quad = \quad -RR \cdot \tau + \frac{1}{2}\alpha\tau^2 \tag{6.5}$$

The following parameters were used: $\alpha = 6\,\mathrm{m/s^2}$, $\tau = 1.2\,\mathrm{s}$, $R_{min} = 5\,\mathrm{m}$.

6.2.4 NHTSA Alert Algorithm

The NHTSA alert algorithm [Brunson et al. 2002] considers slightly more complicated scenarios, as shown in Figure 6.6. It assumes that the lead vehicle brakes constantly at current deceleration level a_L until it comes to a stop if $a_L < 0$, while the host vehicle keeps current constant acceleration level a_H for a reaction time t_r, after which it starts to brake constantly at the maximum deceleration level $a_{H_{max}}$ ($a_{H_{max}} \leq a_L < 0$). Two different situations are considered, depending on whether the lead vehicle stops first or the host vehicle stops first under the above assumptions. The lead vehicle

stopping time t_{LS} and host vehicle stopping time t_{HS} are estimated by

$$t_{LS} = -\frac{v_L}{a_L} \tag{6.6}$$

$$t_{HS} = \begin{cases} t_r - \dfrac{v_H + a_H t_r}{a_{H_{max}}} & v_H + a_H t_r > 0 \\ -\dfrac{v_H}{a_H} & \text{otherwise} \end{cases} \tag{6.7}$$

Usually it is assumed that $v_H + a_H t_r > 0$, otherwise the host vehicle is already decelerating hard enough with a low frontal collision risk. The shaded areas in Figure 6.6 represent the range buffer needed to avoid collisions under both situations, as computed below:

$$R_w = \begin{cases} v_H \cdot t_r + \dfrac{1}{2} a_H t_r^2 - \dfrac{(v_H + a_H t_r)^2}{2a_{H_{max}}} + \dfrac{v_L^2}{2a_L} + D_{thresh} & t_{LS} \le t_{HS} \\ -RR \cdot t_r - \dfrac{1}{2} a_R t_r^2 + \dfrac{(RR + a_R t_r)^2}{2(a_L - a_{H_{max}})} + D_{thresh} & t_{LS} > t_{HS} \text{ or } a_L \ge -1 \end{cases} \tag{6.8}$$

where

$$D_{thresh} = 0.1 \cdot v_H + 2 \tag{6.9}$$

Here the system tries to estimate the relative acceleration ($a_R \equiv a_L - a_H$) in real time from the time derivative of range rate (RR) data measured by radar sensors, then the lead vehicle deceleration level a_L is computed from a_R estimation and a_H measurement, in contrast to previous algorithms where a_L is an assumed constant parameter. The driver reaction time t_r, which includes both the driver and system delays, is normally set to 1.5 s, and is reduced to 0.5 s when brake is applied. The assumed host vehicle maximum braking capability $a_{H_{max}}$ is set to $-5.4\,\mathrm{m/s^2}$ ($-0.55\,\mathrm{g}$) for imminent alerts, and lower levels for cautionary alerts.

6.2.5 CAMP Alert Algorithm

The CAMP alert algorithm [Kiefer et al. 1999] considers essentially the same scenarios with the same assumptions as the NHTSA alert algorithm. The only differences are that D_{thresh} is set to zero and that $a_{H_{max}}$ is replaced by *required deceleration* $a_{H_{req}}$, which is modeled by

$$a_{H_{req}} = \begin{cases} 0.685 \cdot a_L - 0.086(v_H + a_H t_r) - 1.617 & t_{LS} \le t_r \\ 0.685 \cdot a_L + 0.086(RR + a_R t_r) - 0.833 & t_{LS} > t_r \end{cases} \tag{6.10}$$

Note that the acceleration is expressed in $\mathrm{m/s^2}$, velocity and range rate in m/s, distance and range in m, and time in s. Hence $a_{H_{req}}$ varies according to the different underlying dynamic scenarios, and is no longer a prefixed parameter as the $a_{H_{max}}$.

6.2.6 Other Alert Algorithms

There are other alert algorithms developed for use in automotive collision warning and avoidance systems, as summarized in [Yang et al. 2003]. For example, if the current host vehicle acceleration a_H is set to zero in the NHTSA alert algorithm, then the first case of Equation 6.8 simplifies to

$$R_w = v_H \cdot t_r - \frac{v_H^2}{2a_{H_{max}}} + \frac{v_L^2}{2a_L} + R_{min} \qquad a_H = 0, t_{LS} \leq t_{HS} \tag{6.11}$$

In addition, if the lead vehicle keeps a constant speed slower than the host vehicle, i.e., $a_L = a_H = a_R = 0$, then the second part of Equation 6.8 simplifies to

$$R_w = -RR \cdot t_r - \frac{RR^2}{2a_{H_{max}}} + R_{min} \qquad a_L = a_H = 0, v_L < v_H \tag{6.12}$$

Furthermore, if the lead vehicle is stopped or stationary, i.e., $v_L = 0$, then the above equation can be rewritten as

$$R_w = v_H \cdot t_r - \frac{v_H^2}{2a_{H_{max}}} + R_{min} \qquad a_L = a_H = 0, v_L = 0 < v_H \tag{6.13}$$

There are still some other alert algorithms that are based on $TTC1$ $(-R/RR)$, t_h (R/v_H), or a linear combination of the two:

$$R_w = -RR \cdot \tau_1 + v_H \cdot \tau_2 + R_{min} \tag{6.14}$$

where τ_1 and τ_2 are predefined parameters as before.

6.3 New Criterion Proposal

As reviewed in the last section, most warning and overriding criteria used in automotive collision avoidance systems are expressed in terms of range, i.e., a warning and/or overriding range (R_w/R_o) is computed according to current sensor data and the respective algorithm parameters selected, then the control system decides whether to issue an alert or apply automatic braking based on the comparison result of the current range R with R_w and R_o. It is still difficult to clearly quantify the level of danger or threat from the comparison result since the range criteria vary nonlinearly under different dynamic conditions. For instance, a non-dimensional warning level w that varies linearly between R_w and R_o was proposed [Seiler et al. 1998]:

$$w = \frac{R - R_o}{R_w - R_o} \tag{6.15}$$

This is not appropriate since it is known that the danger level does not have a linear relationship with the range criteria. Therefore it is desirable to have a new criterion that directly quantifies the danger or threat level of the current dynamic situation objectively as well as assesses the urgency level for the required evasive action, e.g., braking. A new time-based measure is presented next for this purpose.

6.3.1 Time-to-last-second-braking (T_{lsb}) Measure

Time-to-last-second-braking (T_{lsb}), is a new time-based measure introduced here for rear-end collision threat assessment. It is defined as the time left for the driver or the control system at the current situation to take the last extreme evasive action, e.g., braking at the maximum level, to avoid a rear-end collision. It is calculated based on the assumptions that the lead vehicle would keep current deceleration or acceleration level a_L constantly until it comes to a full stop if $a_L < 0$ and in this case it would remain stopped thereafter, and that the host vehicle also maintains a current acceleration level a_H until the last moment when it will be able to decelerate at the maximum deceleration level $a_{H_{max}}$ to avoid collisions if necessary. Therefore T_{lsb} tries to estimate how long the host vehicle could maintain the current state until it has to brake at the maximum level to just avoid a potential rear-end collision with the lead vehicle. It can be estimated from the following six state variables:

$$T_{lsb} = f(v_H, a_H, R, RR, a_R, a_{H_{max}}) \tag{6.16}$$

where the current host vehicle speed v_H and acceleration a_H can be measured by vehicle state sensors, the current range R and range rate RR between the host and lead vehicles can be measured by on-board radar or lidar sensors, the current relative acceleration a_R between the two vehicles can be estimated from the RR history, and the current available maximum deceleration level $a_{H_{max}}$ can be estimated from tire-road friction coefficient monitor, as reviewed in [Li et al. 2006].

It follows from the definition of the new T_{lsb} measure that it gives a quantitative assessment of the current urgency and severity levels of the potential threats in terms of time, which would be very useful for threat assessment analysis of collision avoidance systems.

6.3.2 Scenario 1: Lead Vehicle Stopped or Moving Slowly ($a_L = 0$)

First, a simple scenario is considered where the lead vehicle is initially stopped or traveling at a constant slower speed than the host vehicle (i.e., $a_L = 0$, $RR < 0$). This is an important type of scenario where a collision avoidance system may be helpful. For instance, an inattentive driver might overlook a stopped or slowly moving vehicle ahead or underestimate its threat level until it is too late. The characteristic of this type of scenario is that the closing speed (RR) is usually high and often an evasive action is necessary even when the range R is still rather large. Hence the

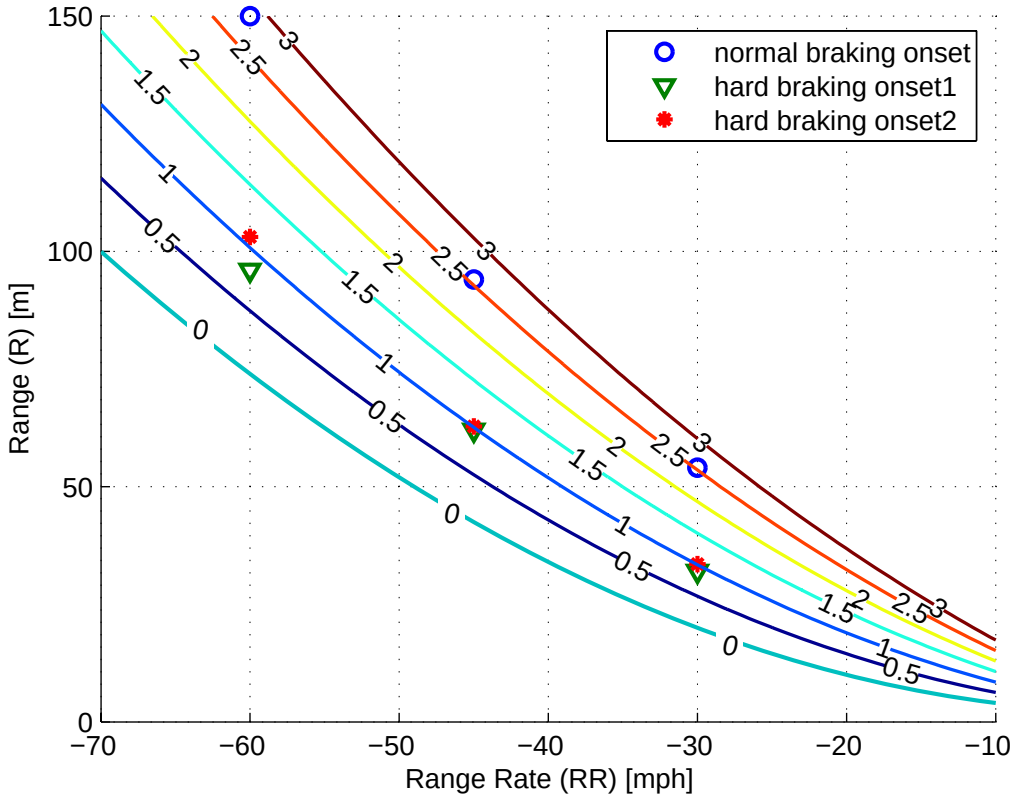


Figure 6.7: T_{lsb} Contours in Seconds with CAMP Data under Scenario 1: Host Vehicle Approaches Stopped or Slow Lead Vehicle ($a_L = a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$).

requirement for the driver or the sensory system to detect an object and estimate the R & RR at a rather far range (up to $150\,\mathrm{m} \sim 200\,\mathrm{m}$) is high in this case.

For simplicity, it is further assumed that the host vehicle currently keeps a constant speed v_H (i.e., $a_H = 0$). Then the time-to-last-second-braking T_{lsb} for this scenario only depends on R, RR, and $a_{H_{max}}$, as computed below:

$$T_{lsb} = -\frac{R - R_{min} + \dfrac{RR^2}{2a_{H_{max}}}}{RR} \tag{6.17}$$

which can be obtained by solving t_r from Equation 6.12. For a given road-tire friction condition, e.g., $a_{H_{max}} = -5\,\mathrm{m/s^2}$, and take $R_{min} = 2\,\mathrm{m}$, then the contours of T_{lsb} can be plotted as parabolic curves on a R-RR plot, as shown in Figure 6.7.

As mentioned in Section 6.1.3, the human drivers' last-second "normal" and "hard" braking onset data were recorded in CAMP experiments [Kiefer et al. 1999, Kiefer et al. 2003], and especially the data for the lead vehicle stationary trials are also plotted in Figure 6.7 using different markers. These data points represent the average range at host vehicle braking onsets under different conditions, i.e.,

last-second normal or hard braking condition, and different initial host vehicle speeds ($v_H = 30$ mph, 45 mph, or 60 mph), respectively. It can be observed that the last-second normal braking data align nicely with the $T_{lsb} = 2.5$ s curve, which implies that alert drivers normally brake 2.5 seconds before the last moment when maximum braking is needed. Furthermore, two sets of CAMP last-second hard braking data both align well with the $T_{lsb} = 1$ s curve, which suggests that an attentive driver would perform a last-second hard braking action about 1 second before maximum (the hardest) braking is needed to avoid a rear-end collision. These observations are especially true when the host vehicle speed is not too high (e.g., $v_H = 30$ mph or 45 mph) and within a range of 100 m or so, which implies that human drivers have a fairly good sense of urgency about when to take a last-second evasive action under an attentive condition and medium threat level, for instance, the host vehicle approaches a red light or a car stopped at an intersection, and their response timings appeared to be rather consistent under these conditions.

Therefore the proposed T_{lsb} measure appears to reflect human drivers' sense of urgency to take the last evasive action, and hence a good candidate for threat assessment analysis.

6.3.3 Scenario 2: Lead Vehicle Decelerating ($a_L < 0$)

In Scenario 2, the lead vehicle and the host vehicle initially travel at the same speed ($RR = 0$) with a certain initial time headway ($t_h = R/v_H$) between them, then the lead vehicle suddenly starts to brake at deceleration level a_L constantly. This type of scenario is also important in the study of collision avoidance systems, since the sudden braking of lead vehicles on freeways is a major cause of traffic accidents. The characteristic of this scenario is that usually the initial range R is not too large (e.g., $R < 50$ m) and the requirement for the driver or the sensory system to detect an abrupt negative change in RR and relative acceleration a_R is high.

For simplicity, it is still assumed that the host vehicle currently keeps a constant speed v_H ($a_H = 0$) in this case. As in the NHTSA alert algorithm described in Section 6.2.4, two different cases are considered in this Scenario 2 to estimate the time-to-last-second-braking T_{lsb}, depending on whether the lead vehicle is expected to stop first or not. The lead vehicle stopping time t_{LS} is still estimated by Equation 6.6, while the estimation of the host vehicle stopping time t_{HS} is slightly changed from Equation 6.7, since it depends on the T_{lsb} instead of t_r now:

$$t_{HS} = T_{lsb} - \frac{v_H}{a_{H_{max}}} \tag{6.18}$$

Accordingly, Equation 6.8 also changes to the following when $RR = 0$ and $a_H = 0$:

$$R = \begin{cases} v_H \cdot T_{lsb} - \dfrac{v_H^2}{2}\left(\dfrac{1}{a_{H_{max}}} - \dfrac{1}{a_L}\right) + R_{min} & t_{LS} \le t_{HS} \\ -\dfrac{1}{2}a_L T_{lsb}^2 + \dfrac{(a_L T_{lsb})^2}{2(a_L - a_{H_{max}})} + R_{min} & t_{LS} > t_{HS} \end{cases} \tag{6.19}$$

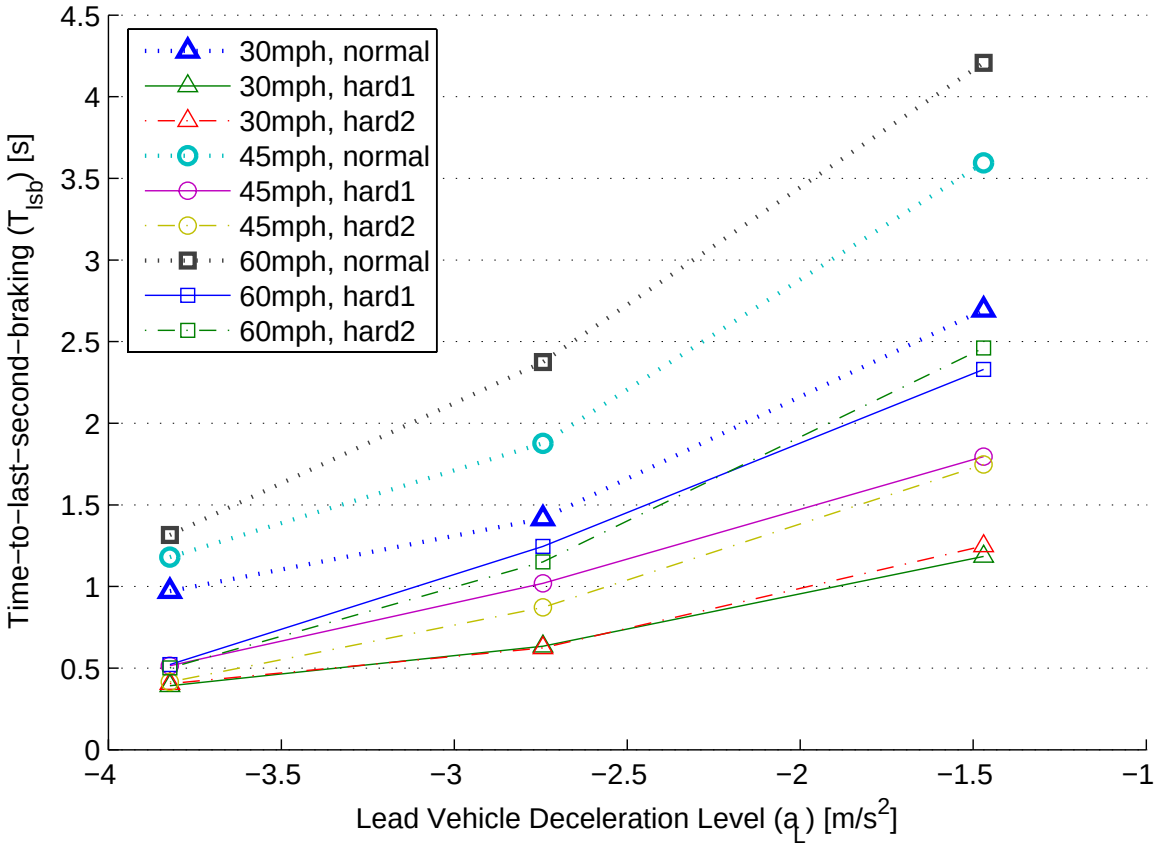


Figure 6.8: CAMP Data Represented in terms of the T_{lsb} Measure under Scenario 2: Lead Vehicle Decelerates ($a_{H_{max}} = -5\,\mathrm{m/s^2}$).

More generally, if $RR \neq 0$ and $a_H = 0$, the above equation has the following form:

$$R = \begin{cases} v_H \cdot T_{lsb} - \dfrac{v_H^2}{2a_{H_{max}}} + \dfrac{v_L^2}{2a_L} + R_{min} & t_{LS} \leq t_{HS} \\ -RR \cdot T_{lsb} - \dfrac{1}{2}a_L T_{lsb}^2 + \dfrac{(RR + a_L T_{lsb})^2}{2(a_L - a_{H_{max}})} + R_{min} & t_{LS} > t_{HS} \end{cases} \tag{6.20}$$

Then, T_{lsb} can be solved from Equation 6.19 or 6.20 depending on the current conditions. In practice, first it is assumed that the lead vehicle stops first ($t_{LS} \leq t_{HS}$), then T_{lsb} can be solved from the first part of the equations, then t_{HS} can be computed from Equation 6.18 and whether the condition $t_{LS} \leq t_{HS}$ holds or not can be verified. If $t_{LS} \leq t_{HS}$ holds, then the computation for T_{lsb} is completed. Otherwise T_{lsb} is solved from the second part of Equation 6.19 or 6.20 where the more positive solution is taken and the other solution discarded.

The human drivers' last-second "normal" and "hard" braking onset data recorded during the lead vehicle decelerating trials in CAMP experiments [Kiefer et al. 1999, Kiefer et al. 2003] can be plugged in the above equations to compute the T_{lsb} measure, as shown in Figure 6.8. The CAMP data include average range R and range rate RR at host vehicle braking onset under different conditions,

Parameter	Noise Distribution
v_H	$U[-0.15, 0.15]$
a_H	$G(-0.07, 0.17)$
R	$G(0.4, 0.025)$
RR	$U[-0.0625, 0.0625]$
a_R	$G(-0.6, 0.1)$

Table 6.3: Input Noise Distributions

i.e., last-second normal or hard braking condition, different initial vehicle speeds (v_H = 30 mph, 45 mph, or 60 mph), and different lead vehicle deceleration levels ($a_L = -1.5\,\mathrm{m/s^2}$, $-2.7\,\mathrm{m/s^2}$, or $-3.8\,\mathrm{m/s^2}$), respectively. It can be noted from the figure that the T_{lsb} measure for all of the last-second hard braking data under the heavy lead vehicle braking scenario ($a_L = -3.8\,\mathrm{m/s^2}$) converge to about 0.5 second while the T_{lsb} for the corresponding last-second normal braking data are between 1 and 1.5 seconds, implying the urgency and severity of this kind of scenario. In addition, the time buffer left until last-second braking increases as the lead vehicle deceleration level decreases and/or the host vehicle speed increases, corresponding to lower threat levels.

6.3.4 General Scenario

In general, as with the NHTSA alert algorithm described in Section 6.2.4, two different cases are considered to estimate the T_{lsb}, depending on whether the lead vehicle is expected to stop first or not. The lead vehicle stopping time t_{LS} is still estimated by Equation 6.6, while the estimation of the host vehicle stopping time t_{HS} is computed as follows:

$$t_{HS} = T_{lsb} - \frac{v_H + a_H T_{lsb}}{a_{H_{max}}} \qquad v_H + a_H T_{lsb} > 0 \tag{6.21}$$

Generally it is assumed that the condition $v_H + a_H T_{lsb} > 0$ holds[2] and the T_{lsb} measure can be solved from the following equations, using the same strategy as described in Section 6.3.3.

$$R = \begin{cases} v_H T_{lsb} + \frac{1}{2} a_H T_{lsb}^2 - \frac{(v_H + a_H T_{lsb})^2}{2a_{H_{max}}} + \frac{v_L^2}{2a_L} + R_{min} & t_{LS} \le t_{HS} \\ -RR \cdot T_{lsb} - \frac{1}{2} a_R T_{lsb}^2 + \frac{(RR + a_R T_{lsb})^2}{2(a_L - a_{H_{max}})} + R_{min} & t_{LS} > t_{HS} \end{cases} \tag{6.22}$$

6.3.5 Error Estimation of the T_{lsb} Measure

From the above calculation process of the T_{lsb} measure, it follows that the error of the estimated T_{lsb} depends on the error or measurement noise of the six underlying state variables as specified in Equation 6.16. For simplicity, it is assumed that the input measurement noise is generated as independent random variables with the distributions given in Table 6.3 [Brunson et al. 2002]. Here

[2] Otherwise the host vehicle is already decelerating hard enough, hence no emergency.

Parameter	Scenario 1	Scenario 2
v_H	$U[20, 30]$	$U[20, 30]$
a_H	$L(0, 0.3)$	$L(0, 0.3)$
R	$U[60, 80]$	$U[20, 40]$
RR	$U[-v_H, -v_H + 5]$	$U[-v_H + 20, -v_H + 30]$
a_R	$L(-a_H, 0.3)$	$L(-5 - a_H, 0.3)$

Table 6.4: True Input Distributions

$U[a, b]$ represents the uniform distribution in the interval from a to b, while $G(\mu, \sigma)$ represents the Gaussian distribution with mean μ and standard deviation σ. All units are metric (m, m/s, and m/s^2) in this section unless otherwise noted. These noise distributions were derived from a noise analysis of data collected from the prototype collision warning system in the Engineering Development Vehicle (EDV) developed under the Automotive Collision Avoidance System field operational test (ACAS FOT) [Brunson et al. 2002].

In order to estimate the error of the T_{lsb} measure, the assumed true input parameters under different scenarios are drawn randomly using the distributions specified in Table 6.4, where $L(\mu, \sigma)$ represents the Laplacian distribution with mean μ and standard deviation σ. The details of Scenario 1 (lead vehicle stopped or moving slowly) and Scenario 2 (lead vehicle decelerating) are described in Section 6.3.2 and 6.3.3, respectively.

In addition, the true maximum available host vehicle deceleration $a_{H_{max,true}}$ is drawn from a truncated Gaussian distribution with mean $-5.9\,\mathrm{m/s^2}$, standard deviation of $1\,\mathrm{m/s^2}$, minimum of $-7.8\,\mathrm{m/s^2}$, and maximum of $-2.9\,\mathrm{m/s^2}$. Also it is assumed that $a_{H_{max}}$ can be estimated based on a friction coefficient monitor within $\pm 10\%$ white noise.

The true time-to-last-second-braking ($T_{lsb,true}$) can be calculated based on the true inputs drawn from the random distributions described above, and the estimated value ($T_{lsb,est}$) can be calculated based on the corresponding noisy sensor inputs data, which are obtained by adding sensor measurement noise (drawn from Table 6.3) to the true inputs.

Then the relative frequency distribution of the error of the T_{lsb} measure (i.e., $T_{lsb,est} - T_{lsb,true}$) can be estimated by repeating the above calculations for a large number of trials and normalizing the data, as shown in Figure 6.9. The various percentile values and statistical measures of the T_{lsb} estimation error are summarized in Table 6.5. It can be observed from the figure and the table that, under the current assumptions, 99% of the T_{lsb} estimation error range is within 1 s, and that the estimated value of T_{lsb} will not exceed the true value by more than 0.25 s with a probability of over 99.9%.

Percentile	0.1%	1%	50%	99%	99.9%	mean	std
Scenario 1	-1.0567	-0.7966	-0.2587	0.1640	0.2433	-0.2672	0.2067
Scenario 2	-0.8133	-0.6631	-0.2714	0.0330	0.1015	-0.2784	0.1577

Table 6.5: Error of T_{lsb} Estimation Due to Sensor Noise in Seconds

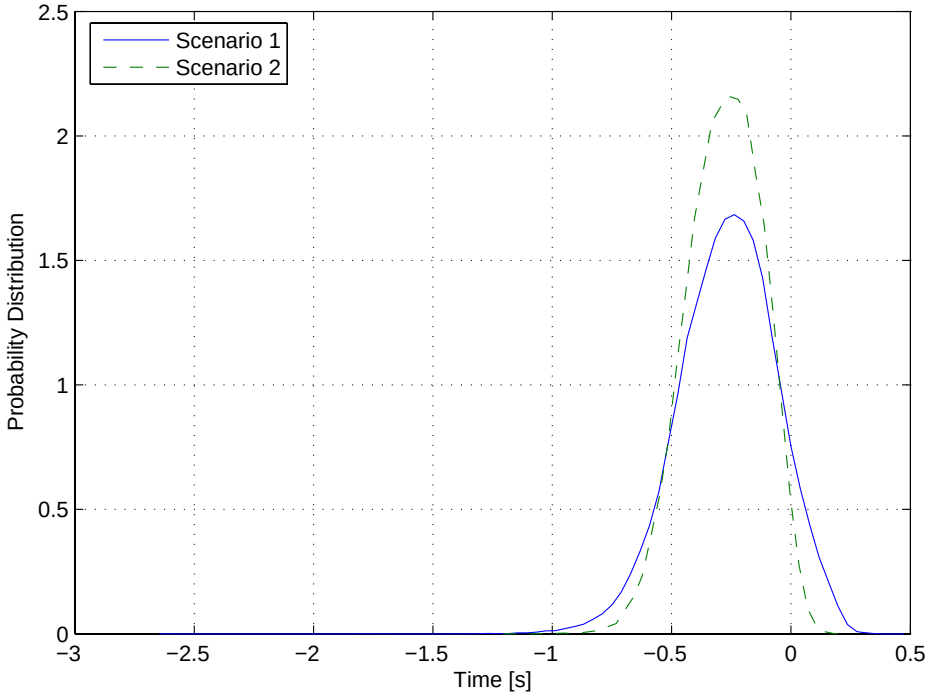


Figure 6.9: Error Distributions of the Estimated T_{lsb} ($T_{lsb,est} - T_{lsb,true}$) due to Sensor Noise under Scenario 1 and Scenario 2

6.3.6 Warning/Overriding Criteria in terms of T_{lsb}

As mentioned above, the T_{lsb} measure provides a straightforward and quantitative threat assessment of the current dynamic situation. From its definition it follows that potential collisions would be avoided if the driver or the control system could react within T_{lsb} with a sufficient level of deceleration.

From previous work on driver reaction times as reviewed in Section 6.1.2, human drivers usually do not have a consistently quick response time on the road. It may take up to 2 s to account for 90% of drivers' reaction times under natural driving scenarios without any warning signals. The situation is slightly better in that 90% of drivers can react within 1.8 s if a visual warning signal is used, 1.55 s if an auditory warning signal is issued, and 1.35 s if visual plus auditory warning signals are applied. However, on the other hand, the interference level of the warning signals also increases (from none, visual, auditory, to visual + auditory signal) as the driver reaction time decreases. The higher the interference level, the more probably drivers would experience the prewarning signals as a nuisance. Hence it is desirable to set the warning timing not too early to reduce the interference level, and at the same time not too late to give most drivers sufficient time to react. As a result of this trade-off it is difficult to achieve a satisfactory performance if the collision avoidance system solely relies on human drivers to take action in an emergency, due to the significant variations in driver behavior.

To overcome human driver limitations, an overriding system can be used at critical moments to automatically apply brakes at the maximum level to avoid collisions. The advantages are that it is not subject to the influence of driver reaction time and braking level variability, and that the T_{lsb} measure can give a relatively accurate estimate of how much time is left for the overriding system to react.

Based on the above discussions and observations, the following warning and overriding criteria in terms of the T_{lsb} measure are proposed:

- $1.5\,\text{s} \leq T_{lsb} < 2.5\,\text{s}$: Cautionary warning (e.g., visual signal)
- $0.5\,\text{s} \leq T_{lsb} < 1.5\,\text{s}$: Imminent warning (e.g., visual + auditory signal)
- $T_{lsb} < 0.5\,\text{s}$: Overriding (automatic braking)

The overriding threshold (0.5 s) is chosen to avoid collisions with a probability of 99.9%, according to the T_{lsb} error distribution described in Section 6.3.5 and assuming the system delay of automatic braking to be 250 ms. The CAMP data shown in Section 6.3.2 and 6.3.3 also suggest that alert drivers would have taken a normal or hard braking action before the T_{lsb} drops down to 0.5 s in most situations.

The two one-second warning stages are defined according to general human driver reaction times and the error distribution of T_{lsb} estimation. The warning thresholds can be further adjusted according to each individual driver's responsiveness and sensitivity level to warnings. For instance, a responsive driver might desire shorter warning time ranges than a slow driver.

The proposed T_{lsb} warning and overriding criteria have several advantages over the previous ones as reviewed in Section 6.2. First, they are defined in time domain instead of distance domain, which is in agreement with natural human sense and judgment of urgency. Besides, T_{lsb} gives a concrete time measure in terms of how much time is left for the driver or the control system to react to avoid a potential rear-end collision ahead, which serves as an excellent direct measure of the urgency and severity of threats under current situations.

Second, the estimation process of the T_{lsb} measure takes into account all possible current dynamic information (i.e., $v_H, a_H, R, RR, a_R, a_{H_{max}}$) while most previous algorithms only used partial updated information and assumed the rest of the state variables to be constants. It follows that the estimation of T_{lsb} will be more sensitive to real-time sensor noise and that the accuracy of T_{lsb} estimates can be improved by increasing the reliability and precision of sensor measurements. However, even when the sensor data are noisy, it is still better than a constant assumption in most cases.

Third, as mentioned above, the T_{lsb} criterion is less sensitive to human driver variability. In contrast to previous algorithms, the computation of the T_{lsb} measure no longer depends on an assumed human driver reaction time t_r, even though the warning criterion is still established with reference

to the human driver reaction times. The overriding criterion depends on neither human driver reaction time nor braking behavior, which are two important human factors in other warning/overriding algorithms.

Fourth, the overriding system can avoid collisions more effectively at the last moment based on the T_{lsb} measure. According to the error distribution analysis of the estimated T_{lsb} measure described in Section 6.3.5, and assuming the automatic braking system delay to be 250 ms, the overriding system is able to avoid rear-end collisions with a probability of over 99.9%.

Finally, the T_{lsb} measure can be combined with TTC information to compare with last-second steering possibilities, which will be discussed in Section 6.4.4.

6.4 Analysis

In this section, the performance of the proposed warning and overriding criteria is analyzed in terms of *probability of miss* (P_{miss}) and *probability of false alarm* (P_{FA}) first. Then the proposed criteria are compared with other warning and overriding algorithms under both Scenario 1 and Scenario 2. Finally the possibilities of last-second braking versus steering are discussed.

6.4.1 Performance in terms of P_{miss} versus P_{FA}

As mentioned above, a good collision avoidance system should assist drivers with effective collision avoidance during emergencies and at the same time reduce its disturbance to drivers under non-emergency situations. In other words, the system should minimize both the probability of miss (P_{miss}) and the probability of false alarm (P_{FA}) simultaneously. A *miss* event is defined as a situation where a warning is not issued (or an overriding action not performed) when the true conditions indicate that an alert (or an overriding action) should be given (taken). Conversely, a *false alarm* event is a situation where an alert is issued to the driver or an overriding action is taken when the real conditions did not warrant such alert level or the overriding action. Mathematically they are defined as follows according to the proposed T_{lsb} warning and overriding criteria:

$$P_{miss} = Prob(T_{lsb,est} \geq T_{lsb,true} + 0.5\,\text{s} \,|\, T_{lsb,true} < 2.5\,\text{s}) \tag{6.23}$$

$$P_{FA} = Prob(T_{lsb,est} \leq T_{lsb,true} - 1\,\text{s} \,|\, T_{lsb,est} < 2.5\,\text{s}) \tag{6.24}$$

Therefore a miss event is declared when the warning or overriding action is at least 0.5 s late, while a false alarm event is identified when the warning or overriding action is more than 1 s early.

Using the same true input and noise distribution assumptions as specified in Table 6.3 and 6.4 of Section 6.3.5, the following results can be obtained for the proposed T_{lsb} warning and overriding criteria after 10^7 test trials are conducted for both Scenario 1 and 2, respectively.

$$\text{Scenario 1:} \quad P_{miss} = 1.0120 \times 10^{-6} \quad \text{and} \quad P_{FA} = 9.5841 \times 10^{-4}$$

$$\text{Scenario 2:} \quad P_{miss} < 1.3792 \times 10^{-7}\ ^{3} \text{ and} \quad P_{FA} = 3.0684 \times 10^{-5}$$

These data imply that Scenario 1 is more dangerous than Scenario 2 under the current assumptions.

In comparison, P_{miss} and P_{FA} were defined in terms of the projected minimum distance measure (D_{min}) [Brunson et al. 2002], as described in Section 6.1.1.

$$P_{miss} = Prob(D_{min,est} \geq 2\,\text{m} \,|\, D_{min,true} \leq 0\,\text{m}) \tag{6.25}$$

$$P_{FA} = Prob(D_{min,est} < 2\,\text{m} \,|\, D_{min,true} \geq 4\,\text{m}) \tag{6.26}$$

The NHTSA alert algorithm with $a_{H_{max,est}} = -5.4\,\text{m/s}^2$ and $t_{r,est} = 1.5\,\text{s}$ has the following results reported:

$$\text{Scenario 1: } P_{miss} = 0.03 \text{ and } P_{FA} = 0.65$$

And similar results were also obtained for Scenario 2.

A significant difference can be observed between these two criteria and their performance results. This is because that the NHTSA warning algorithm and the corresponding P_{miss} and P_{FA} are defined in terms of D_{min} in distance domain where the permissible error range for D_{min} without incurring a miss or false alarm event is only $[-2,\ 2]\,\text{m}$, while the proposed warning/overriding criteria and the corresponding P_{miss} and P_{FA} are defined in terms of T_{lsb} in time domain where the allowable error range for T_{lsb} is $[-1,\ 0.5]\,\text{s}$. In fact $[-2,\ 2]\,\text{m}$ is a narrow error range when the host vehicle speed or range rate is high, which is why the corresponding P_{miss} and P_{FA} are so high. Additionally, the error range represents different lengths in the time domain under different kinematic conditions. Therefore it makes more sense to use the time domain criteria and performance definitions in terms of T_{lsb} since it quantifies our human perception of urgency and it applies the same performance criterion in terms of permissible error range in time under different kinematic conditions, adding one more advantage to the T_{lsb} warning and overriding criteria in addition to those mentioned in Section 6.3.6.

6.4.2 Comparison under Scenario 1 ($a_L = 0$)

In this section, the first four warning and overriding algorithms described in Section 6.2 are compared with the proposed T_{lsb} warning and overriding criteria under Scenario 1, i.e., the host vehicle approaches a stopped or slowly moving lead vehicle with constant speed ($a_L = a_H = 0$).

As mentioned in Section 6.3.2, the T_{lsb} measure only depends on R and RR when $a_{H_{max}}$ is assumed, and its contours can be plotted as parabolic curves on a R-RR plot, as shown in Figure 6.7.

[3] Note that this result is obtained because no miss event is observed during the 10^7 test trials where $T_{lsb,true} < 2.5\,\text{s}$ holds for 7,250,553 trials.

In Figure 6.10, the T_{lsb} contours are plotted again with thick curves representing the warning and overriding thresholds proposed in Section 6.3.6, i.e., $T_{lsb} = 0.5\,\text{s}$, $1.5\,\text{s}$, and $2.5\,\text{s}$.

In contrast, other algorithms have different warning or overriding thresholds for lead vehicle stopped (LVS) scenario and lead vehicle moving (LVM) scenario, respectively, as shown in Figure 6.10. For the LVS scenario ($v_L = 0$), the Mazda overriding curve is close to the $T_{lsb} = 0.5\,\text{s}$ curve, the Berkeley warning curve is roughly close to the $T_{lsb} = 1\,\text{s}$ curve, and the NHTSA warning curve is close to the $T_{lsb} = 1.5\,\text{s}$ curve. Note that these three curves are all concave curves like the T_{lsb} contours. While the Honda warning/overriding and the Berkeley overriding thresholds are straight lines on the *R-RR* plot, implying that they are only based on the $TTC1$ measure with a possible constant distance headway offset adjustment.

For the LVM scenario, only the threshold curves with $v_H = 70\,\text{mph}$ are plotted in Figure 6.10, hence the LVS and LVM curves for the same algorithm intersect at $RR = -70\,\text{mph}$. Note that all LVM threshold curves are convex curves except for the NHTSA warning curve, this is because they all assume certain *constant* lead vehicle deceleration level a_L while the NHTSA alert algorithm uses the *current* a_L values estimated in real time. However, when v_L is close to v_H, the possibility of sudden braking of the lead vehicle has to be considered even the current $a_L = 0$, which converts to the Scenario 2 to be discussed later.

Furthermore, it can be observed from Figure 6.10 that the NHTSA and the proposed T_{lsb} warning criteria are the most conservative under the LVS scenario, giving drivers sufficient prewarning. It is also noted that some warning and overriding thresholds even fall below the $T_{lsb} = 0$ curve, implying that a deceleration level higher than $-5\,\text{m/s}^2$ is needed to avoid collisions at these situations. In particular the Honda and Berkeley overriding algorithms require a deceleration level[4] of more than $1\,\text{g}$ ($-9.8\,\text{m/s}^2$) at the time of overriding when $RR = -70\,\text{mph}$, which is obviously too large for collision avoidance, even though they may still assist with collision mitigation.

6.4.3 Comparison under Scenario 2 ($a_L < 0$)

When the lead vehicle decelerates ($a_L < 0$), the situation becomes more complex with multiple possibilities under different kinematic conditions. For simplicity, it is still assumed that the current host vehicle acceleration $a_H = 0$ and it will be able to decelerate at the maximum level of $a_{H_{max}} = -5\,\text{m/s}^2$.

First a typical case is considered where the lead vehicle and the host vehicle travel at the same speed (e.g., $v_L = v_H = 70\,\text{mph}$) initially, and then the lead vehicle suddenly starts to brake at a constant deceleration level $a_L < 0$. At the moment the lead vehicle starts to brake (i.e., $RR = 0$ still holds), the T_{lsb} contours can be plotted in terms of R and a_L, as shown in Figure 6.11 (a).

[4]The required deceleration level (in m/s^2) for collision avoidance can be simply computed as $-RR^2/(2R)$ in this case where RR is expressed in m/s and R in m.

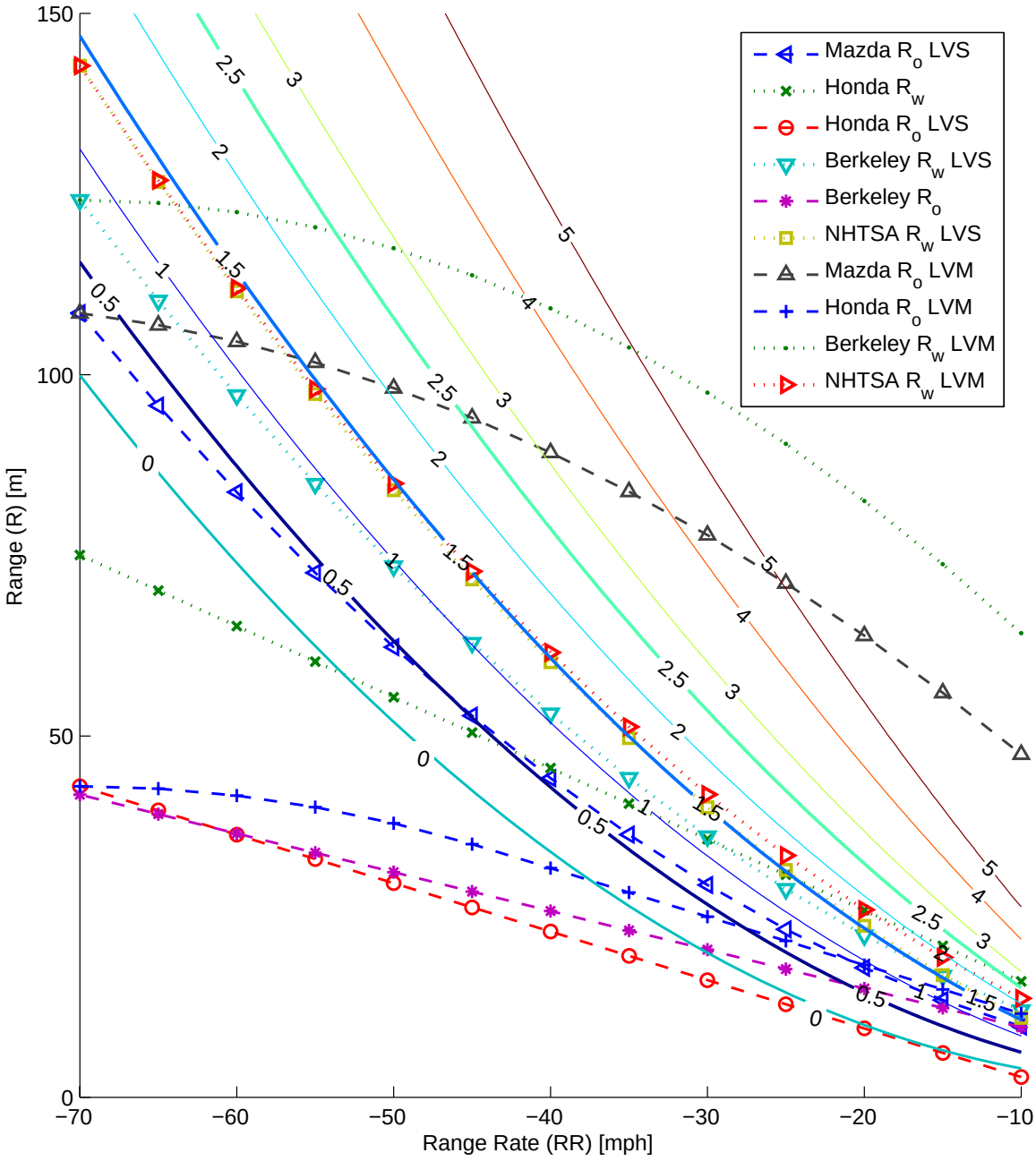


Figure 6.10: T_{lsb} Contours (solid lines) in Seconds with Various Warning (R_w, dotted lines) and Overriding (R_o, dashed lines) Boundary Curves under Scenario 1: Host Vehicle Approaches Stopped or Slow Lead Vehicle ($a_L = a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$).

While Figure 6.11 (b) shows the T_{lsb} contours after the lead vehicle has braked constantly at a_L for 1 second, i.e., $RR = a_L \cdot 1\,\text{s}$. Again the thick curves represent the T_{lsb} warning and overriding thresholds proposed in Section 6.3.6, i.e., $T_{lsb} = 0.5\,\text{s}$, $1.5\,\text{s}$, and $2.5\,\text{s}$.

The first four warning and overriding algorithms described in Section 6.2 are also plotted in Figure 6.11 for comparison. Since all previous algorithms except for the NHTSA do not take current a_L into account, they are all plotted as horizontal lines in Figure 6.11 (a) and slant lines varying with RR in Figure 6.11 (b), while the NHTSA warning curve is still close to the $T_{lsb} = 1.5\,\text{s}$ curve in both plots as expected, since it assumed that $t_r = 1.5\,\text{s}$ in addition to the assumptions used by the T_{lsb} measure. It can be observed from the figure that when the lead vehicle brakes lightly (e.g., $a_L > -3\,\text{m/s}^2$), the Berkeley warning and Mazda overriding thresholds are generally too conservative since heavier a_L was assumed in these algorithms, while the Honda warning/overriding and Berkeley overriding systems might act too late when the lead vehicle brakes hard (e.g., $a_L < -3\,\text{m/s}^2$).

The safe following distance between two vehicles traveling at the same speed could also be derived from Figure 6.11 (a). Suppose the two vehicles have the same maximum braking capability, i.e., the same tire-road friction coefficient, since they are on the same road. Considering the possibility of sudden braking of the lead vehicle at the maximum level, the host vehicle has to keep a time headway t_h no less than the driver or the overriding system's reaction time including brake system delay. Similarly the following warning and overriding criteria based on t_h can be proposed for this special case where $v_L \approx v_H$ and $a_L \approx 0$:

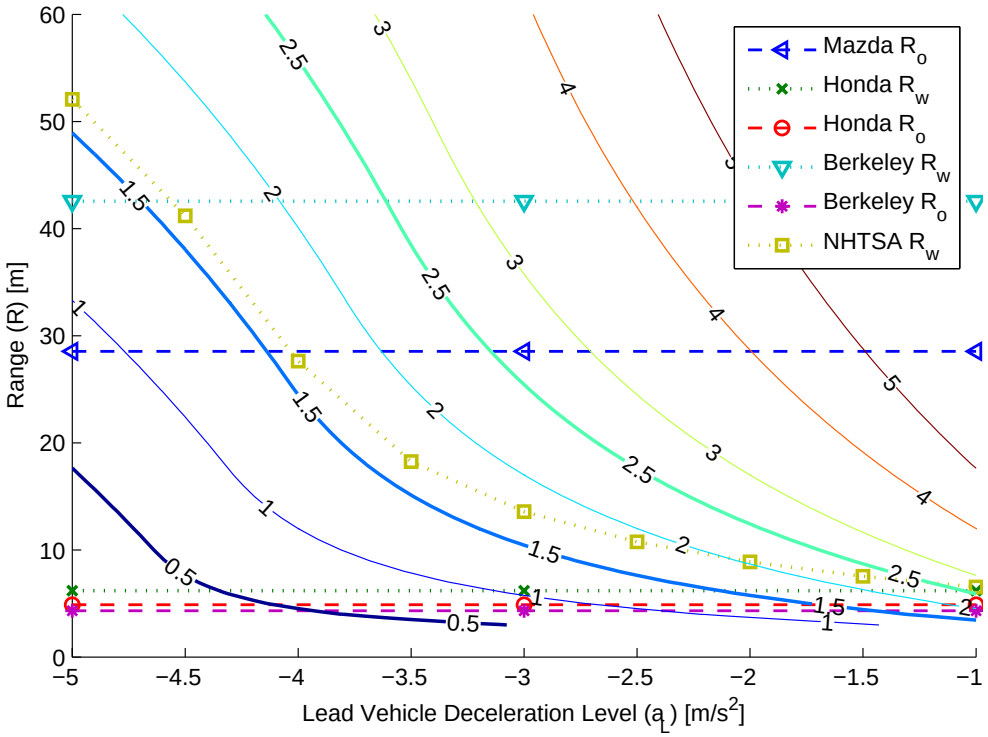
- $1\,\text{s} \le t_h < 1.5\,\text{s}$: Cautionary warning (e.g., visual signal)
- $0.5\,\text{s} \le t_h < 1\,\text{s}$: Imminent warning (e.g., visual + auditory signal)
- $t_h < 0.5\,\text{s}$: Overriding (automatic braking)

where

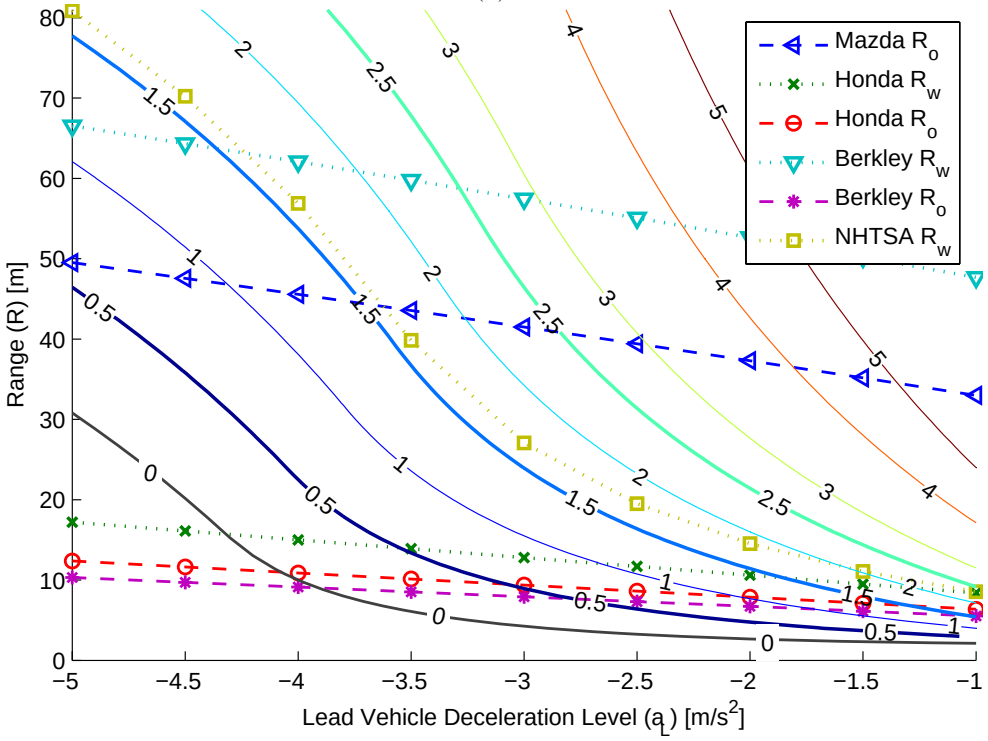
$$t_h = \frac{R - R_{min}}{v_H} \tag{6.27}$$

Smaller time thresholds are adopted in the above criteria for this tailgating mode to account for the extreme possibility of the sudden maximum braking of the lead vehicle. Again these thresholds can be adjusted according to individual driver's alertness and sensitivity. For the example plotted in Figure 6.11 (a) where $v_H = 70\,\text{mph}$ and $R_{min} = 2\,\text{m}$, the host vehicle has to keep a following distance of $49\,\text{m}$ to eliminate visual warnings, $33\,\text{m}$ to avoid imminent warnings, and braking is automatically applied when $R < 17.5\,\text{m}$.

Figure 6.12 shows another example of the T_{lsb} contours varying with a_L and R, where the moment of $v_L = 60\,\text{mph}$ and $v_H = 70\,\text{mph}$ (i.e., $RR = -10\,\text{mph}$) is examined. All previous warning and overriding thresholds except for the NHTSA are horizontal lines again here, since the different situations with different a_L but the same RR are explored, while the NHTSA warning curve is still



(a)



(b)

Figure 6.11: T_{lsb} Contours (solid lines) in Seconds with Various Warning (R_w, dotted lines) and Overriding (R_o, dashed lines) Boundary Curves under Scenario 2: Lead Vehicle Decelerates 0 s (a) and 1 s (b), respectively, where both vehicles initially travel at the same speed 70 mph ($a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$).

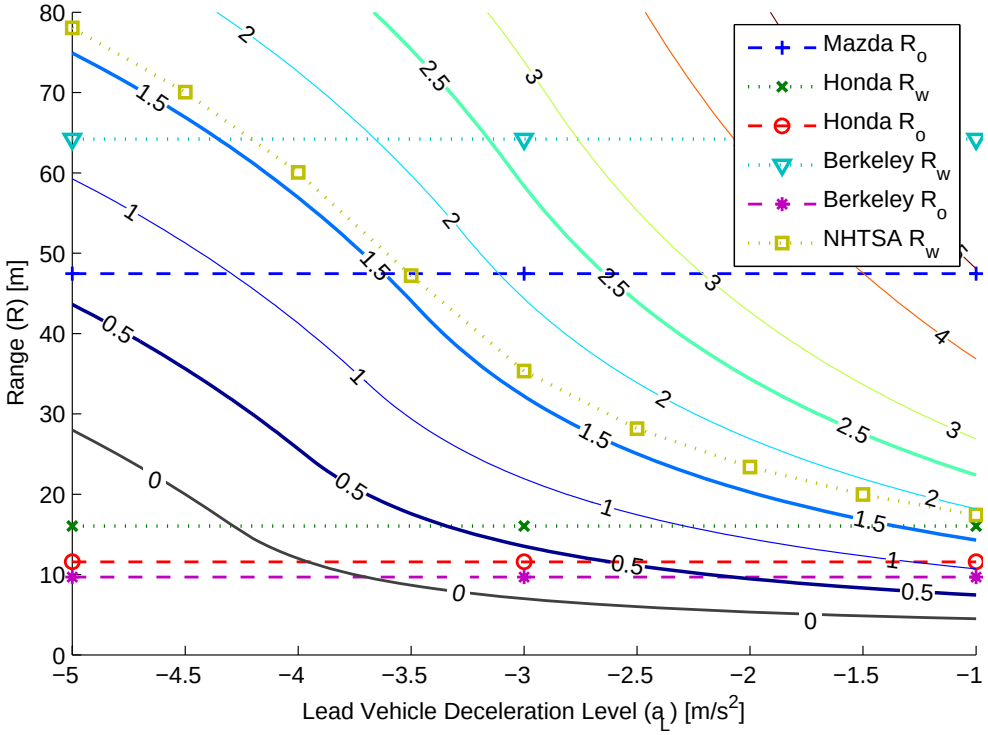


Figure 6.12: T_{lsb} Contours (solid lines) in Seconds with Various Warning (R_w, dotted lines) and Overriding (R_o, dashed lines) Boundary Curves under Scenario 2: Lead Vehicle Decelerates ($v_L = 60\,\text{mph}$, $v_H = 70\,\text{mph}$, $a_H = 0$, $a_{H_{max}} = -5\,\text{m/s}^2$).

close to the $T_{lsb} = 1.5\,\text{s}$ curve as above. Similar observations can be made in this case as above too.

From the above plots and discussions, it can be inferred that it is important for a collision avoidance system to take the lead vehicle deceleration (a_L) information into account to make better threat assessments of the current situations. Even if only a rough and slightly delayed estimate of a_L is available due to noisy sensor data, it would still be better than a constant assumption in most cases. In addition, when the lead vehicle is not braking heavily ($a_L > -3\,\text{m/s}^2$) now, the possibility of sudden heavy breaking of the lead vehicle in the near future has to be considered, which, however, seems to be the only focus of most previous algorithms.

6.4.4 Last-second Braking versus Steering

The discussions so far have only considered one way of rear-end collision avoidance through braking. Normally a road vehicle has two choices for avoiding a potential conflict with other objects ahead in the same lane, i.e., it could either brake or steer to avoid other objects in its path. The proposed T_{lsb} measure as well as the corresponding warning/overriding criteria described above considers one possibility of frontal collision avoidance through braking, which is always an available option. On the other hand, when steering is considered in collision avoidance, special attention is needed to

make sure the steering option is available. For example, the host and lead vehicle may travel on a lane with stationary objects (e.g., a fence, a wall, or parked vehicles) and/or other moving vehicles traveling on either adjacent lane, which could make the steering option unavailable at the moment. In addition, the driver or the control system must ensure that the steering maneuver is performed appropriately without overreaction, which is a major cause for highway accidents. In comparison, braking is a less dangerous maneuver and more intuitive reaction for most people in emergency situations.

When the steering option is available, a time-to-last-second-steering (T_{lss}) measure can be similarly defined as the time left for the driver or the control system to take the steering evasive action at last moment to avoid a potential collision (e.g., with a lead vehicle ahead). From the definition of T_{lss} it can be used to assess not only potential rear-end collisions but also side collisions with other vehicles or road objects, etc. However the major concern here is still focused on typical rear-end collision avoidance.

Similarly it is assumed that the lead vehicle would maintain the current acceleration level a_L constantly until it comes to a full stop if $a_L < 0$ and in this case it would remain stopped thereafter, and that the host vehicle keeps traveling at current acceleration level a_H in the same lane as the lead vehicle until the last moment when it is able to steer to an available adjacent lane to avoid collision with the lead vehicle. Therefore T_{lss} tries to estimate how long the host vehicle could still keep the current state until it has to change lanes just to avoid a potential rear-end collision with the lead vehicle.

To calculate T_{lss}, the time it takes for a lane change has to be estimated first. From an NHTSA study on naturalistic lane changes [Lee et al. 2004], the mean duration of a single lane change under urgent conditions is about 6 seconds. For the purpose of collision avoidance, at least half of the whole lane-change duration is necessary to avoid a potential frontal collision through steering, i.e., the time needed for a successful steering evasive maneuver t_s can be roughly estimated to be 3 seconds. Furthermore, since it is assumed that both vehicles would keep their current acceleration levels, respectively, the $TTC2$ measure defined in Section 6.1.1 can be used to get an approximate estimate of T_{lss}. Because the driver or the control system has to finish the steering maneuver successfully before the vehicles come into contact with each other, then T_{lss} can be simply estimated as follows under the above assumptions:

$$T_{lss} = TTC2 - t_s \tag{6.28}$$

where

$$TTC2 = f(v_H, a_H, R, RR, a_R) \tag{6.29}$$

The calculation of $TTC2$ depends on whether the lead vehicle is expected to stop before it is

hit by the host vehicle from behind. The lead vehicle stopping time t_{LS} can still be estimated by Equation 6.6, and then $TTC2$ can be solved from the following equations:

$$R = \begin{cases} v_H \cdot TTC2 + \frac{1}{2} a_H \cdot TTC2^2 + \frac{v_L^2}{2a_L} & t_{LS} \leq TTC2 \\ -RR \cdot TTC2 - \frac{1}{2} a_R \cdot TTC2^2 & t_{LS} > TTC2 \end{cases} \tag{6.30}$$

In practice, first it is assumed that $t_{LS} > TTC2$ holds and $TTC2$ is solved from the second part of Equation 6.30, where the more positive solution is taken and the other solution discarded. Then whether the condition $t_{LS} > TTC2$ holds or not can be verified. If it holds, then the computation for $TTC2$ is completed. Otherwise $TTC2$ is solved from the first part of Equation 6.30.

Under the conditions of Scenario 1 where the lead vehicle is stopped or moving slowly ($a_L = a_H = 0$) as specified in Section 6.3.2, the $TTC2$ and $TTC1$ measures are equivalent to each other, and they can be simply computed as $-R/RR$. Hence the T_{lss} contours in this case are simply straight lines going through the origin point on a R-RR plot, as shown in Figure 6.13, together with the T_{lsb} contours for comparison. By equating $T_{lss} = -R/RR - 3\,\text{s}$ with Equation 6.17, the condition for $T_{lsb} = T_{lss}$ under Scenario 1 can be solved, which is $RR = -66\,\text{mph}$. Therefore there is more time left for last-second braking rather than steering ($T_{lsb} > T_{lss}$) when the closing speed is not too high (e.g., $RR > -66\,\text{mph}$), and last-second steering may be a better choice for collision avoidance if it is available when the speed difference is rather high (e.g., $RR < -66\,\text{mph}$). This result also agrees with the CAMP experimental findings that drivers tend to make last-second steering maneuvers later than last-second braking when the closing speed is high [Kiefer et al. 2003].

A special case of Scenario 2 is also investigated here, where the vehicles initially travel at the same speed when the lead vehicle suddenly starts to brake, e.g., $v_L = v_H = 70\,\text{mph}$, $a_L < 0$, $a_H = 0$, and $a_{H_{max}} = -5\,\text{m/s}^2$. In this case the Equation 6.30 is simplified to

$$R = -\frac{1}{2} a_L \cdot TTC2^2 \tag{6.31}$$

When R is not too large, and then the T_{lss} can be roughly estimated as follows:

$$T_{lss} = \sqrt{\frac{2R}{-a_L}} - t_s \tag{6.32}$$

Therefore the T_{lss} contours are again straight lines going through the origin point on a range (R) versus lead vehicle deceleration (a_L) plot, as shown in Figure 6.14, together with the corresponding T_{lsb} contours for comparison. It can observed from the contour plot that there is usually more time left for last-second braking rather than steering under the current assumptions in most cases. Only when the initial range is large and the lead vehicle brakes very heavily would the steering be a better option, if available.

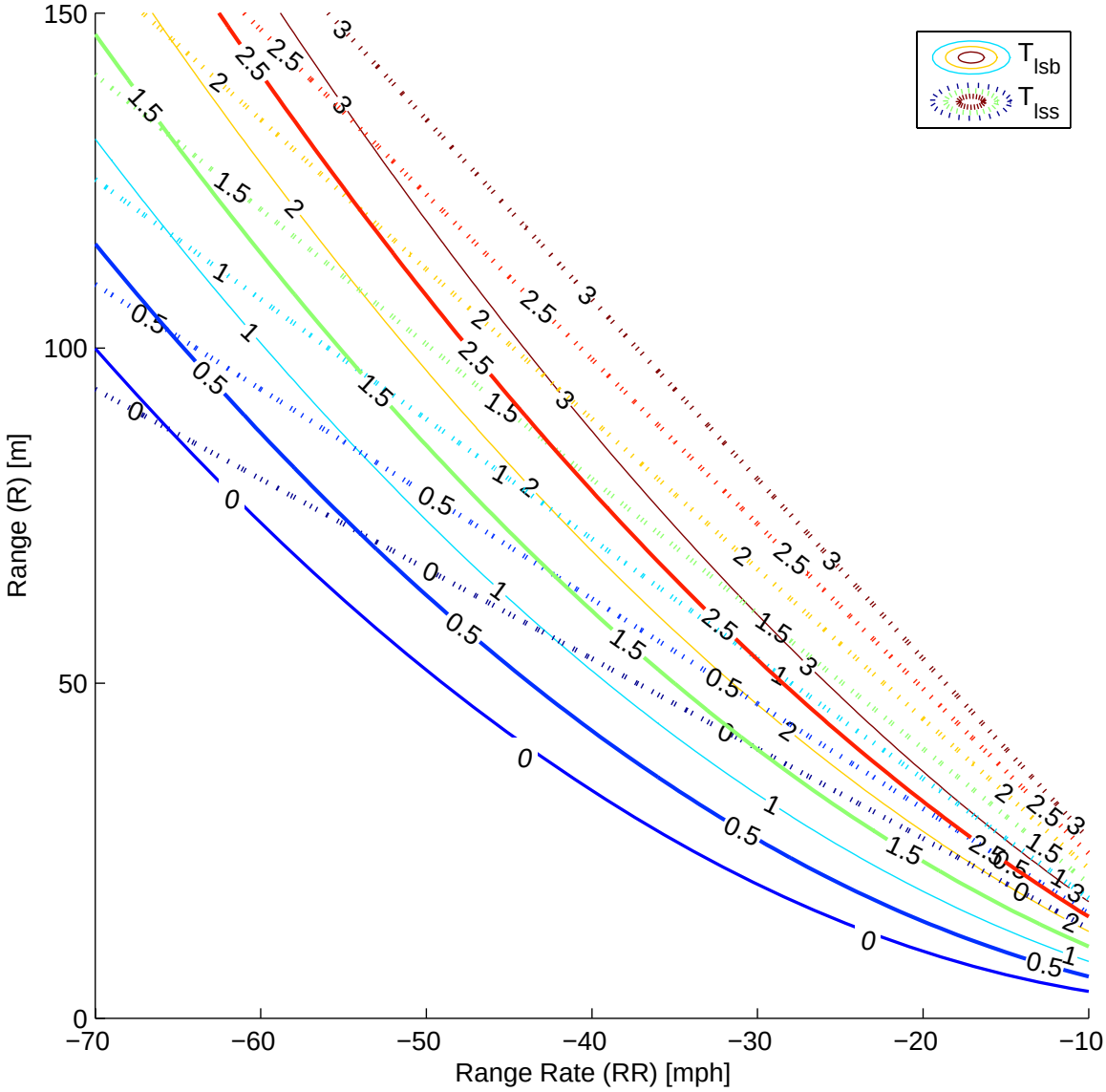


Figure 6.13: T_{lsb} and T_{lss} Contours in Seconds under Scenario 1: Host Vehicle Approaches Stopped or Slow Lead Vehicle ($a_L = a_H = 0$, $a_{H_{max}} = -5\,\mathrm{m/s^2}$).

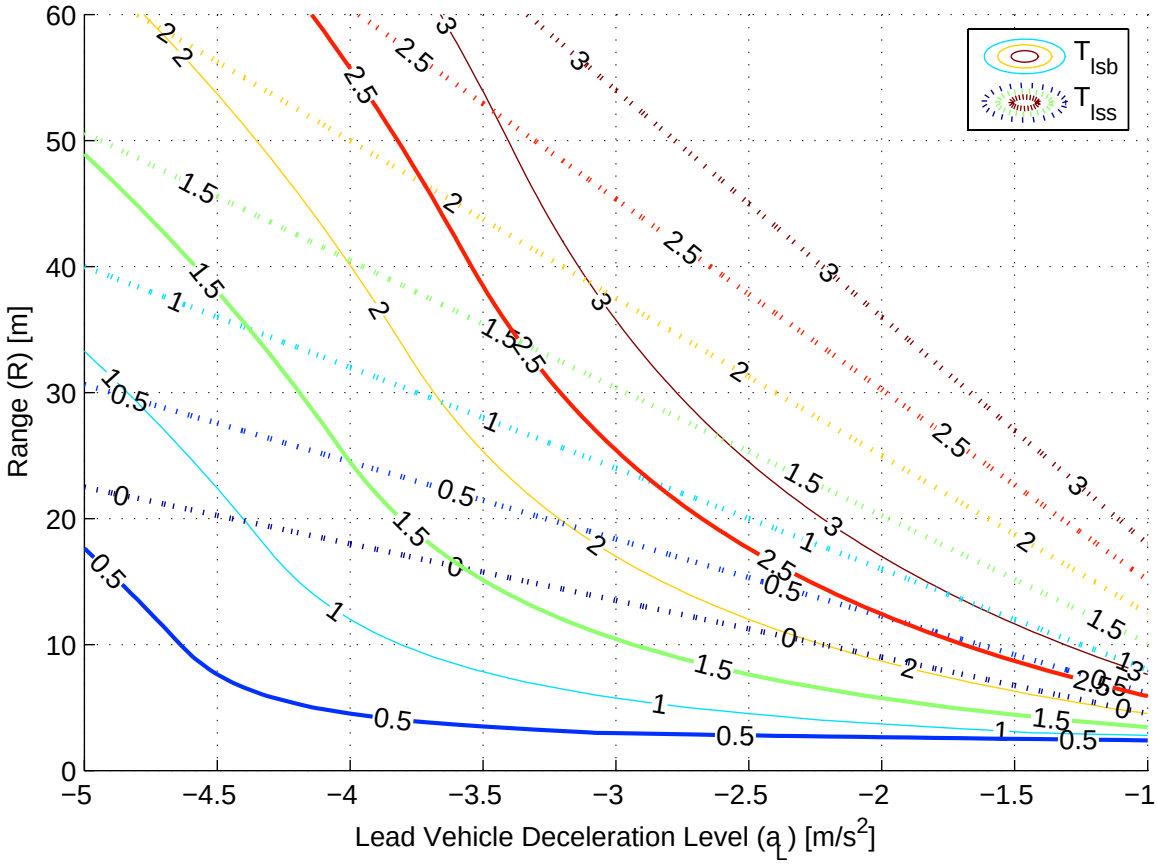


Figure 6.14: T_{lsb} and T_{lss} Contours in Seconds under Scenario 2: Lead Vehicle Just Starts to Decelerate ($v_L = v_H = 70\,\text{mph}$, $a_H = 0$, $a_{H_{max}} = -5\,\text{m/s}^2$).

In summary, braking is an effective and always available option for frontal collision avoidance. When the steering option is considered, special attention needs to be paid to ensure that the intended adjacent lane is clear and available, and the steering maneuver can be performed appropriately. When both options are available, there seems to be more time left for the braking option under the current assumptions in most cases, and only when the initial range is large and the closing speed is high (in Scenario 1) or the lead vehicle deceleration is high (in Scenario 2) would the steering option have more time margin. This is in agreement with our common driving experience. People tend to avoid a slow or stopped object ahead by changing lanes when it is still quite far away, and braking is used more often when the object is close.

6.5 Conclusion

Previous work and research related to the topic of rear-end collision avoidance systems are reviewed, including different measures defined for threat assessment, research efforts made on measuring driver reaction time as well as estimating its probability distributions, CAMP work on CWS based on their

extensive experiments with drivers' last-second maneuvers, and the NHTSA analysis and testing of its alert algorithm, etc. Several warning and overriding algorithms proposed in the literature are reviewed and summarized.

A new threat assessment measure, time-to-last-second-braking (T_{lsb}) is proposed, and its advantages over previous measures defined are discussed. It directly quantifies the danger or threat level of the current dynamic situation objectively as well as assesses the urgency level for the required evasive action, e.g., braking. It is also in agreement with human natural judgment of the urgency and severity of threats. Furthermore, new warning and overriding criteria are proposed based on the new T_{lsb} measure, which is least affected by driver behavior variability with reduced P_{miss} and P_{FA} under statistical performance analysis. The new warning and overriding criteria characterize the current dynamic situations better than the previous criteria under both Scenario 1 and Scenario 2, providing more appropriate warning and more effective overriding at the last moment. It is also shown that the information of lead vehicle deceleration level (a_L) is essential for better threat assessment of the frontal collision avoidance system, which is greatly improved with even a rough estimate of a_L. Finally, the possibility of frontal collision avoidance through steering (lane-changing) is discussed, and similarly the time-to-last-second-steering (T_{lss}) measure is proposed and compared with T_{lsb}, it turns out that braking is a better option under current assumptions in most cases.

In the future, the proposed measures and warning/overriding criteria will be applied to develop real effective collision avoidance systems. The evolutionary design synthesis methodology introduced in Chapter 2 can be applied here again to synthesize an appropriate collective sensory system as well as the corresponding sensor fusion algorithms for collision avoidance systems. The system software and hardware design can also be simultaneously evolved to get an optimal integrated system design. In addition, the warning and overriding system control strategy parameters can be automatically tuned and adapted to each individual driver by machine learning algorithms.

Chapter 7

Conclusions

This thesis investigates the application of formal engineering design synthesis methodologies to the development of sensor and control systems for intelligent vehicles. Both formal engineering design synthesis and intelligent vehicles are rather new research areas; this thesis makes an first attempt to combine these two with a series of meaningful case studies, through which great potential and interesting results are shown.

7.1 Summary

A formal engineering design synthesis methodology based on evolutionary computation is presented, with special emphasis on dealing with modern engineering design challenges, such as high or variable complexity of design solutions, multiple conflicting design objectives, and noisy evaluation results, etc., which are common to encounter when design and optimization of distributed control systems such as intelligent vehicles are considered. The efficacy of the evolutionary design synthesis method is validated through multiple different case studies. In the sensor configuration case study, a variety of novel design solutions are generated using fuzzy fitness functions with different weights and trade-off strategies selected by the human designer to reflect different engineering design trade-offs made on multiple performance measures. In the neural controller evolution case study, it is shown that the various evolved neural network controllers can achieve performances comparable to, if not better than, that of a hand-coded rule-based controller in the same simplified environment. More importantly, this automatic design synthesis method shows great potential to handle more complex design problems with a large number of design variables and multi-modal noise involved, where a good hand-coded solution may be very difficult or even impossible to obtain. In summary, the evolutionary design synthesis methodology appears promising to

- propose a variety of good, novel design solutions according to specified fuzzy fitness functions;
- deal with uncertainty in the problem efficiently;

- adapt to the collective task nature well.

In addition, various vehicle simulation tools and driver behavior models are reviewed and discussed. Multiple levels of vehicle simulation models with different computational cost and fidelity as well as necessary driver behaviors are implemented for different types of simulation experiments conducted for different research purposes. Efforts are made to try to get as much as possible out of limited computational resources, such that good candidate solutions can be generated efficiently with less computational time and human engineering effort.

Furthermore, different threat assessment measures and collision avoidance algorithms are reviewed and discussed. A new threat assessment measure, time-to-last-second-braking (T_{lsb}), is proposed, which directly characterizes human natural judgment of the urgency and severity of threats in terms of time. Based on driver reaction time experimental results, new warning and overriding criteria are proposed in terms of the new T_{lsb} measure, and the performance is analyzed statistically in terms of two typical sample pre-crash traffic scenarios. Less affected by driver behavior variability, the new criteria characterize the current dynamic situations better than the previous ones, providing more appropriate warning and more effective overriding at the last moment. Finally, the possibility of frontal collision avoidance through steering (lane-changing) is discussed, and similarly the time-to-last-second-steering (T_{lss}) measure is proposed and compared with T_{lsb}.

7.2 Limitation and Future Directions

The power of the evolutionary engineering design synthesis methodology presented here awaits more validation studies on more complex design problems defined in more complicated environments. The set of design problems for which the automatic design synthesis method excels or matches other methods in terms of both performance and computational cost needs to be further clarified and categorized. The proposed automatic design synthesis method can be further improved to make it more efficient and pertinent to each specific engineering design synthesis problem. For instance, the hierarchical evaluation test idea mentioned in Section 4.3.2 could be tested in the future to improve computational efficiency.

All the results presented in this thesis are based on computer simulations; it would be desirable and meaningful to implement and validate the sensor and control system designs discussed in this thesis on real intelligent vehicles in the future. In turn those validation studies can help to further improve the different vehicle and driver models used to simulate more realistic and complicated traffic scenarios without slowing down the simulation speed too much. It is also desirable to simulate more sample traffic scenarios to further testify the system designs comprehensively.

The discussions on the new threat assessment measures and warning/overriding criteria are based on purely theoretical computations with basic kinematic equations and statistical predictions with

simplified assumptions. The real traffic scenarios are expected to be much more complicated. For instance, effective multiple target tracking and identification of the most dangerous object have to be implemented from noisy sensor data, and also the sensor fusion and synchronization in both time and space have to be solved if multiple sensors are used. Therefore the effectiveness of the new criteria is yet to be proved and validated with real human drivers driving physical test vehicles in real traffic environments.

Finally, the evolutionary design synthesis methodology can be applied again to develop an optimal integrated hardware and software system design for real collision avoidance systems. All the results presented in this thesis will be useful, especially the threat assessment measures and the warning/overriding control system strategies.

Bibliography

[Aczél 1966] Aczél, J. (1966), *Lectures on Functional Equations and Their Applications*, Academic Press, New York, NY.

[Alm and Nilsson 1994] Alm, H. and Nilsson, L. (1994), "Changes in driver behaviour as a function of handsfree mobile phones—A simulator study", *Accident Analysis and Prevention*, 26(4):441–451.

[Antonsson and Cagan 2001] Antonsson, E. K. and Cagan, J., (Eds.) (2001), *Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge, U.K.

[Antonsson et al. 2003] Antonsson, E. K., Zhang, Y., and Martinoli, A. (2003), "Evolving engineering design trade-offs", In *Proceedings of the 15th International Conference on Design Theory and Methodology*, number DTM-48676, Chicago, IL, ASME.

[Bäck 1996] Bäck, T. (1996), *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York, NY.

[Bäck et al. 2000] Bäck, T., Fogel, D. B., and Michalewicz, Z., (Eds.) (2000), *Evolutionary Computation*, Institute of Physics Publishing, Bristol, U.K.

[Bakker et al. 1987] Bakker, E., Nyborg, L., and Pacejka, H. B. (1987), "Tyre modelling for use in vehicle dynamics studies", SAE Technical Paper No. 870421.

[Bakker et al. 1989] Bakker, E., Pacejka, H. B., and Lidner, L. (1989), "A new tire model with an application in vehicle dynamics studies", SAE Technical Paper No. 890087.

[Bando et al. 1995] Bando, M., Hasebe, K., Nakayama, A., Shibata, A., and Sugiyama, Y. (1995), "Dynamical model of traffic congestion and numerical simulation", *Physical Review E*, 51(2):1035–1042.

[Bentley 1999] Bentley, P. J., (Ed.) (1999), *Evolutionary Design by Computers*, Morgan Kaufmann, London, U.K.

[Bishop 2005] Bishop, R. (2005), "Intelligent vehicle R&D: A review and contrast of programs worldwide and emerging trends", *Annales des Télécommunications*, 60(3–4):228–263.

[Bonabeau et al. 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999), *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, New York, NY.

[Brunson et al. 2002] Brunson, S. J., Kyle, E. M., Phamdo, N. C., and Preziotti, G. R. (2002), "Alert algorithm development program: NHTSA rear-end collision alert algorithm", Final Report DOT HS 809 526, Applied Physics Laboratory, The Johns Hopkins University, National Highway Traffic Safety Administration.

[Bugajska and Schultz 2000] Bugajska, M. D. and Schultz, A. C. (2000), "Co-evolution of form and function in the design of autonomous agents: Micro air vehicle project", In *Proceedings of Workshop on Evolution of Sensors (GECCO 2000)*, pages 240–244, Las Vegas, NV.

[Burgett et al. 1998] Burgett, A. L., Carter, A., Miller, R. J., Najm, W. G., and Smith, D. L. (1998), "A collision warning algorithm for rear-end collisions", In *Proceedings of 16th International Technical Conference on Enhanced Safety of Vehicles*, number 98-S2-P-31, pages 566–587, Washington, DC.

[Campbell et al. 2003] Campbell, B. N., Smith, J. D., and Najm, W. G. (2003), "Examination of crash contributing factors using national crash databases", Technical Report DOT HS 809 664, John A. Volpe National Transportation Systems Center, National Highway Traffic Safety Administration.

[Chang et al. 1985] Chang, M. S., Messer, C. J., and Santiago, A. J. (1985), "Timing traffic signal change intervals based on driver behavior", *Transportation Research Record*, 1027:20–30.

[Chen and Ulsoy 2001] Chen, L.-K. and Ulsoy, A. G. (2001), "Identification of a driver steering model, and model uncertainty, from driving simulator data", *Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control*, 123(4):623–629.

[Cho and Hedrick 1989] Cho, D. and Hedrick, J. K. (1989), "Automotive powertrain modeling for control", *Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control*, 111(4):568–576.

[Correll and Martinoli 2004] Correll, N. and Martinoli, A. (2004), "Modeling and optimization of a swarm-intelligent inspection system", In *Proceedings of the 7th Symposium on Distributed Autonomous Robotic System (DARS)*, pages 351–360, Toulouse, France, Springer-Verlag.

[Correll and Martinoli 2005] Correll, N. and Martinoli, A. (2005), "Modeling and analysis of beaconless and beacon-based policies for a swarm-intelligent inspection system", In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2488–2493, Barcelona, Spain.

[Curry et al. 2005] Curry, R. C., Greenberg, J. A., and Kiefer, R. J. (2005), "Forward collision warning requirements project—Task 4 final report: NADS versus CAMP closed-course comparison examining 'last-second' braking and steering maneuvers under various kinematic conditions", Final Report DOT HS 809 925, Crash Avoidance Metrics Partnership, National Highway Traffic Safety Administration.

[Darwin 1859] Darwin, C. (1859), *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, John Murray, London, U.K.

[De Jong 1975] De Jong, K. A. (1975), *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan, Ann Arbor, MI.

[De Jong et al. 1995] De Jong, K. A., Spears, W. M., and Gordon, D. F. (1995), "Using Markov chains to analyze GAFOs", In Whitley, L. D. and Vose, M. D., (Eds.), *Foundations of Genetic Algorithms 3*, pages 115–137, Morgan Kaufmann, San Francisco, CA.

[Doi et al. 1994] Doi, A., Butsuen, T., Niibe, T., Yakagi, T., Yamamoto, Y., and Seni, H. (1994), "Development of a rear-end collision avoidance system with automatic braking control", *JSAE Review*, 15(4):335–340.

[Easton and Burdick 2005] Easton, K. and Burdick, J. (2005), "A coverage algorithm for multi-robot boundary inspection", In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain.

[Epiney 2004] Epiney, L. (2004), "Modeling of car dynamics and realistic traffic patterns in Webots", Technical Report SWIS-SU1, Swarm-Intelligent Systems Group (SWIS), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.

[Fitzpatrick and Grefenstette 1988] Fitzpatrick, J. M. and Grefenstette, J. J. (1988), "Genetic algorithms in noisy environments", *Machine Learning*, 3(2-3):101–120.

[Fleming 2001] Fleming, W. J. (2001), "Overview of automotive sensors", *IEEE Sensors Journal*, 1(4):296–308.

[Fogel 1992] Fogel, D. B. (1992), *Evolving Artificial Intelligence*, PhD thesis, University of California, San Diego, CA.

[Fogel et al. 1966] Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966), *Artificial Intelligence Through Simulated Evolution*, John Wiley and Sons, Inc., New York, NY.

[Fujita et al. 1995] Fujita, Y., Akuzawa, K., and Sato, M. (1995), "Radar brake system", In *Proceedings of the 1995 Annual Meeting of ITS America*, volume 1, pages 95–101, Washington, DC.

[Fuller 1981] Fuller, R. G. C. (1981), "Determinants of time headway adopted by truck drivers", *Ergonomics*, 24(6):463–474.

[Gartner et al. 1997] Gartner, N. H., Messer, C. J., and Rathi, A., (Eds.) (1997), *Traffic Flow Theory: A State of the Art Report*, Transportation Research Board, Revised Monograph on Traffic Flow Theory.

[Gazis et al. 1960] Gazis, R., Herman, R., and Maradudin, A. (1960), "The problem of the amber signal light in traffic flow", *Operations Research*, 8(1):112–130.

[Gers et al. 2002] Gers, F. A., Schraudolph, N. N., and Schmidhuber, J. (2002), "Learning precise timing with LSTM recurrent networks", *Journal of Machine Learning Research*, 3:115–143.

[Gipps 1981] Gipps, P. G. (1981), "A behavioral car-following model for computer simulation", *Transportation Research Part B: Methodological*, 15(2):105–111.

[Goldberg 1989] Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

[Goldberg 2002] Goldberg, D. E. (2002), *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, Boston, MA.

[Green 2000] Green, M. (2000), "'How long does it take to stop?' Methodological analysis of driver perception-brake times", *Transportation Human Factors*, 2(3):195–216.

[Guo et al. 2004] Guo, K., Ding, H., Zhang, J., Lu, J., and Wang, R. (2004), "Development of a longitudinal and lateral driver model for autonomous vehicle control", *International Journal of Vehicle Design*, 36(1):50–65.

[Harned et al. 1969] Harned, J. L., Johnston, L. E., and Scharpf, G. (1969), "Measurement of tire brake force characteristics as related to wheel slip (antilock) control system design", SAE Technical Paper No. 690214.

[Hayes et al. 2003] Hayes, A. T., Martinoli, A., and Goodman, R. M. (2003), "Swarm robotic odor localization: Off-line optimization and validation with real robots", *Robotica*, 21(4):427–441.

[Hedrick et al. 1993] Hedrick, J. K., McMahon, D. H., and Swaroop, D. (1993), "Vehicle modeling and control for automated highway systems", Technical Report UCB-ITS-PRR-93-24, Longitudinal Control Group, University of California, Berkeley, California PATH Program.

[Helbing et al. 2001] Helbing, D., Hennecke, A., Shvetsov, V., and Treiber, M. (2001), "Master: Macroscopic traffic simulation based on a gas-kinetic, non-local traffic model", *Transportation Research Part B: Methodological*, 35(2):183–211.

[Helbing et al. 2002] Helbing, D., Hennecke, A., Shvetsov, V., and Treiber, M. (2002), "Micro- and macro-simulation of freeway traffic", *Mathematical and Computer Modelling*, 35(5–6):517–547.

[Hertz et al. 1991] Hertz, J., Krogh, A., and Palmer, R. G. (1991), *Introduction to the Theory of Neural Computation*, Perseus Books, Reading, MA.

[Hess and Modjtahedzadeh 1990] Hess, R. A. and Modjtahedzadeh, A. (1990), "A control theoretic model of driver steering behavior", *IEEE Control Systems Magazine*, 10(5):3–8.

[Hoess et al. 2004] Hoess, A., Slater, S., Sjögren, A., Beutner, A., Bullinger, W., Möhle, K., Maier, D., Rasshofer, R., Saroldi, A., Miglietta, M., Rohling, H., Lübbert, U., Schiementz, M., Garrod, A., Rickett, B., Pycock, D., Hoare, E., Castanie, F., Doerfler, R., and Brandt, M. (2004), "Multifunctional automotive radar network (RadarNet)", Final Report D40, RadarNet Consortium, Information Societies Technology.

[Holland 1975] Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[Honda 2005] Honda (2005), "Acura RL luxury performance sedan combines sleek styling, super handling all-wheel drive™ (SH-AWD) and a comprehensive list of technology features", http://corporate.honda.com/press/article.aspx?id=2005081757434, Honda Press Releases.

[Jiang et al. 2001] Jiang, R., Wu, Q., and Zhu, Z. (2001), "Full velocity difference model for a car-following theory", *Physical Review E*, 64(1):017101.

[Johansson and Rumar 1971] Johansson, G. and Rumar, K. (1971), "Drivers' brake reaction times", *Human Factors*, 13(1):23–27.

[Kageyama 2006] Kageyama, Y. (2006), "Nissan develops gas pedal safety feature", Associated Press.

[Kiefer et al. 2005a] Kiefer, R. J., Cassar, M. T., Flannagan, C. A., Jerome, C. J., and Palmer, M. D. (2005a), "Forward collision warning requirements project—Tasks 2 and 3a final report: Surprise braking trials, time-to-collision judgments, and 'first look' maneuvers under realistic rear-end crash scenarios", Final Report DOT HS 809 902, Crash Avoidance Metrics Partnership, National Highway Traffic Safety Administration.

[Kiefer et al. 2003] Kiefer, R. J., Cassar, M. T., Flannagan, C. A., LeBlanc, D. J., Palmer, M. D., Deering, R. K., and Shulman, M. A. (2003), "Forward collision warning requirements project: Refining the CAMP crash alert timing approach by examining 'last-second' braking and lane change maneuvers under various kinematic conditions", Final Report DOT HS 809 574, Crash Avoidance Metrics Partnership, National Highway Traffic Safety Administration.

[Kiefer et al. 2005b] Kiefer, R. J., LeBlanc, D. J., and Flannagan, C. A. (2005b), "Developing an inverse time-to-collision crash alert timing approach based on drivers' last-second braking and steering judgments", *Accident Analysis and Prevention*, 37(2):295–303.

[Kiefer et al. 1999] Kiefer, R. J., LeBlanc, D. J., Palmer, M. D., Salinger, J., Deering, R. K., and Shulman, M. A. (1999), "Development and validation of functional definitions and evaluation procedures for collision warning/avoidance system", Final Report DOT HS 808 964, Crash Avoidance Metrics Partnership, National Highway Traffic Safety Administration.

[Koza 1992] Koza, J. R. (1992), *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge, MA.

[Koza 1994] Koza, J. R. (1994), *Genetic Programming II*, The MIT Press, Cambridge, MA.

[Lee 2002] Lee, C.-Y. (2002), *Efficient Automatic Engineering Design Synthesis via Evolutionary Exploration*, PhD thesis, California Institute of Technology, Pasadena, CA.

[Lee and Antonsson 2000] Lee, C.-Y. and Antonsson, E. K. (2000), "Variable length genomes for evolutionary algorithms", In Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., and Beyer, H.-G., (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, page 806, Las Vegas, NV, Morgan Kaufmann.

[Lee and Peng 2005] Lee, K. and Peng, H. (2005), "Evaluation of automotive forward collision warning and collision avoidance algorithms", *Vehicle System Dynamics*, 43(10):735–751.

[Lee et al. 2004] Lee, S. E., Olsen, E. C. B., and Wierwille, W. W. (2004), "A comprehensive examination of naturalistic lane-changes", Final Report DOT HS 809 702, Virginia Tech Transportation Institute, National Highway Traffic Safety Administration.

[Li et al. 2006] Li, L., Wang, F.-Y., and Zhou, Q. (2006), "Integrated longitudinal and lateral tire/road friction modeling and monitoring for vehicle motion control", *IEEE Transactions on Intelligent Transportation Systems*, 7(1):1–19.

[Lipson and Pollack 2000] Lipson, H. and Pollack, J. B. (2000), "Automatic design and manufacture of robotic lifeforms", *Nature*, 406:974–978.

[Lubashevsky et al. 2003] Lubashevsky, I., Wagner, P., and Mahnke, R. (2003), "Rational-driver approximation in car-following theory", *Physical Review E*, 68(5):056109.

[Lutz 2005] Lutz, N. (2005), "Evolution of a robot neural controller for keep lane behavior", Technical Report EDRL-SWIS-SU3, Engineering Design Research Laboratory, California Institute of Technology, Pasadena, CA.

[MacAdam 1981] MacAdam, C. C. (1981), "Application of an optimal preview control for simulation of closed-loop automobile driving", *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(6):393–399.

[MacAdam 2003] MacAdam, C. C. (2003), "Understanding and modeling the human driver", *Vehicle System Dynamics*, 40(1–3):101–134.

[Martin and Stewart 2000] Martin, K. and Stewart, C. V. (2000), "Real time tracking of borescope tip pose", *Image and Vision Computing*, 10(18):795–804.

[Martinoli 1999] Martinoli, A. (1999), *Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Control Strategies*, PhD thesis, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland.

[Martinoli et al. 2002] Martinoli, A., Zhang, Y., Prakash, P., Antonsson, E. K., and Olney, R. D. (2002), "Towards evolutionary design of intelligent transportation systems", In *Proceedings of the 11th International Symposium of the Associazione Tecnica dell'Automobile on Advanced Technologies for ADAS Systems (ATA 2002)*, Siena, Italy.

[McEvoy et al. 2005] McEvoy, S. P., Stevenson, M. R., McCartt, A. T., Woodward, M., Haworth, C., Palamara, P., and Cercarelli, R. (2005), "Role of mobile phones in motor vehicle crashes resulting in hospital attendance: A case-crossover study", *BMJ (British Medical Journal)*, 331(7514):428.

[McRuer et al. 1977] McRuer, D. T., Allen, R. W., Weir, D. H., and Klein, R. H. (1977), "New results in driver steering control models", *Human Factors*, 19(4):381–397.

[Mende et al. 2005] Mende, R., Behrens, M., and Milch, S. (2005), "A 24 GHz ACC radar sensor", In *Proceedings of the International Radar Symposium (IRS 2005)*, Berlin, Germany.

[Mendel 1866] Mendel, G. (1866), "Versuche über pflanzen-hybriden", In *Verhandlungen des Naturforschenden Vereins*, volume 4, Brünn, Czech Republic.

[Michel 2004] Michel, O. (2004), "Webots: Professional mobile robot simulation", *Journal of Advanced Robotic Systems*, 1(1):39–42.

[Miglino et al. 1995] Miglino, O., Lund, H. H., and Nolfi, S. (1995), "Evolving mobile robots in simulated and real environments", *Artificial Life*, 2(4):417–434.

[Miller and Goldberg 1996] Miller, B. L. and Goldberg, D. E. (1996), "Optimal sampling for genetic algorithms", In Dagli, C. H., Akay, M., Chen, C. L. P., Fernandez, B. R., and Ghosh, J., (Eds.), *Proceedings of the Artificial Neural Networks in Engineering Conference (ANNIE 96)*, volume 6, pages 291–297, St. Louis, MO, ASME Press.

[Mitchell 1996] Mitchell, M. (1996), *An Introduction to Genetic Algorithms*, The MIT Press, Cambridge, MA.

[Modjtahedzadeh and Hess 1993] Modjtahedzadeh, A. and Hess, R. A. (1993), "A model of driver steering control behavior for use in assessing vehicle handling qualities", *Transactions of the ASME: Journal of Dynamic Systems, Measurement, and Control*, 115(3):456–464.

[Mondada et al. 1994] Mondada, F., Franzi, E., and Ienne, P. (1994), "Mobile robot miniaturisation: A tool for investigation in control algorithms", In *Proceedings of the Third International Symposium on Experimental Robotics*, pages 501–513, Kyoto, Japan, Springer-Verlag.

[Newman 2000] Newman, M. (2000), "Applied mathematics: The power of design", *Nature*, 405:412–413.

[NHTSA 2006] NHTSA (2006), "Traffic safety facts 2004: A compilation of motor vehicle crash data from the fatality analysis reporting system and the general estimates system", Final Report DOT HS 809 919, National Center for Statistics and Analysis, National Highway Traffic Safety Administration.

[Nolfi and Floreano 2000] Nolfi, S. and Floreano, D. (2000), *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*, The MIT Press, Cambridge, MA.

[Nolfi and Parisi 2002] Nolfi, S. and Parisi, D. (2002), "Evolution of artificial neural networks", In Arbib, M. A., (Ed.), *Handbook of Brain Theory and Neural Networks*, pages 418–421, The MIT Press, Cambridge, MA.

[Olson 1989] Olson, P. L. (1989), "Driver perception response time", SAE Technical Paper No. 890731.

[Otto and Antonsson 1991] Otto, K. N. and Antonsson, E. K. (1991), "Trade-off strategies in engineering design", *Research in Engineering Design*, 3(2):87–104.

[Patel et al. 2001] Patel, M., Honavar, V., and Balakrishnan, K., (Eds.) (2001), *Advances in the Evolutionary Synthesis of Intelligent Agents*, The MIT Press, Cambridge, MA.

[Phillips and Harbor 2000] Phillips, C. L. and Harbor, R. D. (2000), *Feedback Control Systems*, Prentice Hall, Upper Saddle River, NJ, fourth edition.

[Polychronopoulos et al. 2004] Polychronopoulos, A., Tsogas, M., Amditis, A., Scheunert, U., Andreone, L., and Tango, F. (2004), "Dynamic situation and threat assessment for collision warning systems: The EUCLIDE approach", In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2004)*, pages 636–641, Parma, Italy.

[Pugh et al. 2005] Pugh, J., Zhang, Y., and Martinoli, A. (2005), "Particle swarm optimization for unsupervised robotic learning", In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2005)*, pages 92–99, Pasadena, CA.

[Pursula 1999] Pursula, M. (1999), "Simulation of traffic systems—An overview", *Journal of Geographic Information and Decision Analysis*, 3(1):1–8.

[Rechenberg 1965] Rechenberg, I. (1965), "Cybernetic solution path of an experimental problem", Technical report, Royal Aircraft Establishment, Farnborough, Hants, U.K., Library Translation No. 1122.

[Schwefel 1977] Schwefel, H.-P. (1977), *Numerische Optimierung von Computer—Modellen mittels der Evolutionsstrategie*, volume 26 of *Interdisciplinary Systems Research*, Birkhäuser, Basel, Switzerland.

[Scott and Antonsson 1998] Scott, M. J. and Antonsson, E. K. (1998), "Aggregation functions for engineering design trade-offs", *Fuzzy Sets and Systems*, 99(3):253–264.

[Scott and Antonsson 2000] Scott, M. J. and Antonsson, E. K. (2000), "Using indifference points in engineering decisions", In *Proceedings of the 11th International Conference on Design Theory and Methodology*, number DTM-14559, Baltimore, MD, ASME.

[Segel 1956] Segel, L. (1956), "Theoretical prediction and experimental substantiation of the response of the automobile to steering control", *Proceedings of The Institution of Mechanical Engineers, Automobile Division*, 7:310–330.

[Seiler et al. 1998] Seiler, P., Song, B., and Hedrick, J. K. (1998), "Development of a collision avoidance system", In *Proceedings of 1998 SAE Conference*, pages 97–103, Detroit, MI, SAE Technical Paper No. 980853.

[Sens et al. 1989] Sens, M. J., Cheng, P. H., Wiechel, J. F., and Guenther, D. A. (1989), "Perception/reaction time values for accident reconstruction", SAE Technical Paper No. 890732.

[Sharke 2003] Sharke, P. (2003), "Smart cars", *Mechanical Engineering*, 125(3):50–52.

[Sivak et al. 1982] Sivak, M., Olson, P. L., and Farmer, K. M. (1982), "Radar measured reaction times of unalerted drivers to brake signals", *Perceptual and Motor Skills*, 55:594.

[Sivak et al. 1981] Sivak, M., Post, D. V., Olson, P. L., and Donohue, R. J. (1981), "Driver responses to high-mounted brake lights in actual traffic", *Human Factors*, 23(2):231–235.

[Taborek 1957] Taborek, J. J. (1957), *Mechanics of Vehicles*, Penton.

[Taoka 1989] Taoka, G. T. (1989), "Brake reaction times of unalerted drivers", *Institute of Transportation Engineers (ITE) Journal*, 59(3):19–21.

[Treiber et al. 2000] Treiber, M., Hennecke, A., and Helbing, D. (2000), "Congested traffic states in empirical observations and microscopic simulations", *Physical Review E*, 62(2):1805–1824.

[Treiber et al. 2006] Treiber, M., Kesting, A., and Helbing, D. (2006), "Delays, inaccuracies and anticipation in microscopic traffic models", *Physica A*, 360(1):71–88.

[van der Horst 1990] van der Horst, A. R. A. (1990), *A Time-based Analysis of Road User Behavior in Normal and Critical Encounters*, PhD thesis, Delft University of Technology, Delft, The Netherlands.

[van Eldik Thieme and Pacejka 1971] van Eldik Thieme, H. C. A. and Pacejka, H. B. (1971), "The tire as a vehicle component", In Clark, S. K., (Ed.), *Mechanics of Pneumatic Tires*, National Bureau of Standards Monograph 122, chapter 7, pages 545–839, U.S. Department of Commerce, Washington, DC.

[Versino and Gambardella 1997] Versino, C. and Gambardella, L. M. (1997), "Learning real team solutions", In Weiss, G., (Ed.), *Distributed Artificial Intelligence Meets Machine Learning*, pages 40–61, Springer-Verlag, Berlin, Germany.

[Whitcomb and Milliken 1956] Whitcomb, D. W. and Milliken, W. F. (1956), "Design implications of a general theory of automobile stability and control", *Proceedings of The Institution of Mechanical Engineers, Automobile Division*, 7:367–391.

[Wolpert and Macready 1995] Wolpert, D. H. and Macready, W. G. (1995), "No free lunch theorems for search", Technical Report SFI-TR-95-02-010, The Santa Fe Institute, Santa Fe, NM.

[Wong and Litt 2004] Wong, E. and Litt, J. S. (2004), "Autonomous multi-agent robotics for inspection and repair of propulsion systems", In *Proceedings of AIAA First Intelligent Systems Technical Conference*, number AIAA-2004-6364, Chicago, IL.

[Wong 2001] Wong, J. Y. (2001), *Theory of Ground Vehicles*, John Wiley and Sons, Inc., New York, NY, third edition.

[Yang et al. 2003] Yang, L., Yang, J. H., Feron, E., and Kulkarni, V. (2003), "Development of a performance-based approach for a rear-end collision warning and avoidance system for automobiles", In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2003)*, pages 316–321, Columbus, OH.

[Yao 1999] Yao, X. (1999), "Evolving artificial neural networks", *Proceedings of the IEEE*, 87(9):1423–1447.

[Zhang 2005] Zhang, Y. (2005), "Towards evolution of collective sensory systems for intelligent vehicles", Annual report, Engineering Design Research Laboratory, California Institute of Technology, METRANS.

[Zhang et al. 2006] Zhang, Y., Antonsson, E. K., and Martinoli, A. (2006), "Evolving neural controllers for collective robotic inspection", In Abraham, A., Baets, B., Köppen, M., and Nickolay, B., (Eds.), *Applied Soft Computing Technologies: The Challenge of Complexity*, Advances in Soft Computing, pages 721–733, Springer, Germany.

[Zhang et al. 2003a] Zhang, Y., Martinoli, A., and Antonsson, E. K. (2003a), "Evolutionary design of a collective sensory system", In *Proceedings of the AAAI 2003 Spring Symposium on Computational Synthesis*, pages 283–290, Palo Alto, CA.

[Zhang et al. 2003b] Zhang, Y., Martinoli, A., Antonsson, E. K., and Olney, R. D. (2003b), "Evolution of sensory configurations for intelligent vehicles", In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2003)*, pages 351–356, Columbus, OH.

[Zuvich 2000] Zuvich, T. (2000), "Vehicle dynamics for racing games", In *Proceedings of the Game Developers Conference (GDC 2000)*, San Jose, CA.