DEVELOPMENT OF PATH TRACKING SOFTWARE FOR AN AUTONOMOUS STEERED-WHEELED ROBOTIC VEHICLE AND ITS TRAILER

By

ARTURO L. RANKIN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1997

UMI Number: 9801142

UMI Microform 9801142
Copyright 1997, by UMI Company. All rights reserved.

This microform edition is protected against unauthorized
copying under Title 17, United States Code.

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

ACKNOWLEDGEMENTS

The author would like to express his gratitude to the members of his supervisory committee, Dr. Carl Crane, Dr. Joseph Duffy, Dr. Gary Matthew, Dr. Paul Mason, and Dr. Sencer Yeralan, for their guidance and support. Special thanks go to Dr. Crane who brought the author on board the autonomous navigation project and has provided invaluable advice and templates for graphic simulation programs.

This work would have not been possible without the support of Wright Laboratory, Tyndall Air Force Base, Florida. Thanks go to Mr. Al Nease and the rest of his staff for the suggestions given on our trips to their robotics facility.

The author's work is not intended to stand alone. It is piece of a puzzle, put together by the autonomous navigation team at the Center for Intelligent Machines and Robots. Particular thanks go to the project manager, David Armstrong, for his invaluable input on every phase of the project, and office mate David Novick, for his insightful advice. Special thanks go to the rest of the current team, Jeff Wit, Takao Okui, Dean Hutchinson, Allen Senior, and Eric Jackson, for their collaboration and dedication.

Rob Murphy is to be acknowledged for animating the Kawasaki Mule 500. Because of his excellent work, the author's simulation demonstrations look good.

Finally, special thanks go to his wife, Donna Marie Samuels Rankin, who has provided the moral support and fire needed to finish this dissertation.

TABLE OF CONTENTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

DEVELOPMENT OF PATH TRACKING SOFTWARE FOR AN AUTONOMOUS STEERED-WHEELED ROBOTIC VEHICLE AND ITS TRAILER

By

Arturo L. Rankin

May 1997

Chairperson: Dr. Carl D. Crane III
Major Department: Mechanical Engineering

Wright Laboratory, at Tyndall AFB, Florida, has contracted the University of Florida to develop autonomous navigation systems for a variety of robotic vehicles, capable of performing tasks associated with the location and removal of bombs and mines. One of the tasks involves surveying plots of land for buried munitions. The navigation subtasks include vehicle positioning, path planning, path following, and obstacle avoidance.

The author's master's research involved implementing, on a navigation test vehicle (NTV), a path planner that generates the shortest collision-free path between two vehicle configurations. Subsequent to this work, a second planner was implemented that generates an efficient route for surveying a polygon shaped field. These routes ensure 100% of the surface area is covered, provided that the path is accurately followed.

Accuracy in path following is critical to the task. There are hundreds of acres that currently require surveying. The sites are typically divided into regions, where each mission can take up to 4.5 hours. These sites are usually surveyed using parallel rows. By improving the accuracy of path following, the distance between the rows can be increased to nearly the detection width of the ground penetrating sensors.

There are two ways to improve the accuracy of path following: improve vehicle positioning and reduce control errors. The NTV is currently capable of calculating its position in real-time, accurate to within 6 cm on an average. The author's doctoral research involves developing a strategy that takes advantage of an accurate positioning system and ensures accurate path following for any steered-wheeled vehicle over a range of speeds (0-4.5 m/s).

This strategy involves a process for standardizing and smoothing a given path, calculating a target position in real-time, combining the desirable features of a high-level PID and adaptive pure pursuit steering controller, autonomously calibrating the steering feedback data, autonomously tuning control parameters, and scheduling the control gains as a function of the vehicle speed. Auto-tuning is accomplished with a relay tuner and a golden section search (GSS) optimization. This strategy was demonstrated in simulation and implemented on the NTV.

CHAPTER 1
INTRODUCTION

The following problem statement was presented at the author's oral Ph.D. qualifier's examination on March 4, 1996, as a proposal for dissertation research.

Problem Statement

Given: A piecewise linear path, either generated by a path planner, prerecorded during manual or remote operation, or synthesized as a set of line segments and circular arcs,

Develop: A high-level, geometry-based control strategy that can be applied to any all-terrain steered-wheeled vehicle (that already has the functions of steering, throttle, braking, and gear-shifting under low-level control) and accomplishes accurate path following over a wide range of vehicle speeds (0-4.5 m/s). Autonomous land vehicles (ALVs) differ from automatic guided (factory) vehicles in that the dynamics of the system are more difficult to model, since the wheel/surface interaction changes as the terrain changes. If the center of the rear axle is chosen as the vehicle control point, then the functions of steering and propulsion control can be decoupled and performed independently. Two geometry-based steering control methods, pure pursuit and proportional-integral-derivative (PID), will be evaluated in simulation using several indices of performance. A weighted control solution will be implemented on an autonomously operated navigation test vehicle (NTV). PID control will be implemented for velocity control. A program will be written to autonomously tune all control parameters. Gain scheduling will be used, based on the current vehicle velocity.

This proposal was approved by the author's supervisory committee on March 4, 1996, with one change. The task of implementing and auto-tuning a velocity controller was eliminated. The remainder of this thesis will outline the author's work in accomplishing these tasks.

Project Background

The author has worked for the Center for Intelligent Machines and Robotics (CIMAR) from the Spring of 1991 until the present. During this time, CIMAR has been under contract by Wright Laboratory, Tyndall Air Force Base, Florida, to develop an autonomous navigation system capable of navigating a variety of robotic vehicles. A system has been successfully developed and tested on an automated Kawasaki Mule all-terrain navigation test vehicle (NTV), a six- wheeled John Deere autonomous survey vehicle (ASV),

Figure 1.1. Navigation Test Vehicle (NTV).

Figure 1.2. Autonomous Survey Vehicle (ASV).

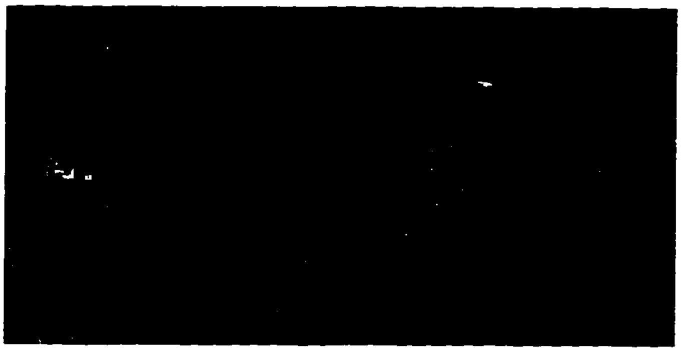

Figure 1.3. Autonomous D7G bulldozer.

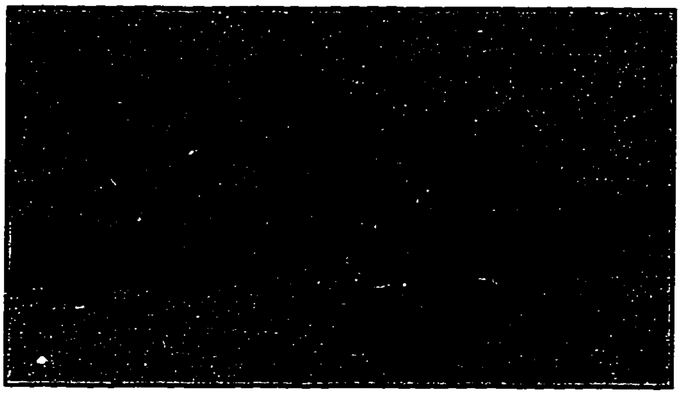

Figure 1.4. Autonomous John Deere excavator.

a John Deere excavator, and a D7G military bulldozer (see Figures 1.1-1.4). Positioning is currently accomplished by the integration of an inertial navigation system (INS) with a global positioning system (GPS). Other subtasks of the navigation system include path planning, path following, and a sonar-based obstacle avoidance capability for the NTV.

The focus of some of the research Wright Laboratory is funded for is to develop a capability of locating and removing unexploded buried munitions. The ASV is capable of surveying a polygon shaped site, achieving near 100% coverage, while towing behind it a sensor package (containing an array of cesium vapor magnetometers and ground penetrating radar) to detect magnetic objects. A Caterpillar excavator is remotely navigated to the excavation site from a command station and the buried object is removed via remote control. Control algorithms are currently being written to autonomously control a high-speed Eagle Picher excavator.

Accurate and smooth control of the vehicle is essential to accurately characterizing the area surveyed. The author's contribution to the project has been in the area of motion planning and motion execution. His master's work focused primarily on developing efficient algorithms for calculating the shortest-distance path between two robot configurations. His doctoral work involved developing a suite of programs that enables a steered-wheeled robot to achieve accurate path following.

Research Requirements

The goal of this research was to develop a path following strategy for autonomous steered-wheeled robotic vehicles where accuracy is easily attained. To meet this goal, the specified path should not only be safe and efficient, it should be achievable. The vehicle must be able to evaluate how well it is performing the task. And the manual tuning of control parameters should be replaced by autonomous procedures.

By applying a self-tuning algorithm during autonomous vehicle operation on a sample plot of land, accurate path following should be achieved in a timely manner (within 4 hours would be ideal, as beyond this time the system would need to be refueled). It is expected that this self-tuning algorithm would need to be run only once and repeated only when there was a large change in the vehicle dynamics or the terrain.

For purposes of this research, accurate path following is defined as a position deviation of less than 0.1 meters over the sampling period. The position deviation is defined as the perpendicular (or lateral) distance from the current calculated vehicle position to the path. Here, the focus is bounding the tracking error due to the control methodology. The sensor-related positioning errors were not considered here. This work was restricted to reducing the error between where the vehicle thought that it was (real-time Kalman filtered positioning solution) with respect to the path, not where the vehicle actually was (post-processed GPS positioning solution) with respect to the path.

As a proof of concept, accurate path following was demonstrated in simulation on a Silicon Graphics workstation over a range of speeds (0-4.5 m/s) for forward vehicle motion only. Three types of steering control were autonomously tuned and evaluated: 1) proportional, integral, and derivative (PID) control applied to the error in the vehicle's heading, 2) pure pursuit control, and 3) a weighted PID/pure pursuit solution. Actual tests on the NTV were performed at a constant velocity of 1.34 m/s.

Assumptions

The control parameters obtained from the self-tuning procedures are expected to vary based on the terrain characteristics in the vehicle's environment (sand, grass, pavement, dry, wet, rocky, etc.) and the speed at which the vehicle is operated. It is assumed that the terrain at the site of operation is uniform and the elevation is fairly constant. The test site used at the University of Florida meets this criteria.

Motivation

The motivation for this research is two part: a general lack of information in the literature that makes it difficult to compare the merits of various geometry based control methods and the tedium required in tuning the previously developed strategy. A typical reference from the literature will contain only simulation results, where the assumptions used in updating the vehicle's position and heading are not specified. The results section will typically provide a small graph with one arc labeled planned path and another arc labeled actual path. The reader is left to interpret the accuracy of the path tracking with a magnifying glass. The author's view is that each reporting group should provide with their results the average deviation, maximum deviation, and standard deviation for position and heading along straight and curved path segments (where the system is initially perturbed) over a range of vehicle speeds. The control strategy chosen for this research allows for the development of such tools for both real-time and post-mission evaluation.

The parameters related to the control of the NTV under a previously developed path tracking strategy by the author, involved the following:

1) A subgoal removal (pop) distance.
2) A constant distance between subgoals on straight path segments.
3) A constant angle between subgoals around curved path segments.
4) A distance to the next corner where the vehicle is directed to slow down.
5) A distance from the last corner where the vehicle returns to normal operating speed.
6) A gain proportional to the sharpness of the next corner for determining a safe desired velocity.
7) Steering position proportional, integral, and derivative gains.
8) Throttle position proportional, integral, and derivative gains.

The list is long enough to make manual tuning tedious and extremely inefficient. The current settings do not meet the standard set by the author for accurate path following. Deviation from the path around corners was a problem in this implementation. The parameters, for the most part, were tuned for one particular vehicle speed (1.34 m/s). The inadequacies of manual tuning make self-tuning or autonomous tuning an attractive alternative.

Benefits of this Research

This research would be of benefit to autonomous land vehicles used for surveying, mowing, or plowing large and relatively flat plots of land where 100% coverage is desired. Also, this research is applicable to autonomous vehicles for which obstacle avoidance capabilities are limited (such as the ASV) and the pre-planned path is guaranteed to exist in free-space. The control strategy outlined in this proposal allows for the evaluation of other geometry-based, subgoal-following control methods in a "plug and play" fashion.

Required Equipment and Supplies

The equipment required to accomplish this research includes the automated NTV as the vehicle platform, a Modular Azimuth Positioning System (MAPS) integrated with an global positioning system operating in phase-differential mode (PDGPS) for positioning, the MAPS for vehicle heading and velocity information, a Silicon Graphics workstation for developing the control strategy and simulating auto-tuning and accurate path following, and a large open outdoor area for conducting vehicle tests. All outdoor tests were performed at Flavet Field, University of Florida.

The NTV consist of a Kawasaki Mule 500 all-terrain vehicle, outfitted with actuators on the four functions of steering, throttle, brake, and transmission, two on-board computer systems, a generator, and two battery back-up units. The NTV is a platform upon which to develop and test sensors, software, and sensor/software modules used to accomplish remote operation and autonomous navigation.

The primary computer system consists of a VME backplane chassis, a floppy and hard disk, two Force 30 CPU boards, two Force 33 CPU boards, and a VIPC610 Quad

IndustryPack (IP) carrier board (manufactured by GreenSpring Computers). The IP board provides an interface with the VME bus for two IP-Servo modules and two IP Analog to Digital Input/Output (ADIO) modules. A secondary VME computer is used to acquire data from a stereo vision system. This information is serially communicated to the primary computer. Both computers operate under the VxWorks operating system.

The advantage of using multiple CPU boards is that a large number of programs can be run in parallel at a high rate (10 Hz), each communicating information needed by other programs across the backplane of the computer. At boot up, a block of shared memory is set up in random access memory (RAM) on a Force 30 board. Programs running on other boards have access to this block of memory across the bus. Similarly, a block of memory is used where each program checks to see if it has any messages.

Chapter 2 contains a broad overview of the control methods applied to mobile robot control, from factory operated automatic guided vehicles (AGVs), to autonomous land vehicles (ALVs), or also called unmanned ground vehicles (UGVs), to autonomous highway vehicles (AHVs). Chapter 3 discusses the author's strategy for specifying a path to be tracked. Paths can be specified in a variety of forms by either the user or the path planning programs. This flexibility predicates the need for path smoothing. The path following algorithm does not assume that the user will always specify a completely achievable path.

Chapter 4 introduces the control strategy proposed for this research. Chapter 5 outlines the strategy for developing a computer simulation to demonstrate path tracking and auto-tuning. And finally, Chapter 6 contains some results regarding the implementation of this strategy and conclusions that can be drawn from it.

CHAPTER 2
REVIEW OF THE LITERATURE

What are Automated Vehicles?

Automated Vehicles are Mobile Robots

The simple answer to the question "What are automated vehicles?" is, "They are mobile robots." The question then arises, "What is a mobile robot?". A mobile robot is a robot that has mobility in the sense that it is not bolted down to the floor or to another surface. Mobile robots have been described as fourth generation robots. The first three generations of robots were industrial robots used for 1) pick and place maneuvers, 2) welding, and 3) painting and assembly. Mobile robots are used to accomplish repetitive tasks normal executed by humans, such as, the transportation of materials or people from one place to another. It is anticipated that mobile robots will become commonplace at home, in office environments, and on roadways.

Mobile robots can be classified on the basis of the space in which they operate. One-dimensional robots run on railways or in pipes. Two-dimensional robots operate on flat surfaces such as a floor or wall. Three-dimensional mobile robots operate underwater, on terrain of varying elevation, and in space. The functions responsible for mobility for a mobile robot or automated vehicle are placed under computer control. Note that a mobile robot does not necessarily need to have wheels to accomplish mobility. The mode of mobility can be suction cups, legs, or whatever a designer decides to use.

Mechanical Configurations

Cox and Wilfong [Cox90] provide a good review of the various types of mechanical configurations that mobile robots can have. The two major mechanical configurations for wheeled land vehicles are steered-wheeled vehicles and differential-drive vehicles. Steered-wheeled vehicles usually have a fixed rear axle and one or two steerable front wheels. Steered-wheeled mobile robots with two steerable front wheels have steering configurations identical to automobiles. The dynamics and kinematics of a steered-wheeled mobile robot are usually modeled using the tricycle configuration. Four-wheeled vehicles are considered in this category by approximating the two steered wheels with a single steered wheel at the center of the front axle.

Differential-drive vehicles commonly have tracks, such as military tanks and construction excavators, or two side wheels with any number of free rotating castor wheels for stability. Steering is accomplished with different command velocities at each wheel. The mobility advantage that differential drive vehicles have over steered-wheeled vehicles is the ability to turn about a point. In effect, they possess a zero minimum radius of curvature. Steered-wheeled vehicles and differential-drive vehicles both lack the ability of moving in the direction of the axis through the fixed axle.

A typical omnidirectional vehicle contains four Ilonator wheels, giving it the capability of arbitrary motion in any direction. An Ilonator wheel consists of twelve rollers mounted on a hub. Each roller is rotated 45° from the wheel orientation. With such an arrangement, the wheel can move in any direction without slipping.

Steered-Wheeled Mobile Robots

A vehicle's minimum radius of curvature is directly related to the range of motion allowed by the steering mechanism. Car-like vehicles have Ackerman steering linkages. Stable steering at slow speeds requires all the wheels on a vehicle to completely roll, without any lateral slip. For all the wheels on a vehicle to purely roll, they must all move along their rolling axes. When steering around a turn, pure rolling will take place when the pure slip axis of each wheel intersects at a point called an instant center. The rolling axes of the front wheels have different orientations with respect to the vehicle's current heading. The Ackerman linkage is designated to give the correct steering angle on each steerable wheel in a turn.

A vehicle's position or pose in the plane is represented by three parameters (x,y,θ), two for translation and one for orientation. However, for a steered-wheeled vehicle (containing an Ackerman linkage), there are only two-degrees of freedom for control, the steering angle and vehicle velocity. Such systems are referred to as nonholonomic. The motion of a nonholonomic vehicle is constrained by the relation,

$$-\sin\theta\, dx + \cos\theta\, dy = 0, \tag{2.1}$$

where θ is the current orientation of the robot vehicle and (dx,dy) are incremental displacements in position. A nonholonomic equality constraint is a nonintegratable equation involving the configuration parameters and their derivatives. A nonholonomic mobile robot can achieve any pose (x,y,θ) in two-dimensional space but the combinations of incremental translations (dx,dy) are limited.

Car-like mobile robots are nonholonomic since their constraint equation is not integratable. As a result, the analysis of their kinematics, dynamics, and control is complex. In general, car-like robots have three degrees-of-freedom (DOF) in position and orientation but only two DOF for motion control. The command variables for a steered-wheel vehicle are usually the steering angle and vehicle velocity. The command variables for a differential-drive vehicle are usually right wheel velocity and left wheel velocity.

Kinematics of a Mobile Robot

The kinematics of a mobile robot are expressed in a similar manner to that of manipulators. The kinematics of a manipulator are referred to as forward and reverse kinematics or analysis. In forward analysis, the joint angles (displacements) are used to determine the unique position and orientation of the end effector. In reverse analysis, the position and orientation of the end effector is used to determine all possible combinations of joint angles that would place the end effector in such a pose.

With an ALV, or any mobile robot, differential displacements of the driving wheels and/or the steering angle are used to determine the current pose of the vehicle. This is referred to as positioning or position estimation. The calculated position is an estimate when the slip between the wheels and the surface cannot be exactly quantified. The kinematics of a mobile robot is thus instrumental in determining its position and heading. They are also used when estimating a robot's current position and heading in computer simulations. The inverse kinematics of a mobile robot involves obtaining a set of control variables that will effect specific wheel displacements and place the robot at a specific position and orientation. The specific positions and orientation can be generated using a path planning program.

Muir and Neuman [Mui87] offer a comprehensive overview of the kinematics of steered-wheeled, differential-drive, omnidirectional, and ball-wheeled vehicles. Alexander and Maddocks's [Ale89] analysis of the kinematics of wheeled mobile robots is aimed at determining what planar rigid motions are possible, given a certain wheel configuration, and what steering and wheel velocities are required to perform these motions. The conditions required to guarantee pure rolling are derived. In an earlier work, Alexander and Maddocks [Ale88] derived equations that govern the motion of steered-wheeled vehicles when pure rolling occurs. Optimal steering around tight corners is considered.

The influence of friction between an automated vehicle and the driven surface on turning performance has been examined by Takano [Tak90]. For the case of steered-wheeled mobile robots, only the cornering force (frictional force acting in the direction perpendicular to the robot's motion when the wheel rotating direction differs from that of the vehicle motion) has a significant effect on turning performance.

Zhao and BeMent [Zha92] used differential geometric theory to prove controllability for three 2-DOF wheeled mobile robots and three 3-DOF wheeled mobile robots, where the DOF is defined as the number of variables used to specify the systems configuration minus the number of independent equations of constraint.

They also modeled the kinematics and dynamics of the so-called synchro-drive mobile robot vehicle. These systems contain three wheels locked together by a set of belts, chains, or concentric shafts with bevel gears. Two motors are used, one to apply velocity to the wheels and another to steer the wheels. The widely used Cybermation and Denning mobile robots are synchro-drive vehicles. They further summarized the requirements for a mobile robot to use nonlinear feedback control for path tracking.

Position Estimation for Mobile Robots

Dead reckoning is the process of estimating the vehicle's position and heading by measuring wheel rotations. The cost of dead reckoning sensors is relatively low. Positioning error, however, usually compounds over time. Dead reckoning is not usually precise enough for outdoor navigation at high speeds.

Other common methods of positioning for mobile robots include the use of gyros to determine the orientation of the robot with respect to magnetic north and accelerometers to determine the acceleration of the robot along several normal axes. (These accelerations can be integrated once to obtain an instantaneous velocity and a second time to determine the total displacement with respect to the starting position). Global positioning systems (GPS) use receivers that communicate with several orbiting satellites. The time required for a receiver to obtain data from each satellite in view is used to obtain the straight-line distance to each satellite. A method similar to triangulation is used to obtain the position and elevation of the ALV. The orientation of the ALV cannot be directly determined from GPS.

For an overview of position systems commonly used for ALV's and other mobile robots see Feng et al. [Feng94]. Here, the author is concerned only with developing a control method that reduces the control error. To this end, the accuracy in positioning is a separate issue, left to be addressed elsewhere. For the scope of this work, the author is only concerned about positioning to the extent that it is continuous and smooth. The primary method used for positioning for ALV's developed at CIMAR is outlined by Crane et al. [Cra95].

Types of Automated Vehicles

There are three general classes of autonomous vehicles: automatic guided vehicles (AGVs), autonomous land vehicles (ALVs), and automated highway vehicles (AHVs). Automated vehicle control has been effectively applied to factory automation (with the advent of AGVs), and autopilots by the airlines industry. Velocity control has been applied to automobiles by the use of cruise control. Autonomous vehicle navigation is now being applied to ALVs and AHVs.

Automatic guided vehicles, commonly referred to as mobile robots, have been successfully used in factories for over a decade, mainly for material delivery. The primary purpose of AGVs is to relieve humans of repetitious tasks, such as delivering material, patrolling a building, or scrubbing a floor. AGVs move at slow speeds and can be free-ranging or constrained to follow semi-permanent paths. Factory AGVs usually follow painted lines, buried wires, or permanent tracks.

Free-ranging AGVs require sensors to keep track of its location and relationship to its surroundings. The bulk of AGV research has used encoders, sonar, infrared and laser light, radar, and vision sensors for free-ranging navigation. The current thrust in AGV research is to develop free-ranging robots that make factories more flexible environments.

Dead reckoning is commonly used to estimate an AGVs current position and heading. The cumulative errors in dead reckoning can come from wheel slippage, an imprecise measure of a wheel's rolling radius, and separation of the two contact surfaces. The majority of AGVs have two driven wheels and one or more free-rotating castor wheels. Some are steered-wheeled and kinematically modeled as tricycles. Others can translate in every direction and are thus termed omnidirectional. AGVs operate in controlled environments, at slow speeds, and in general pose a low risk to humans.

Optimizing the control of AGVs is beyond the scope of this research. However, to the extent that some AGV control strategies may be partially applicable to ALVs, a set of the most interesting ones will be evaluated in the section *How are Autonomous Vehicles Controlled?*. This work is limited to the operation of steered-wheeled ALVs.

Autonomous land vehicles are designed to navigate on well defined roads, on poorly defined roads, and on off-road terrain. Often, very little is known about the environment in which an ALV will be expected to operate in. The thrust of research in the area of ALVs centers around image processing, object recognition, path planning, and obstacle avoidance. Artificial intelligence techniques are often applied to improve image quality and computational speed. Because scenes tend to be complex, ALVs typically operate at speeds below 24 km/h. With ALVs, exact path following is not always critical. The need to quickly react and avoid obstacles in the environment is often a higher concern.

Vehicle dynamics do not generally play an important role in vehicle control for ALVs since they do not move very fast. Centrifugal and coriolis forces are rarely considered in developing control laws. Vehicle statics, on the other hand, presents a larger challenge for vehicles that rely on dead reckoning for position estimation. An ALV moves by frictional reaction forces acting on the driven surface. Any slippage between the wheels and the surface contributes to the uncertainty of the vehicle's location.

The control of wheeled ALVs is currently a popular research activity and there are several reasons for this. Wheeled ALVs tend to be energy efficient and are structurally simple. Furthermore, there exists a wealth of technical information available from the automobile industry on automotive engineering.

Highway automation is currently under study as a remedy to recurring traffic congestions and frequent accidents that account for thousands of deaths and injuries in the U.S. each year. It is anticipated that the increase in the number of vehicles operated in the U.S. in the next two decades will place a considerable strain on the capacity and safety of the present highway system. Highway automation may be a partial solution to improving highway safety and capacity.

Highway automation involves two interrelated areas of research: automatic vehicle control and traffic control strategy. Central to both of these areas is the need to gather information from sensors, intelligence or decision making, and control.

The deployment of autonomous vehicles on highways is considered a viable highway automation option and is a part of the Advanced Vehicle Control Systems (AVCS) and the Intelligent Vehicle Highway Systems (IVHS). It is believed that the number of auto accidents can be drastically reduced by eliminating human error and by taking advantage of faster response times available to the automated vehicle. It is further estimated that a high speed lane could increase highway capacity by up to 800 percent if automated traffic control systems became a reality.

Automated highway vehicles (AHVs) are typically controlled using two independent control laws: one for lateral (steering) control and one for longitudinal (velocity) control.

Lateral control involves keeping the vehicle within a specific lane. The lane is usually identified by boundary stripes or a marker placed in the middle of the lane. Lateral control is typically accomplished using steering commands only. Longitudinal control refers to controlling the velocity of an AHV during platooning or vehicle-tracking.

The disadvantage of the AHV control methods is that they are limited to path tracking where the path curvature is small, a weakness that eliminates their use for path tracking on nonhighway roads and off-road for ALVs [Smi91].

How are Automated Vehicles Controlled?

Control Architectures

There are two general types of control architectures used for real-time systems such as autonomous vehicles: centralized and decentralized. In centralized control, the decision making responsibility is partitioned as a carefully ordered decomposition of specialized tasks. A hierarchy exists where, for example, the high level modules concentrate on sensor data acquisition and motion planning and the subservient low level (motion controller) modules are responsible for carrying out the will of the motion planner.

In decentralized control, the decision-making responsibility is partitioned as a loosely ordered cooperation between shared tasks. Decision making capabilities are placed in the motion controller, where steering and speed commands are computed directly from sensor data. Here, the system is not a true control hierarchy. The lower level modules are not subordinate to the higher level modules.

In practice, most control architectures fall somewhere along the spectrum between being completely centralized and being completely decentralized. The author's work utilizes a centralized control architecture, as it is applied to the execution of an explicit path by an autonomous land vehicle (ALV). For an overview of decentralized control, see Payton and Bihari's discussion of the vehicle called ALV [Pay91]. ALV originally referred to a 20,000 lb, 8-wheeled mobile robot built by Martin Marietta thru funding by DARPA. The ALV project ran from 1984-1988. Today, the acronym ALV is used more often to refer to a class of mobile robots, not the DARPA funded ALV.

How do Humans Control Vehicles?

Many investigators attempt to model the control of autonomous vehicles in such a way that the perceived "way" in which humans drive is emulated. There is a variation of explanations, however, on the dominant behaviors in human driving, and consequently on the actual mechanics of human driving. Here is a sample of explanations the author has compiled.

> *In contrast to path planning, a human driver bypasses the computation of a path trajectory and turns the steering wheel in direct reaction to an observed heading angle and range* [Keh91].

> *When a human drives an automobile, he perceives the local environment through eye-brain cooperation. He selects a way point ahead of the vehicle as an intermediate subgoal toward his global goal, and steers the vehicle to attain the subgoal* [Che87].

> *When a human drives a car along a road, he or she cares most about keeping a car within the road boundaries. Vehicle speed is not changed to follow a time history of vehicle positions. Speed is changed in response to some higher level of characteristics such as road curvature, proximity of obstacles, or conditions of the road, vehicle states and the human driver* [Shi92].

> *When a skilled car-driver makes a turn, the steering wheel angle is smoothly increased to some maximum angle, then smoothly reduced back to neutral position as the car comes out of the turn. It seems reasonable that paths generated for an*

> *autonomously guided vehicle should likewise have turn segments which permit such smooth steering commands for guiding the vehicle around turns in the path* [Nel89].
>
> *When humans drive their vehicles within the confines of a passageway, they do not necessary change the planned route when another vehicle crosses their path. Typically any traverse of the road map is continually adjusted in terms of stopping, slowing down, or accelerating in attempt to avoid obstacles* [Gri90].

Reflexive vs. Explicit Path Following

A path tracking or following algorithm's job is simply to keep an autonomous vehicle on a path until a goal point is reached. Although the job description of a path tracker is simple, the way in which the task is accomplished has been difficult to solve at high speeds and over a range of velocities. There are two general types of path tracking: explicit path following or reflexive vehicle control.

In reflexive path following, the path of the vehicle is perceived in real-time via stereo vision sensors or other sensors. In road following, a subgoal is generated from the image at some fixed lead distance ahead of the vehicle that represents a midpoint of the perceived lane. In vehicle-tracking, a subgoal is generated that represents a vision cue on the back of the lead vehicle. Reflexive control is often applied in AHV research in the form of lateral and longitudinal control.

Lateral control entails autonomous lane following on highways, accomplished by generating steering commands that minimize the lateral error in a vehicle's position. These methods usually assume constant velocity and address only steering control. Longitudinal control is used to autonomously control each vehicle in a platoon in a way such that the distance between vehicles remains constant. Longitudinal control methods address only speed control.

A relatively new area of reflexive path following uses artificial potential fields. This method places artificial repelling forces on all obstacles and an attracting force on the goal point. The robot vehicle is guided through the resulting field to the goal, avoiding all obstacles along the way. Artificial potential fields can be used to plan paths if the environment is known a priori, or to navigate reflexively as objects are detected. Artificial potential field methods, in their basic form, are applicable to only differential-drive and omnidirectional mobile robots.

In explicit path following, an explicit path, either generated off-line by a path planner or previously recorded by driving the course, is provided to a path tracking algorithm. Here also, a subgoal is normally generated at a lead distance ahead of the vehicle and on the path. Common methods used for steering control include PID and other classical control methods, pure pursuit, and curve fitting, usually with a quintic polynomial.

The path tracking task for both of these cases is the same: develop a control law that will direct the vehicle to pass through the current subgoal. The emphasis of the author's work is placed on accurately path tracking an explicit path. This literature review, however, will include methodologies used in path tracking a perceived path since many of those methodologies can be directly applied to path tracking an explicit path.

Explicit Path Following

There are two general ways in which an explicit path is tracked: reference posture methods and real-time subgoal following methods. Reference posture methods [Nel88, Wil90, Kan88] of path tracking use a reference path generator (RPG) to obtain a set of reference postures $(x,y,\theta)^t$ from a given path off-line. A sequence of reference postures is given to a vehicle at a constant rate, and a path tracking algorithm operates to move the vehicle through these postures. The error (e_x, e_y, e_θ) between the current posture and a given reference posture are continuously multiplied by gains to determine the command vehicle steering angle and throttle position. These methods boast of inch level accuracy at slow velocities (2-6 m/s); however, tracking accuracy degrades as speed increases.

These methods are generally applied to only indoor mobile robots where high precision in positional control and safety are high priorities and the speed of operation has a lower priority. These robots are generally used to follow predetermined routes in factories or offices and are called automatic guided vehicles (AGVs).

Many of these routes are made up solely by line and circular arc segments. The curvature of a path at a line arc transition point is discontinuous, however. Such discontinuities can contribute to tracking error. Nelson [Nel89] has proposed using polar polynomials in place of circular arc segments so that the defined path to be followed is a continuous curvature path. Reference postures are obtained from a RPG that lie on the line segments and polar polynomials.

Shin and Singh [Shi90] divide the navigation problem into four subproblems: world perception, path planning, path generation, and path tracking. In the path generation stage, clothoid curves are used to convert *objective points* (supplied by a course planner) into the actual path to be tracked.

Clothoid curves are a family of curves that are continuous with respect to posture, tangent direction, and curvature. They are distinct in that their curvature varies linearly with the length of the curve. Shin and Ollero [Shi95] convert *objective points* into path segments of finer detail using clothoid as well as B-spline curves. The limitation of these methods is that the smoothed path can differ significantly from the planned path to the extent that obstacle space may be invaded.

Boukas [Bou92] has compared three methods of control for a differential-drive robot containing two drive wheels: a PID controller, a state feedback controller, and a sliding mode controller. The dynamics of this AGV are nonlinear. The state feedback controller is based on a linearized model, while the PID controller and sliding mode controller are based on reducing the movements of the vehicle to linear and angular displacements.

Methods of Control

This section of the literature review will be limited to control methods for autonomous steered-wheeled vehicles (with Ackerman linkages) with emphasis on the methods used for explicit path following. Although the author's work is only applicable to ALVs, control methods used for AGVs and AHVs will be reviewed since many of these methods can be easily implemented on an ALV.

An impressive work in the area of explicit path tracking for ALVs was completed by Shin et al. [Shi91, Shi92]. Under the *FastNav* project, position based navigation was accomplished over featureless terrain at 11 m/s using a full-size modified van at Carnegie Mellon University (CMU) called NavLab. The four major subsytems were explicit path tracking, obstacle detection, obstacle avoidance, and the development of a real-time computing architecture [Sin91].

The problem of path tracking is simplified by kinematically decoupling the control of propulsion and steering. By choosing the midpoint of the rear axle as the vehicle control point, vehicle speed is independently controlled from the problem of planning steering motions. Vehicle speed is decided based on the current path curvature and the vehicle's proximity to perceived obstacles. An inertial navigation system (INS) was used to determine the vehicle's position and heading at a frequency of 20 Hz.

Since dead reckoning was not necessary, a kinematic model was not used. Nor was an explicit dynamical model used. Relatively simple experiments were performed that identified first order effects and a dynamical model was approximated. This model of vehicle dynamics is used by a feedforward module (preview controller) which is combined with feedback control. The feedback control module fits the current pose and a desired pose (placed at a constant lead distance ahead of the vehicle) with a quintic polynomial. The quintic polynomial is used to generate the current desired steering angle.

The most dominant characteristic behavior of the system's dynamics is latency. There is an inherent delay in when a command steering angle is issued and when it is achieved, particularly significant at high speeds. Feedforward control is used to anticipate an imminent line arc transition. The immediate goal of the feedforward controller is to have the steering angle reach 63% of the desired steering angle by the time a line arc transition is reached.

For high speed position based navigation, the delay in processing and executing commands is an important factor. Here, it is more important to know **now** where the vehicle will be **several seconds later** if a particular command is issued now. Some methods model the system behavior to predict the posture of the vehicle with respect to the path after some time increment if a certain steering angle is commanded [Oll91, Kel94, Mur94].

Ollero and Amidi [Oll91] utilized generalized predictive control (GPC) to develop the Generalized Predictive Path Tracking (GPPT) method, which is based on the receding horizon approach. This method uses a simple model of the vehicle's kinematics and a first-order dynamic model of the steering system. By assuming the vehicle's velocity remains constant, the vehicle's motion is represented by a locally linearized Controlled Auto-Regressive Integrated Moving Average (CARIMA) time discrete model.

The GTTP formulation is intended to minimize a cost index with the errors between future desired outputs and predicted outputs, and future increments of control. Results of experiments conducted on the NavLab vehicle were presented only for path tracking at a constant velocity of 3.4 m/s. The maximum range of the receding horizon within which outputs are predicted is said to be automatically tuned, although no details are given.

Ollero et al. [Oll94] conducted further experiments with the GPPT method (as well as the pure pursuit method) on a RAM-1 mobile robot testbed, built for research in indoor and outdoor industrial environments. The RAM-1 has two driven wheels and a top speed of 1.6 m/s. A fuzzy supervisory system was designed to auto-tune the control parameters of both methods using 81 fuzzy rules. The inputs are as follows:

1) the distance from the vehicle to the nearest point in the path being tracked,

2) the actual velocity of the vehicle,

3) the curvature at the current target point, and

4) the difference between the heading of the vehicle and the heading of the nearest point in the path.

The fuzzy rules are tedious to generate, requiring refinement through experimentation with the vehicle it controls.

Kelly [Kel94a, Kel94b] describes a feedforward simulator that models many aspects of a vehicle's kinetics that influences an adaptive pure pursuit algorithm during actual (not simulated) path tracking. The feedforward algorithm estimates where the vehicle will be after some time increment for several candidate steering commands. The candidate command which will place the vehicle closest to the target point becomes the vote of the strategic controller.

The solution of this problem is mathematically complex because it involves solving a coupled set of eight nonlinear differential equations that form the vehicle's state space model. At each step in the simulation loop, suspension, propulsion, steering, and position are estimated. This work was performed under the Real-time Autonomous Navigator with Geometric Engine (RANGER) project at Carnegie Mellon University, Pittsburgh, Pennsylvania.

Murphy [Mur94] has reported on the method of steering control used by the National Institute of Standards and Technology (NIST) High Mobility Multi-purpose Wheeled Vehicle (HMMWV). This research was performed by the Robotic System Division in support of the Department of Defense's (DOD) Robotic Testbed program and the Department of Transportation's (DOT) Intelligent Vehicle Highway Systems program.

Video cameras are mounted to the windshield of the vehicle and are used to track the painted stripes on a road that defines the edges of a lane. Steering control is accomplished using the pure pursuit method. The vehicle steers an amount proportional to the lateral position of the perceived lane. A subgoal is chosen a constant look-ahead distance in front of the vehicle. The steering angle chosen at each instant is the one that would cause the vehicle's control point to pass directly over the subgoal if this steering angle could be instantaneously applied.

The controller cycles at 15 Hz and the vehicle has accurately tracked the lane of a road at 55 mph. A model of the system has been developed that properly predicts that increases in speed causes a decrease in stability while decreases in controller delay causes an increase in stability. The paper examines the effect of vehicle speed, computational delay, and the look-ahead distance on system stability. This work uses delay compensation, effectively eliminating the pure delay associated with the vision processing stage. The transfer function developed to analyze stability is based on the assumptions that the tracking error is small and the vehicle tracks paths with small curvatures. No results, however, were provided with this work.

Three path tracking strategies were analyzed and compared by Amidi and Thorpe [Ami90] using the CMU developed NavLab robotic vehicle: pure pursuit, a quintic polynomial fit method, and a control theory approach. A set of perception modules were used to pass the motion controller module a desired path in terms of two-dimensional world coordinates. An inertial navigation system was used to estimate the vehicle's position and heading. The motion controller module has three components, a mapper, a tracker, and a velocity regulator.

The mapper keeps the list of points specifying the desired path as consistent as possible. It interpolates between given points (x,y) to estimate the heading of the midpoint.

The difference in heading from one midpoint to the next is used to compute the curvature at each path point. If navigator modules change the desired path, all the points in front of the vehicle are replaced by the most recent set of path points. The mapper can be configured to record a motion profile (x, y, z, ψ, ϕ, v) as the vehicle moves. The minimum turning radius of NavLab is 7 m and data is recorded every half meter.

The tracker chooses a point somewhere ahead of the vehicle on the desired path as the subgoal point. Then, it is uses the information at the subgoal to define what steering angle to command. The look-ahead distance the tracker module uses is 5-15 m and is a function of the vehicle's speed. Note that even at constant speed, the look-ahead distance is not constant. A particular subgoal may be considered several cycles through the tracker module before it is time to look at the next subgoal.

The control theory approach corrected for lateral and heading errors. The new curvature was calculated using the equation

$$\nu = g_1(\phi_{subgoal} - \phi_{vehicle}) - g_2(x_{subgoal} - x_{vehicle}), \tag{2.2}$$

where g_1 was determined experimentally and g_2 was chosen such that $g_1 > 2[g_2]^{1/2}$ (i.e., an over-damped system). Note that a positive lateral error $\Delta x=(x_{subgoal} - x_{vehicle})$ contributes a negative component to the steering angle and a positive heading error $\Delta\phi=(\phi_{subgoal} - \phi_{vehicle})$ contributes a positive component to the steering angle. Reasonable performance was reported at speeds less than 3 m/s. At speeds exceeding 3 m/s, attempts to overcome small heading errors, due to slop in the steering mechanism, led to small oscillations in path following.

The second approach tested continuously fit a quintic polynomial between the current vehicle location and the subgoal location. A new command curvature was continuously computed based on the curvature of the quintic at the vehicle's current location. This approach was based on a constant speed assumption.

The results using this method were not always good. Several issues were cited that limits this methods practicality. First, it was unclear how far to travel on a quintic before computing a new one. Secondly, for a quintic curve to respect a vehicle's minimum turning radius, longer look-ahead distances are required. Tracking performance, however, is very dependant on the look-ahead distance. A further issue that contradicts the goals of this method is that the entire length of a given polynomial is never completed, indicating that meeting the heading and goal constraints at the subgoal is not critical. Only simulation results are shown for this method.

The last approach tested by Amidi and Thorpe was a pure pursuit controller functioning at 10-15 Hz. The control equation used to determine the new curvature ν was

$$\nu = \frac{2}{L^2}x, \tag{2.3}$$

where L is the look-ahead distance and x is the lateral error in the vehicle's position. Notice that pure pursuit is simply a proportional controller using the x displacement as the error. The gain is easily tuned by changing the look-ahead distance L. Good results were reported up to 8 m/s. A look-ahead distance that was too long resulted in the vehicle cutting corners, whereas, a look-ahead distance that was too short resulted in the vehicle oscillating about the path.

Pure pursuit steering control, as applied to mobile robots, was borrowed from a maneuver commonly used by military pilots. The pure pursuit maneuver involves pointing the pursuer directly at the object being pursued. This is in contrast to the *lead* pursuit method, where the pursuer is pointed a fixed number of degrees ahead of the object being pursued. The pursuer will always converge on the target if the pursuer has a velocity greater than the target [Hom91].

Ollero and Heredia [Oll95] analyzed the stability of the pure pursuit algorithm for path tracking at constant speed (3, 6, and 9 m/s) for straight and constant curvature path sections, estimating the time lag for computing, communications, and actuator delays. For each velocity, the minimum look-ahead distance that made the motion stable and the maximum look-ahead distance that made the motion unstable were determined. The stability limit lies between these values. Experiments with a computer controlled HMMWV yielded stability limits that agreed with the analytical formulation which considers delay.

Krogh [Kro85] has described what he calls "*guaranteed steering control*" for AGVs using artificial potential fields. The conditions for attaining the goal state are outlined in terms of a Lyapunov-type attraction test function and a set of barrier functions. Krogh and Thorpe [Kro86] proposed a method for using the output of a path planner in conjunction with a generalized potential field (GPF) to accomplish path following. The task of moving to a single goal is effectively broken up into the task of sequentially moving to several critical points or subgoals. GPF's are used for local feedback control to guide the robot to each subgoal.

Feng and Krogh [Fen90] have presented an algorithm for reflexively steering a steered-wheeled vehicle to a goal point through an obstacle dense environment. Their two-stage algorithm involves first generating a visible subgoal by a subgoal selection algorithm (SSA), and then generating and implementing recommended steering and velocity commands with a feedback steering control algorithm. Local sensor data are used to update the visible subgoals while the vehicle is in motion. This algorithm was tested on the CMU NavLab vehicle. Experiments showed the algorithm was able to drive NavLab at 1.5 m/s through an obstacle course.

The concept of longitudinal control for AHVs envisions automatic controlled platoons of vehicles, each containing sensors, actuators, and communication equipment. The goal of longitudinal control is to decrease the required intra-vehicular spacing for a platoon of vehicles traveling at high speeds. These methods usually develop control laws for each vehicle in the platoon using the lead vehicle's velocity and acceleration, the preceding vehicle's velocity and acceleration, and the distance between the vehicle and the preceding vehicle [She91,Hed91].

The concept of lateral control for AHVs involves developing steering control laws capable of keeping an automated vehicle inside a perceived lane. Hessburg et. al. [Hes91] has investigated the performance and limitations of a PID/feedforward controller for a 50 kg computer controlled scale model (4WSD) car on a mock automated highway. The emphasis of the study, called the Program on Advanced Technology for the Highway (PATH), was determining the feasibility of a proposed discrete magnetic marker/sensing system. A secondary objective was the formulation of a control law that achieves "*good vehicle tracking to the center of the lane while maintaining smooth ride quality, in terms of lateral motion*".

A series of magnetic markers were placed in the center of a predefined path for the vehicle and magnetometers were mounted on the front center of the vehicle to measure the magnetic field from markers. The standard deviation of the reference positioning error was calculated to be 0.4 cm. The control law is reported to have performed well under nominal operating conditions but oscillations resulted when operating conditions deviated from normal values (such as simulated wind gusts).

Peng and Tomizuka [Pen91] have investigated a preview control law as applied to the lateral control problem. The addition of two feedforward terms to a feedback control algorithm based on the frequency-shaped linear quadratic (FSLQ) control theory, has been shown (in simulation) to greatly improve tracking performance and reduced the peak value of lateral acceleration.

Typically the control of AHVs are anticipatory rather than compensatory. A preview of the road geometry (curvature) may be encoded in the reference system (in the road surface) and then used in a feedforward term. Ackermann et. al. [Ack94] has presented a nonlinear control strategy based on the sliding mode theory which requires no preview of the road curvature. Two cascaded controllers are used to accomplish lateral control. The first uses the sensed lateral deviation from the guideline and the steering angle and the second uses the vehicle yaw rate. Velocity control is not addressed and only simulation results are provided. The control parameters were hand tuned.

Some path tracking methods consider the possibility of moving obstacles being present during path execution. Griswold [Gri90] has developed a control law where sensed objects are avoided by controlling the speed of the autonomous land vehicle along a predetermined path. The input to the speed controller is the vehicle's speed and a description of each sensed object's velocity. The output of the speed controller is an acceleration and the time of duration of the acceleration. The only change that occurs when obstacles are encountered is the speed along the predefined path.

Kehtarnavaz and Sohn [Keh91] have developed a steering control algorithm for a Dodge Caravan vehicle using neural networks. The system is applied to the vehicle tracking problem. A stereo vision sensor provides a vision cue to the controlling algorithm, which is used to detect the range and heading of the lead vehicle in real-time. This information is used to determine the appropriate steering and vehicle commands to track the lead vehicle. The network is trained by real data obtained from live vehicle-tracking test runs. Only simulation results are provided.

Chen [Che87] has described a human-like reflexive navigation algorithm for an ALV. A vision system provides images of the local environment from which a way point is selected at a constant distance ahead of the vehicle. The subgoal's direction and constrained open space are represented by a cone. The cone's vertex, spine, and orientation are parameterized as inputs to the control algorithm. The position and heading offsets are used to obtain a set of nonlinear differential equations that describes the vehicle's navigation performance. The navigation law is analogous to proportional navigation for air vehicles. Simulation results are provided for speeds up to 6 m/s, a lead distance of 24 m, and a curvature of 1/150 m^{-1}.

DeSantis [deS93] has developed an elegant solution to path tracking for car-like robots. It is valid when the path is a set of lines and circular arcs, and if one can make the assumptions that 1) the motion is planar and exempt from side slippage, 2) the lateral, heading, and velocity-tracking offsets (errors) are kept small, and 3) the assigned tracking velocity is constant. Under these assumptions and conditions, path tracking is shown to be equivalent to the stabilization of a linear system. Simulated experiments are alluded to but no results are provided.

Muller [Mul92] has outlined a procedure for superimposing feedback control and a rule-based feedforward control law to vision based road following that involves turning from one road to another. This work is built on a four-dimensional approach to modeling the world thru machine vision. No results are provided with this work.

Samson and Ait-Abderahin [Sam91] also proposed nonlinear control laws for the stable tracking of a trajectory for a nonholonomic cart with two driven rear wheels and a fully rotatable front wheel. d'Andrea-Novel [dAn91, dAn92] described dynamic feedback linearization control laws for stable path tracking for three-wheeled robots (with two front wheels and one steerable rear wheel) and omnidirectional vehicles.

Canudas de Wit and Sordalen [Can91] proposed a piecewise continuous pure feedback law for a nonholonomic mobile robot with two DOF that exponentially stabilizes the cart about the origin. They later expanded this work to obtain stabilization about an arbitrary point in space [Sor92]. This was used to track a sequence of via-points along a path composed of line segments and circular arcs. A velocity vector and error vector is specified for each via-point along the path. Desired velocities could be specified as either positive or negative.

Similarly, Kanayama et al. [Kan88, Kan90] proposed a method that takes a sequence of reference postures that define the planned path and passes them to the vehicle's control system, one at a time at a constant rate. The posture control loop determines the error in the vehicle's position and orientation with respect to the current reference posture. Then the error components are PID filtered to produce new command velocities. A new control rule is proposed and proven stable using a Liapunov function.

The concept behind the method used by Nelson and Cox [Nel88] is virtually identical to the early Kanayama approach. A local path planner passes a path segment to a reference path generator which determines the appropriate reference state for the robot to track. Path segments can be lines, arcs, or splines. The reference state to be tracked is updated at each control cycle. New command variables are a function of the previous command variables plus the weighted errors in position and heading.

Rathbone et al. [Rat87] argued that the specified path to be followed should include rate of change of direction information. Path following around a curve did not significantly increase their computational cost but offered better predictability of the actual vehicle's path. van Turennout et al. [van92] have studied feedback control for tracking a wall at constant speed. The distance to the wall, obtained using ultrasonic sensor data, is maintained during vehicle motion.

Open loop strategies for path tracking have been considered by Murray and Sastry [Mur90], and Lafferiere and Sussman [Laf91].

CHAPTER 3
SPECIFYING A PATH

Many robotic applications benefit from the systematic execution of an efficient preplanned path. In autonomously clearing a beach of mines, or surveying a field for buried munitions, achieving 100% coverage of the area is critical. Here, it is a imperative that the vehicle accurately follow the specified route. Such navigation systems are position based. The positioning solution is required to be extremely accurate and available at a high rate.

This chapter discusses the various ways in which a path can be specified. The path execution software is flexible in that it is able to receive a variety of path types, each to be explicitly followed.

Area Map Manager

With position-based navigation systems, information about the environment is usually available to the onboard computer. This information is used to generate safe and efficient routes for the vehicle to travel. The NTV's onboard computer stores two-dimensional information about the operating environment in data files on the hard disk.

As shown in Table 1, an *environment* file contains a work space perimeter polygon, circular and polygon shaped obstacles, and center points along roads and preferred paths. A *survey* file contains the perimeter of regions to be surveyed. A *path* file contains prerecorded paths, where the path points are separated by one meter, or are specified by a series of line segments and circular arcs that are to be concatenated together.

Table 3.1. Area map data management.

File Type	Data	Scope of Memory
environment	Work space perimeter	global (shared)
	Obstacles (circular or polygon)	
	Preferred path points	
survey	Perimeter of regions	local
path	Prerecorded paths	
	Line/arc paths	

Map information is recorded in these data files in geodetic coordinates (latitude, longitude). The standard sign convention is followed, where negative longitudes are in the western hemisphere and negative latitudes are in the southern hemisphere. It is convenient to use these absolute coordinates since everybody can agree on where they are. It is not convenient, however, to plan paths in a polar coordinate system. The area map manager uses standard formulas (implemented by the author and Mr. David Novick) for converting geodetic coordinates into a Cartesian coordinate system called the Universal Transverse Mercator (UTM) grid [Dep58]. The parameters associated with the 1984 World Geodetic System (WGS84) are used.

The area map manager can be sent messages to read these data files. The *environment* data are placed directly into shared memory where other programs, such as path planners, have direct access to them. The other data are stored locally in dynamically allocated memory. When this information is required by other programs, a request for it is sent to the area map manager.

Path Types

Paths can be computer generated (planned), user generated, or prerecorded by manually driving or remotely operating the vehicle. Table 2 illustrates the many ways in which a path can be generated. The author has implemented a suite of path planning algorithms, each of which returns a set of via points. These points define a safe path when connected together with line segments.

Table 3.2. Types of paths that can be specified.

Types of Paths	Ways to Generate Paths
Computer generated	Preferred Paths Planner
	Obstacle-Avoiding Planner
	Site Survey Planner
	Goal-Ordering Planner
User generated	Select via points from GUI
	Define a sequence of lines and arcs
Prerecorded	Record a path during manual operation
	Record a path during remote operation

The user can specify a path by transmitting a series of via points to the vehicle from a computer at a command center. A graphical user interface (GUI) designed at Wright Laboratory allows the user to generate a path by selecting points with a mouse that are outside of obstacles but within the work space.

A user can also design a path by specifying a sequence of line segments and circular arcs that are to be concatenated together. Line/arc paths are defined with respect to the vehicle's current position and heading. Line/arc paths are specified in a *path* data file and are stored locally within the area map manager. They can be requested from a user interface program when needed.

Prerecorded path points can be recorded while manually driving the vehicle or during remote operation of the vehicle. Path points are stored at one meter intervals.

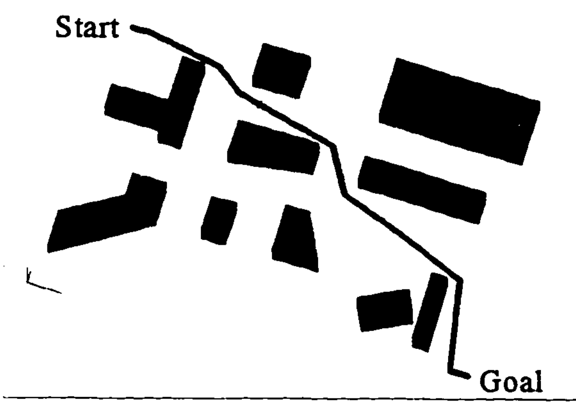

Figure 3.1. Shortest collision-free path between a start pose and goal pose.

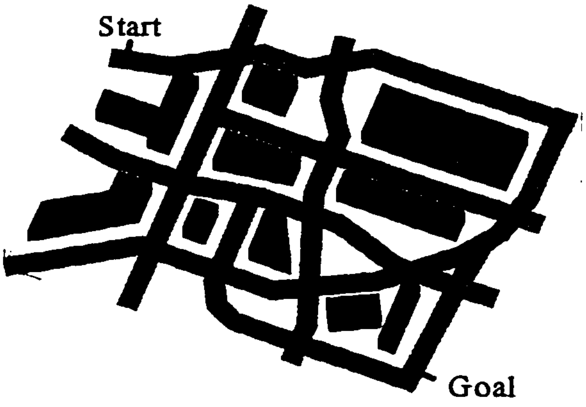

Figure 3.2. Shortest path between two points along a network of preferred paths.

Path Planning

Several planar path planning algorithms have been implemented on the NTV in an effort to enable it to make intelligent decisions on how to get to where it needs to go. An A* search algorithm, which is used to generate the shortest collision-free path between two vehicle poses, was implemented by the author as a part of his Master's research (Figure 3.1).

Subsequent to this work, the search algorithm was modified to generate the shortest path along a set of preferred paths (Figure 3.2). The preferred paths can be roads, sidewalks, hallways, aisles, trails, etc. Notice in Figure 3.2 that the start and goal points can be specified off of the preferred paths. The algorithm considers off-path travel valid but twice as costly as on-path travel [Ran96c].

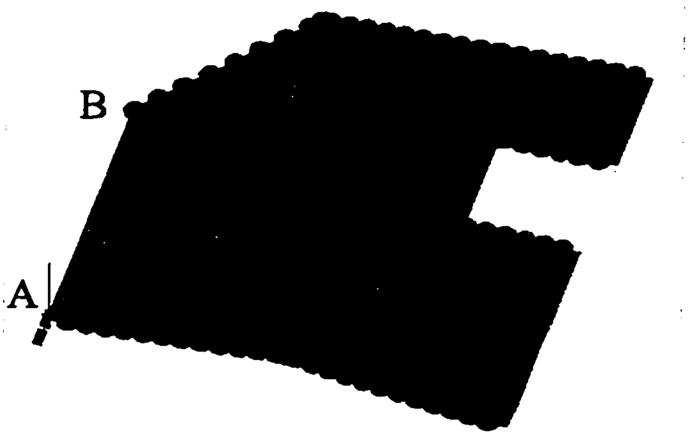

Figure 3.3. A site survey plan for achieving 100% coverage of the site.

A third planner was then developed called a Site Survey Planner (SSP). The goal of a SSP is to specify an efficient path in a bounded region that, if followed accurately, will cause the vehicle to traverse 100% of the site (Figure 3.3). The work space is divided into a set of parallel rows. The total path is then organized such that each row is traversed only once (except when there are an odd number of rows, in which case the middle row is traversed twice). The algorithm is smart enough to plan alternate routes around obstacles that lie inside the site to be surveyed. The shortest alternate route is specified, unless it would require a turning radius less than the vehicle's minimum turning radius. In such cases, the planner incorporates the longer alternate route into the path [Ran96a].

Finally, a high-level planner was implemented that sequentially calls the obstacle-avoiding planner, and determines the order in which a set of goal poses should be visited so that the total path length is minimized. This planner concatenates the recommended goal-to-goal paths and outputs the optimized total path (Figure 3.4). This planner has been tested in simulation but not on the NTV.

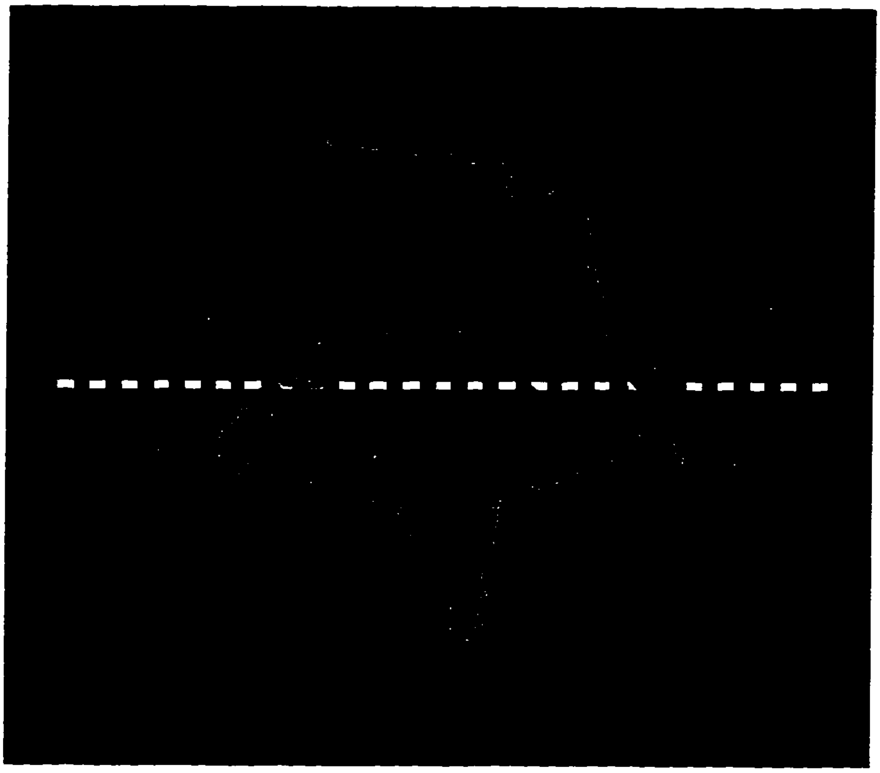

Figure 3.4. A goal ordering algorithm minimizes the total path length.

Each of these planners outputs a set of via points that defines an optimal path when connected by line segments.

Standardizing and Smoothing a Path

There are several types of paths that the vehicle can be requested to execute: a path planned by one of the four off-line planners, a path that was previously recorded during manual operation, a path synthesized by a user from a GUI at a command center, or a path specified by the concatenation of a set of circular arcs and line segments. The suite of planners returns a set of via points that define a piecewise linear path when connected with line segments.

Flexibility in path specification is desirable from the user friendly standpoint. It does present a control problem, however. Flexibility would appear to necessitate the rewriting of the control algorithm for each path representation. This would be tedious, though, since every new control algorithm tested on the NTV would require several forms. A better way to solve this problem is to convert each allowable path representation into a standard path representation. This is accomplished by a program called the *subgoal generator.*

As shown in Figure 3.5, the *subgoal generator's* primary role is to cast any of the

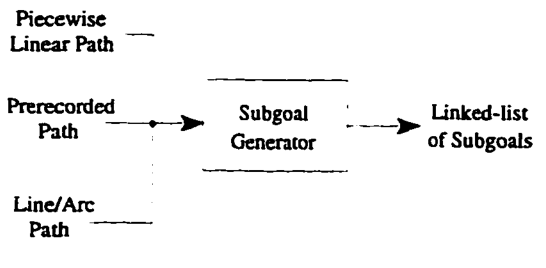

Figure 3.5. The *subgoal generator* program converts paths to a linked-list of subgoals.

acceptable path representations into a format that the control algorithm expects; a linked-list of subgoals where the top node is the first path point and the bottom node is the goal point. The controlling program is then passed a pointer to the top of this list. At any given time during path execution, a new list of path points can be sent to the controlling program by the *subgoal generator* and the vehicle will instantly begin to track the new path.

A secondary role of the *subgoal generator* is to smooth out the rough edges in a path. Since a user is allowed to generate a path, it is not assumed that all paths can be autonomously tracked (without using both the forward and reverse gears). A user may not have a good feel for how sharp a turn the vehicle can execute. Figure 3.6 illustrates how the *subgoal generator* filters a user defined path.

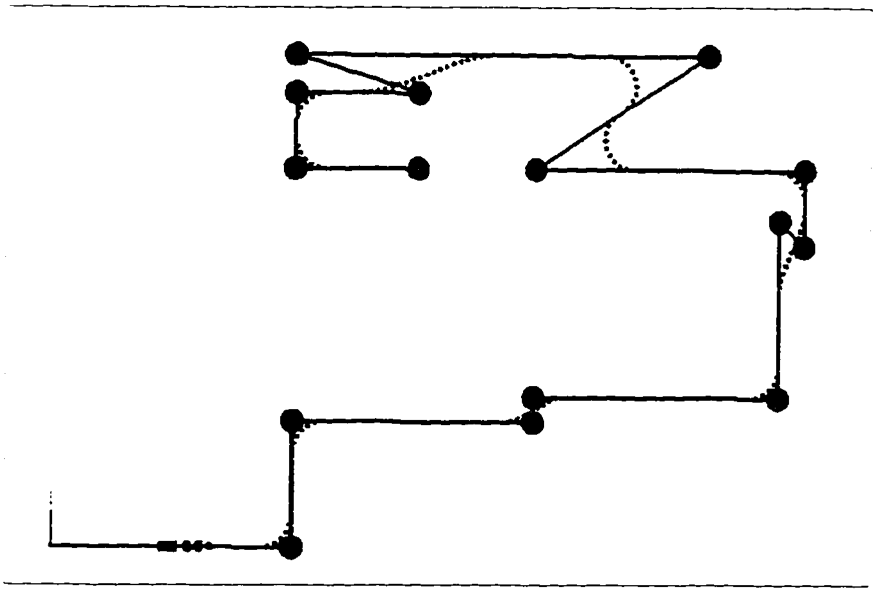

Figure 3.6. A path filtered by the *subgoal generator* program.

The largest dots represent a sequence of points specified by a user. Path segments are fitted with points at 1 meter intervals that lie on a circular arc having a radius equivalent to the vehicle's minimum turning radius. Segments that cannot be fit with circular arcs are fit with cubic polynomials. Cubic polynomials are used to bypass path segments that cannot be tracked. Figure 3.7 illustrates path tracking along a path around an obstacle that has been smoothed by the *subgoal generator.*

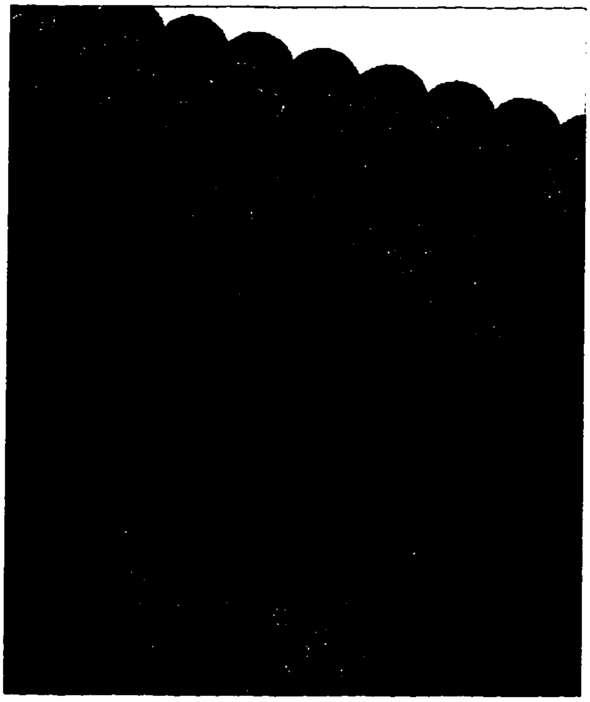

Figure 3.7. Traveling along a cubic curve around the side of an obstacle.

CHAPTER 4
CONTROL STRATEGY

This chapter describes the strategy developed by the author to autonomously control a steered-wheeled all-terrain land vehicle. The strategy is geometry-based in that the command variables are specified based on the geometric characteristics of the vehicle. Most of the techniques used to control automatic guided vehicles (AGV's) in factory settings are not applicable to autonomous land vehicles (ALV's), since the properties of the surface on which ALV's are expected to operate do not remain constant. An accurate mathematical model of the system requires some knowledge of the terrain/wheel interaction, and this changes from one type of terrain to another.

As with geometry based control systems, the vehicle control point is chosen to be the center of the rear axle. This choice allows for the decoupling of the steering and propulsion control. Velocity control is under study by a graduate student who is implementing a fuzzy logic controller on the functions of throttle and braking. For the author's research, velocity control is treated like the cruise controller on automobiles. A command velocity is specified prior to the commencement of path tracking. The author's task is to design and tune steering controllers that accomplishes accurate path following over a range of speeds (0-4.5 m/s).

The strategy the author has developed is based on a *carrot following* approach. A subgoal is generated in real-time at a specific look-ahead distance in front of the vehicle's current position and on the preplanned path, or on the circular or cubic arc that connects the current path segment with the next path segment. At any given instant, the vehicle is directed to move towards the current subgoal. Two steering control methods are to be evaluated: PID control where the feedback error is the vehicle's heading error, and the pure pursuit method.

Software Control Architecture

Figure 4.1 illustrates the flow of information between computer programs during autonomous navigation. Each box, with the exception of the positioning solution box, can be considered a program that handles a specific task. The positioning solution box is a suite of programs that interface positioning sensors. The data from an inertial navigation system (INS) and a differential global positioning system (DGPS) are integrated with an external Kalman filter. The output is the system's best estimate of the vehicle's current attitude.

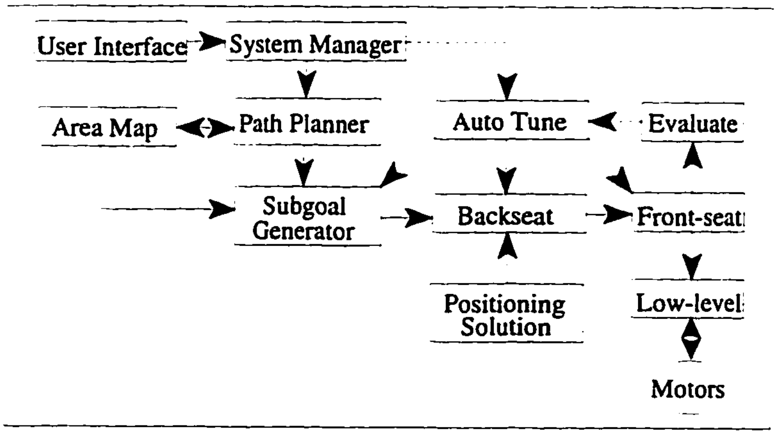

Figure 4.1. The flow of information between the autonomous navigation programs.

As an example, the user specifies the vehicle's start and goal configurations and then sends a command to the *system manager* (from the *user interface*) to move the vehicle to the goal configuration. The *system manager* instructs the path planner to generate a safe and efficient path that guarantees to avoid collisions with all known obstacles in the *area map* database. The planned path is passed to the *subgoal generator,* which filters the path and loads a linked-list of subgoals. A linked-list of subgoals is the standard form of an executable path.

The path is passed to the *backseat* program and vehicle motion begins. The *backseat* program generates in real-time carrot positions on the path that are a specific look-ahead distance in front of the vehicle. The *backseat* program also estimates the curvature of the path in the neighborhood of the real-time carrot and calculates a desired vehicle velocity. The *front-seat* program looks at the current carrot position and uses PID or pure pursuit control to specify a command steering angle. In the current implementation, PID control is used to specify command throttle, brake, and transmission positions. An alternate approach for steering control that has been evaluated is a weighted PID/pure-pursuit solution.

The *low-level* program evaluates the steering, throttle, brake, and transmission motor's feedback values and moves the motors towards the command values. If a new command value is issued before the old command value is reached, the old command value if forgotten and the new command value is tracked.

Table 4.1 shows the program distribution on four Force CPU boards (called Bart, Homer, Lisa, and Marge) in the NTV's VME computer. Bart is used to run programs that estimate the vehicle's current attitude (latitude, longitude, elevation, roll, pitch, yaw, and velocity). Homer runs programs involved in obstacle avoidance and site characterization. Lisa runs programs that are required to plan and execute paths. Magie runs miscellaneous nonreal-time programs (such as a user interface) and a data logger.

Table 4.1. Navigation software distribution on four NTV computer board.

Force 33 (Marge)	Force 30 (Bart)	Force 33 (Lisa)	Force 30 (Homer)
User interface	INS interface	System manager	Sonar interface
Data logger	GPS interface	Area map manager	Vision communication
	Kalman filter	Path planners	Sensor fusion / Obstacle avoidance
		Subgoal generator	
		Backseat driver	
		Front-seat driver	
		Low-level controller	
		Radio-modem communications	
		Performance evaluator	
		Steering calibration / Auto-tuning	

Calculating a Target Position in Real-Time

The *backseat* program is responsible for recommending a general direction in which the vehicle ought to move. The backseat program receives a pointer to a list of subgoals from the *subgoal generator* program. This list of subgoals represents the executable path. The first task the *backseat* program performs is it makes a local copy of the list of subgoals. If the path is commanded to be executed once, the copied path is sequentially removed from memory as the vehicle finishes traversing path segments. When no path segments are left, or the vehicle is within a predefined tolerance of the goal position, the *backseat* program reports to the *system manager* that the goal pose has been reached.

The global copy of the path continues to exist in shared memory until the *subgoal generator* is instructed to generate a new executable path. The *subgoal generator* can generate a new executable path without affecting the local copy currently being tracked. With this structure, *backseat's* work is shielded from other programs. Path planning, subgoal generation, path execution, and obstacle avoidance are all tasks that can occur concurrently without the vehicle being required to stop.

Path tracking is accomplished by generating a target position on the path and ahead of the vehicle by a look-ahead or lead distance. There are two ways in which to apply the look-ahead distance. In the first method, the target position is located by finding the first normal to the path that intersects the vehicle's current position. In Figure 4.2, the intersection of the normal and the path is the lateral path point *P*. The target point *T* is located on the path a distance L_a (which is a function of the vehicle speed) from the lateral path point.

In the second method, the target point *T* is determined by finding the intersection (ahead of the vehicle) between the circle centered at the vehicle's current position *V* with

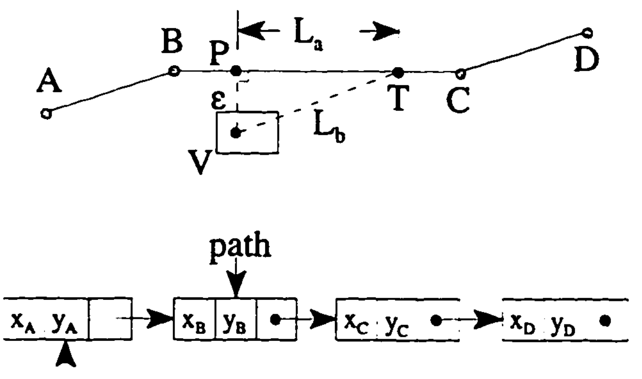

Figure 4.2. Calculating the real-time carrot from a path linked-list.

radius L_b, and the path. The look-ahead distance L_b is normally used with the standard pure pursuit method.

The current lateral path point *P* and target point *T* are placed in shared memory by the *backseat* program at 10 Hz. The distance between the lateral path point and the vehicle's current position *V* is the current tracking (or lateral) error ε. The tracking error ε is also calculated by the *backseat* program and placed in shared memory. The error is negative if the vehicle is to the right of the path and positive if the vehicle is to the left of the path.

In Figure 4.2, the executable path is a linked list of the (x, y) points *A*, *B*, *C*, and *D*. The lateral point *P* is currently between path points *B* and *C*. The pointer *path* always points to the path point just behind the lateral point. The subroutine that obtains the real-time carrot can operate in two modes. In the first mode, the specified path is continuously repeated. This is particularly useful in auto-tuning. In the second mode, the path is executed until the goal point is reached, at which time the subroutine returns a status indicating so.

If the vehicle is commanded to continuously repeat the path, the linked list is made cyclic by having the last node point to the first node. If the vehicle is commanded to execute the path once, then the points prior to the one pointed to by *path* are removed from the linked list in real-time. If the vehicle's current position is behind to the first leg of the path, then the extended first path segment is tracked. If the vehicle is close to the goal point and the target point moves off-path, then the target is placed on the last path segment extended.

The subroutine that finds target positions also estimates the path curvature in the neighborhood of the vehicle. This curvature estimate can be used to modify the command velocity and to rotate the target off-path a slight amount so that the path tracking of a trailer is improved around path corners.

Evaluating Path Tracking Performance

A survey of the literature indicates a lack of tools that can be used to evaluate the accuracy with which autonomous navigation is currently achieved. All too often, a paper includes a plot of a test run in its results section with one line labeled planned path and another line labeled actual vehicle path, and the reader is left to subjectively decide how accurate navigation was accomplished, and where the work fits in with other methods that achieve accurate navigation.

Accurate navigation along an explicit path depends not only on the accuracy of real-time positioning but the accuracy of vehicle control. Positioning is accomplished on the NTV by the integration of INS and DGPS data with an external Kalman filtering algorithm. An Ashtech Z-12 GPS receiver internally logs time-stamped position data at 1 Hz. This data is post-processed by software provided by Ashtech Inc. to determine the actual path of the vehicle to within a few centimeters.

The total path following error is the sum of the positioning error and the control error. Positioning error is determined post mission by taking the difference between the time-stamped post-processed GPS position data and the time-stamped real-time (Kalman filtered) positioning solution. The control error can be determined in real-time by comparing the real-time positioning solution to the path that is being tracked.

The control error can be thought of as the error in position between where the vehicle ought to be and where the vehicle thinks it currently is. If the real-time positioning data are not smooth, control error can be induced. This work focuses on measuring and reducing control error during path tracking. Other graduate students are working to improve the accuracy of real-time positioning.

The real-time performance evaluator calculates the vehicle's average, maximum, and standard deviation from the path at 10 Hz. These parameters are reset each time the subgoal generator passes a pointer (to a new executable path) to the *backseat* program. The average and standard deviation values for an executed path can be skewed, however, if the desired velocity is not accurately maintained. As a result, these measures are not ideal for making critical decisions concerning the modification of control parameters.

The performance index used by the author in making such decisions is the area between the actual path of the vehicle (as indicated by the real-time positioning solution) and the executable path, normalized by the path length. The area of the quadrilateral V_i - V_{i-1} - P_{i-1} - P_i is calculated each cycle through the performance evaluator using equation 4.1, where *a*, *b*, *c*, and *d* are the lengths of the sides, *p* is the distance between the *a-b* and *c-d* corner points, and *q* is the distance between the *a-d* and *b-c* corners points.

$$Area = \frac{\sqrt{4p^2q^2 - (b^2 + d^2 - a^2 - c^2)^2}}{4} \tag{4.1}$$

Other indices of performance includes an oscillation index and a smoothness index. A buffer zone of 0.2 meters is centered about the path. The oscillation index is defined as the number of times the vehicle intersects both sides of the buffer zone per 100 meters of the path. A buffer zone of 4° is centered around the home steering position (0°). The smoothness index is defined as the number of times the steering wheel intersects both sides of this buffer zone per 100 meters of the path.

PID Steering Control

One method of steering control investigated by the author is proportional-integral-derivative (PID) control, as applied to the error in the vehicle's heading. As shown in Figure 4.3, that there are two levels of steering control. High-level control encompasses all the steps taken in obtaining a desired steering position. Low-level control encompasses all the steps taken to move the steering wheel to, or towards, the desired steering position.

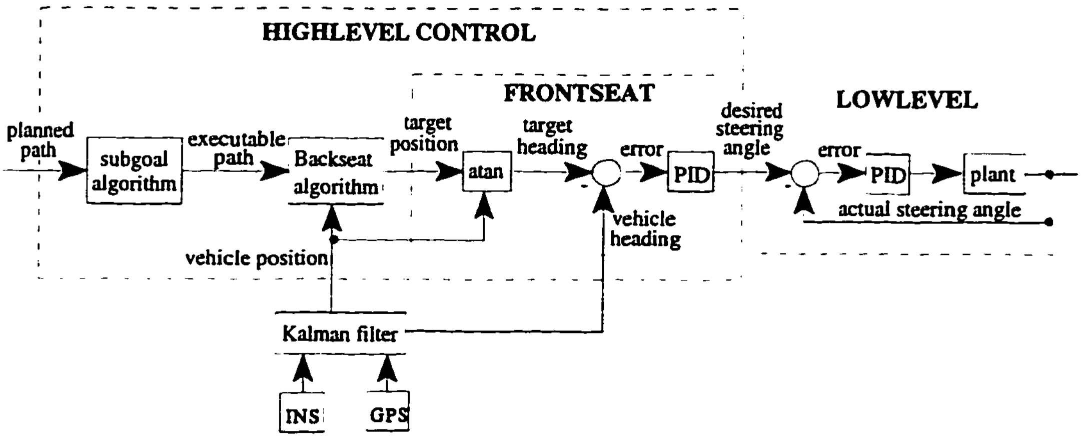

Figure 4.3. Steering control strategy using a high-level PID algorithm.

A low-level program issues steering commands to a servo module that drives a motor attached to the steering column. High-level control occurs at the software level, whereas low-level control occurs primarily at the hardware level. The author's work focuses on high-level control. The NTV's low-level control is discussed in the *Modeling the NTV's Steering Behavior* section of chapter 5.

As shown in Figure 4.3, the input to the high level steering controller is a general path. The general path is smoothed and cast into a standardized form by the *subgoal generator*. The output of the *subgoal* algorithm is an executable path. The executable path is compared with the vehicle's current position at 10 Hz by the *backseat* algorithm, and a target position is produced. The *front-seat* algorithm calculates the vehicle's current error e_n and a PID controller generates a desired steering angle ϕ_n.

The idealized equation of a PID controller is shown in Equation 4.2. A PID

$$\phi(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right] \tag{4.2}$$

controller contains a term that is proportional to the error, one that is proportional to the integral of the error, and one that is proportional to the derivative of the error. The integral term acts as a spring (in a spring-mass-damper system) in that it eliminates steady state error. The derivative term acts as a damper. The parameters characteristic to the system are the proportional gain *K*, the integral time T_i, the derivative time T_d, and the sampling time *T*. For small sample times *T*, the idealized equation can be written as a nonrecursive difference equation, as shown in Equation 4.3.

$$\phi_n = K\left[e_n + \frac{T}{T_i}\sum_{j=0}^{n-1} e_j + \frac{T_d}{T}(e_n - e_{n-1})\right] \tag{4.3}$$

In this form, rectangular integration was applied.

The heading error e_n is calculated by finding the orientation (in radians) of the vector from the vehicle's control point to the target position with respect to the axis in the direction of the vehicle's current heading (see Figure 4.4). The error is bounded by the equation

$$-\pi \le e_n \le \pi. \tag{4.4}$$

A positive error results in a left turn and a negative error results in a right turn. An error greater than π/4 automatically results in the maximum steering position.

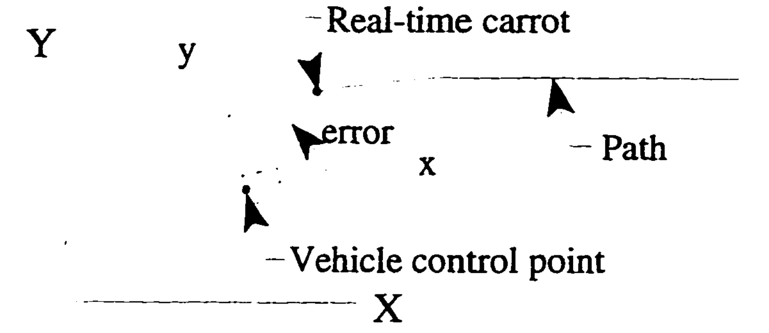

Figure 4.4. PID steering control is based on the current heading error.

Equation 4.2 can be rewritten as the difference equation

$$\Delta\phi_n = K_0 e_n + K_1 e_{n-1} + K_2 e_{n-2}, \tag{4.5}$$

where,

$$\begin{aligned} K_0 &= K(T_d/T) \\ K_1 &= K(-1 + T/T_d - 2T_d/T) \\ K_2 &= K(T_d/T) \end{aligned} \tag{4.6}$$

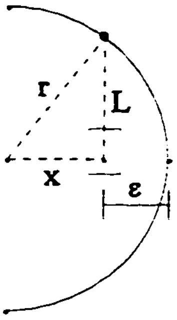

Figure 4.5. Inherent PID error ε around curves.

PID control works to force the heading error to zero so that the vehicle is always pointed towards the current carrot position. The advantages and disadvantages of using PID control are discussed in the *Weighted PID/Pure Pursuit Steering Control* section of this chapter. Here, it is noted that an inherent lateral error ε to the path exists when tracking curves, even when tracking is perfect, as shown in Figure 4.5. A method for determining the characteristic parameters of a PID controller is discussed on the *Auto-Tuning Control Parameters* section of this chapter.

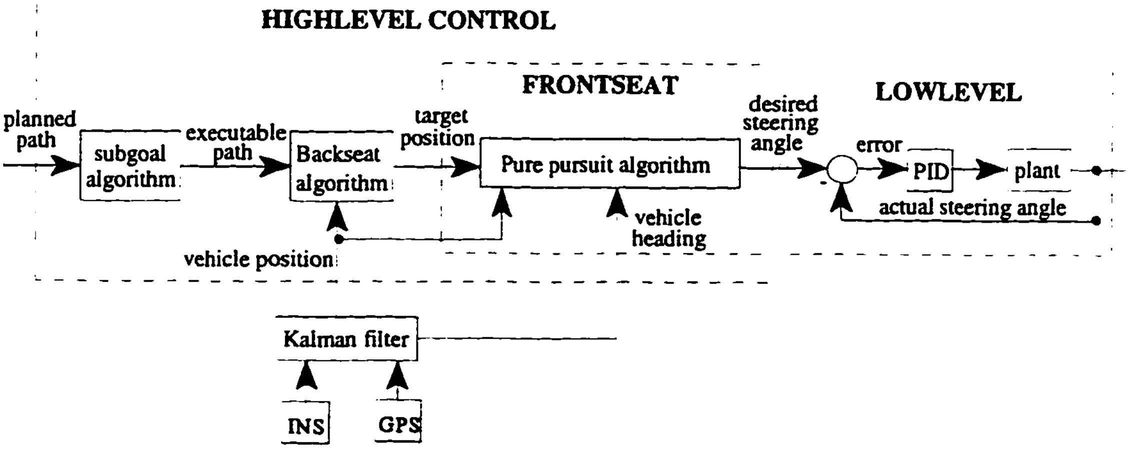

Figure 4.6. Steering control strategy using a high-level pure pursuit algorithm.

Pure Pursuit Steering Control

A second method of steering control has been evaluated by the author. Pure pursuit is probably the most popular method of high-level steering control used on ALV's. Figure 4.6 illustrates the steering control strategy when pure pursuit is used. This strategy and the one described in Figure 4.3 (which uses PID control) differ only in the *front-seat* algorithm.

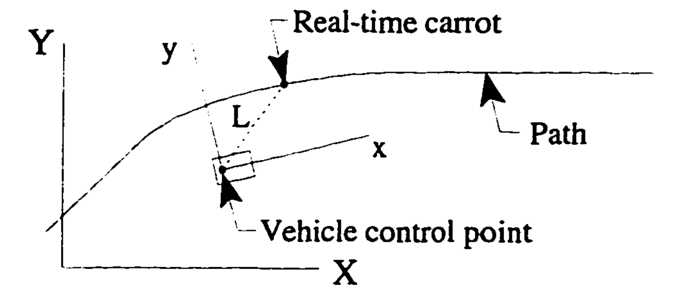

Figure 4.7. Look-ahead L to the target for standard pure pursuit steering control.

The real-time carrot to be pursued by the vehicle under standard pure pursuit control is a point on the path a specific look-ahead distance *L* from the vehicle's current position, as shown in Figure 4.7. If the vehicle if farther than *L* from the path, then the real-time carrot coincides with the lateral path point *P*. Adaptive pure pursuit, a variant of the standard form, makes the look-ahead distance *L* a function of the lateral path error ε.

The pure pursuit method generates a command steering angle ϕ that would drive the vehicle through the current real-time carrot if the steering angle could be changed instantaneously, as shown in Figure 4.8. The controlling equation for pure pursuit is

$$\phi = atan\left[\frac{Wy_{ref}}{L^2}\right], \tag{4.7}$$

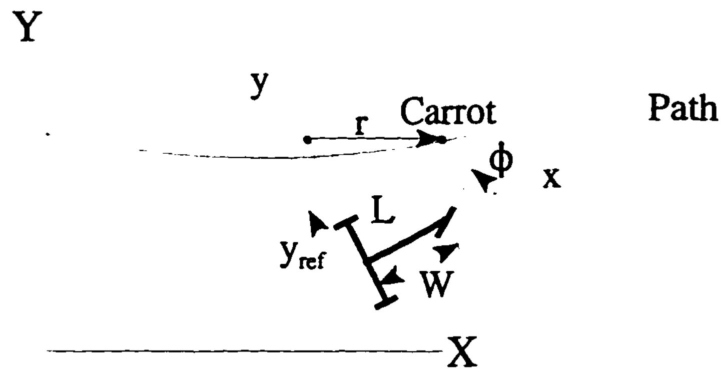

Figure 4.8. Pure Pursuit determines the command steering angle, ϕ, that will drive the vehicle's control point through the carrot.

where, *W* is the distance between the axles on the vehicle and y_{ref} is the position error along the y-axis of the vehicle coordinate system.

Weighted PID/Pure Pursuit Steering Control

Both the PID and pure pursuit methods of high-level steering control have advantages and disadvantages. The PID method is stable when the vehicle's velocity is small and the distance to the carrot is large. The disadvantages of the PID method is that 1) it is difficult to accurately tune its control parameters, 2) it is unstable at high velocity, and 3) it causes corners to be cut due to an inherent error.

The pure pursuit controller is easy to tune and performs well over the tested velocity range (0-4.5 m/s) when the vehicle is started on the path. If the lateral error is large, however, this method becomes unstable. Stability can be improved by using an adaptive rather than standard pure pursuit controller. With the adaptive version, the look-ahead distance is a function of the lateral path error. Even with the adaptive controller, however, stability can still be a problem when converging to a path.

The NTV is capable of detecting and avoiding obstacles that block the planned path. After avoiding an obstacle, the NTV is expected to return to the planned path and continue tracking it. Thus, it is important that path tracking be stable, even when there is a large lateral path error.

The author has attempted to combine the desirable features of both controllers into a single controller. In general, the pure pursuit method performs well when the vehicle is near the path and the PID method performs well when the vehicle is approaching the path from a distance. The weighted solution will use 100% of the PID recommendation when far from the path, 100% of the pure pursuit recommendation when near the path, and a combination of the two recommendations when in between those conditions.

Autonomous Steering Calibration

When a certain steering position is commanded and a circle is traced out with the vehicle, the diameter of the circle is not what would be expected from the geometry. There are several reasons for this:

1) The steering encoder on the NTV measures how much the steering column has rotated. The meaning of this steering angle can be thought of in this way: If the vehicle had one front wheel, as tricycles do, this would be the angle that the wheel makes with the body. But the vehicle does not have just a single front wheel.

2) It is tedious to precisely find the steering home position (0°). The procedure requires "homing" the steering, driving forward, and checking to see if the vehicle's heading has changed. Error in defining the home position will be reflected in moving to all other positions.

3) Under ideal conditions (no wheel slippage), the relationship between the steering angle ϕ and the curvature γ of the path traced out by the vehicle is $\gamma=\tan(\phi)/W$, where, W is the distance between the front and rear axle. Ideal conditions usually don't exist, however. The curvature of the path traced out by the vehicle is a function of the steering angle, surface conditions, vehicle speed, tire stiffness, etc.

The output of the high-level controller is a desired steering position. The purpose of performing the following calibration procedure is to collect information useful in predicting the steering position that should be executed, in order that the curvature of the path traced out by the vehicle corresponds to the specified desired steering position.

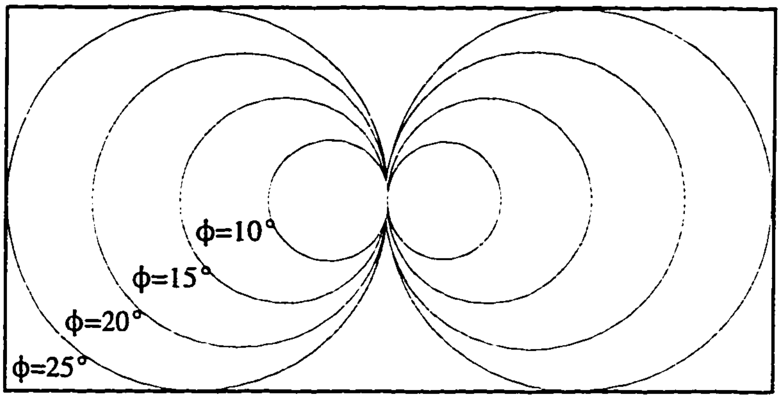

Figure 4.9. Eight circles autonomously executed to calibrate the steering.

Figure 4.9 shows the steering angles used to trace out eight circles. At the press of a button, each circle is repeated three times without stopping at a speed of 1.34 m/s. The diameter of each circle is measured six times in software and recorded. (The diameter is calculated when the vehicle's heading has changed by π). The first measurement for each circle is discarded since the vehicle had to accelerate from a stop during the first part of that lap. After each circle has been repeated three times, the vehicle stops for 30 seconds to allow the INS to correct for drift.

After the procedure is completed, the data are fit with a line using the method of least squares. The slope and y-intercept are recorded in a calibration file. The area of clear space needed to complete this procedure with the NTV is approximately 17.3 m x 34.6 m.

Auto-Tuning Control Parameters

As pointed out in the *Motivation* section of chapter 1, manually tuning control parameters is usually a straight-forward but tedious process. Ball-park values have been established which can be used as starting points for fine tuning. Fine tuning may need to be repeated, however, if the terrain or system dynamics of the vehicle have significantly changed since the last time the procedure was completed. These facts makes this application ideal for auto-tuning.

The gains that need to be tuned for the PID method includes a proportional gain *K*, an integral time T_i, a derivative time T_d, and the look-ahead distance *L*. The only gain that needs to be tuned for the pure pursuit method is the look-ahead distance *L*. Adaptive pure pursuit is used in this study, where the adaptive look-ahead distance is a function of the tuned look-ahead and the lateral path error. The PID gains are tuned using the method specified in the next section. The look-ahead distance for both the PID and pure pursuit methods are tuned using the golden section search (GSS) optimization technique.

PID Relay Auto-Tuning

The majority of controllers used in the industry are the PID type. The fundamental goal of a self-tuning PID controller is conceptually simple: observe the behavior of the process and modify the controller's gains until the closed-loop system performs as desired. In practice, however, implementing a self-tuning controller that performs at least as well as a manually tuned controller can be a complex problem [Van94].

There are two major ways to tune a PID controller: the step response method and the frequency response method. In the step response method, the process open-loop response to a step input is analyzed graphically to determine the *dead time* and *time constant*. In the frequency response method, the proportional gain is increased (with the other gains set to zero) under closed-loop control until oscillations are sustained.

The proportional gain that produced sustained oscillations is the *ultimate gain* K_u. The *ultimate period* T_u can be measured from a graph of the response. The *ultimate gain* and *period* describe the point of intersection of the Nyquist curve with the negative real-axis for an open-loop system.

The Ziegler-Nichols tuning rules are one of the most popular methods for specifying the PID gains, once the *ultimate gain* and *period* for a system are determined. The primary design criteria was to reduce the effects of load disturbances. A decay ratio of 1/4 was specified as a design criteria. The Ziegler-Nichols tuning rules for a PID controller are

$$\begin{aligned} K &= 0.6K_u \\ T_i &= 0.5T_u \\ T_d &= 0.125T_u \end{aligned} \tag{4.8}$$

It is difficult, however, to automate the procedure where the proportional gain is increased until sustained oscillations are kept under control. A simple relay auto-tuner described in [Ast95] has been implemented by the author to determine the system's characteristic parameters. The method is based on the observation that a system with a phase lag of at least π at high frequencies may oscillate with period T_u under relay control.

The vehicle is placed into either autonomous PID control mode or relay tuning mode by the press of a button. This is represented by the switch in Figure 4.10. In autonomous PID mode, the vehicle tracks the given path. In relay mode, the vehicle is oscillated about

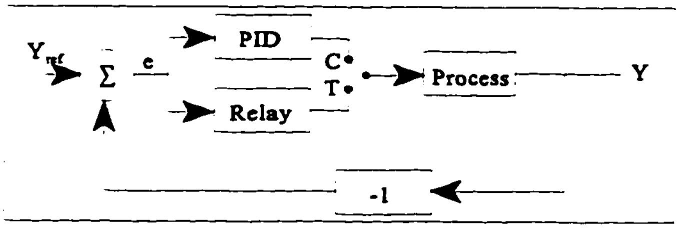

Figure 4.10. Astrom PID relay auto-tuner.

a straight line at a given speed and the *ultimate gain* and *period* are calculated. The *ultimate gain* can be estimated as

$$K_u = \frac{4d}{\pi a}, \tag{4.9}$$

where, d is the amplitude of the commanded steering angle and a is the amplitude of the sustained oscillations.

Figure 4.11 illustrates the procedure. When the error in heading is positive, the command steering angle is set to a constant negative value, which corresponds to a turn right.

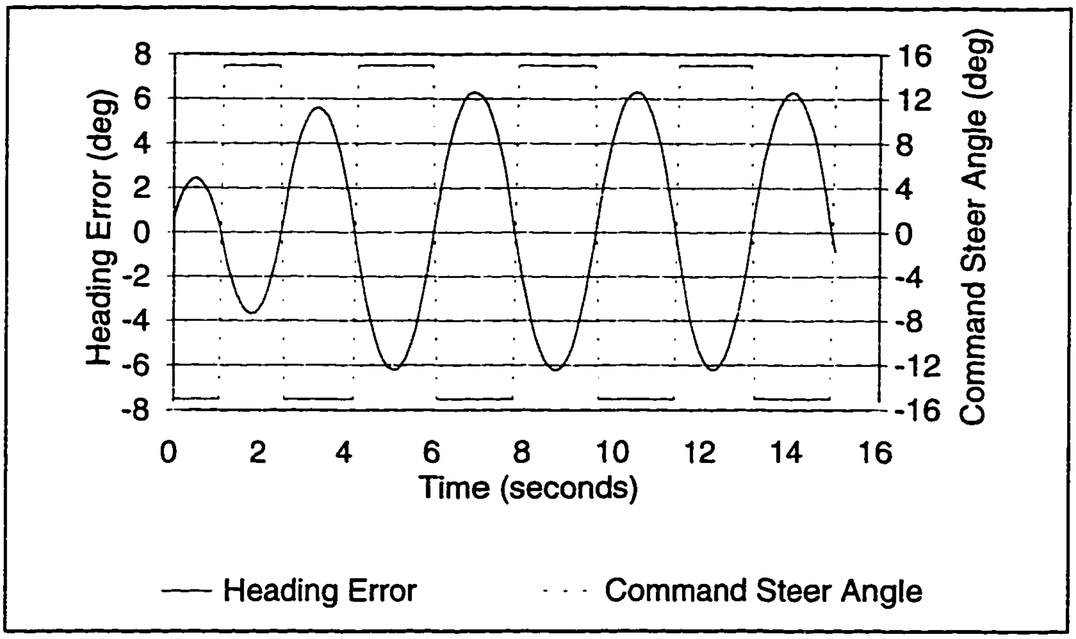

Figure 4.11. Simulation of PID relay auto-tuning at 1.34 m/s.

Similarly, when the heading error is negative, the command steering angle is set to a constant positive value, which corresponds to a turn left.

Auto-Tuning the Look-Ahead Distance

If the look-ahead distance is too small, path following could become oscillatory. (Under pure pursuit, but not PID, the system would probably need to be perturbed if the vehicle is started on-path.) If the look-ahead distance is too long, however, the vehicle will begin to cut the path corners.

The look-ahead distance is autonomously tuned with an implementation of the golden section search (GSS) optimization method. A lower and upper bound on the search region is estimated. The optimum look-ahead distance is targeted by progressively eliminating portions of the search region that are not expected to contain it. The procedure is an iterative process which is complete when the search region is reduced to 0.03 m or less.

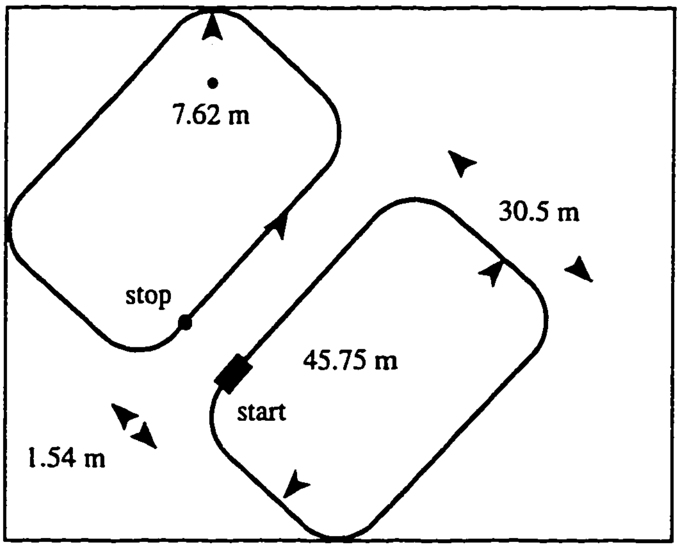

Figure 4.12. Calibration paths for auto-tuning the look-ahead distance.

With the press of a button the procedure begins. The calibration paths shown in Figure 4.12 are used. First, a path is generated 1.54 m to the left of the vehicle. The vehicle then executes the path using a GSS specified look-ahead distance. During path execution, the area between the calibration path and the path traced out by the vehicle is calculated as a performance index. When the vehicle arrives at the goal, the performance index is passed to the GSS subroutine and it returns the next recommended look-ahead distance.

The left and right paths are alternated so that the vehicle does not work its way across the test field. The calibration paths are always oriented parallel to the vehicle's current heading. The GSS subroutine reports when the search region is smaller than the specified tolerance. The GSS is used to auto-tune the look-ahead distance for both the PID and pure pursuit methods.

A variation of this method is to start the vehicle on the generated path. When the vehicle has traveled one-third of the first leg of the calibration path, the *auto-tune* program sends a message to the *backseat* program to translate the path 1.54 m normal to the vehicle's start heading. Here, the vehicle is allowed to reach the operational speed at which the look-ahead distance is to be tuned prior to perturbing the system. If the vehicle is tracking the left calibration path, it is translated to the left. If the vehicle is tracking the right calibration path, it is translated to the right.

During auto-tuning, a constant velocity is commanded. Experiments have shown that the performance index can be noisy due to fluctuations in the vehicle's velocity. As a result, the GSS may at times eliminate the region containing the optimum look-ahead distance. To minimize the effect of this, all the data collected during a search are fit with a cubic polynomial using the method of least squares. The minimum of this cubic (between the lower and upper bounds) is recorded in a data file as the optimum look-ahead distance for this commanded velocity.

Gain Scheduling

The optimum look-ahead distance is a function of the vehicle's velocity. (Evidence of this fact will be shown in the *Results and Conclusions* chapter.) As a result, the procedure for auto-tuning the look-ahead distance should be repeated for several speeds. In a nongraphical simulation program, the procedure is repeated for 10 speeds between 0-4.5 m/s. The optimum look-ahead distance for each speed is recorded in a data file.

In a graphical path tracking simulation program, the data file is read and a look-up table is loaded with the optimal gain data. The user can increase or decrease the vehicle speed by toggling a key on the keyboard. Each cycle through the program, the look-up table is accessed and the optimum look-ahead distance for the current speed is returned. Linear interpolation is performed between entries in the look-up table.

Gain scheduling has not been implemented on the NTV since the current speed controller is tuned for one speed (1.34 m/s).

CHAPTER 5
COMPUTER SIMULATION

Computer simulations are recognized and used widely as valuable design and testing tools. The benefits of a robot simulation are numerous. They can be used to develop control algorithms for a robot and verify the correct operation of the robot. Initial development and testing can be accomplished in simulation, making the robot available for other use. Initial testing can be accomplished in an air-conditioned environment. Furthermore, it is easier to detect motion problems in a control algorithm by observing the simulated motion of the robot rather than analyzing a set of printed data.

The simulation discussed in this chapter is used to develop and test control algorithms that improve the tracking of paths by a car-like vehicle (which contains an Ackerman steering linkage) and a two-wheeled trailer. An auto-tuning program was written and tested in simulation to autonomously tune the control parameters for a steered-wheeled robotic vehicle. An accurate representation of the trailer's motion would be useful in developing autonomous backing-up control algorithms and path tracking control algorithms where the control point is the center point of the axle on the trailer.

Modeling the NTV's Steering Behavior

A major component of the simulation program is estimating the vehicle's current position and heading at 10 Hz. This estimate is based on the motion of the steering wheel and the vehicle velocity. This section reviews the author's work in modeling the motion of the NTV's steering motor.

The *front-seat* program of the high-level controller calculates a command steering position at 10 Hz. This command steering position is communicated to an IndustryPack Servo Module (manufactured by GreenSpring Computers) by the *low-level* program. The servo module is responsible for moving the steering servo motor to the commanded position.

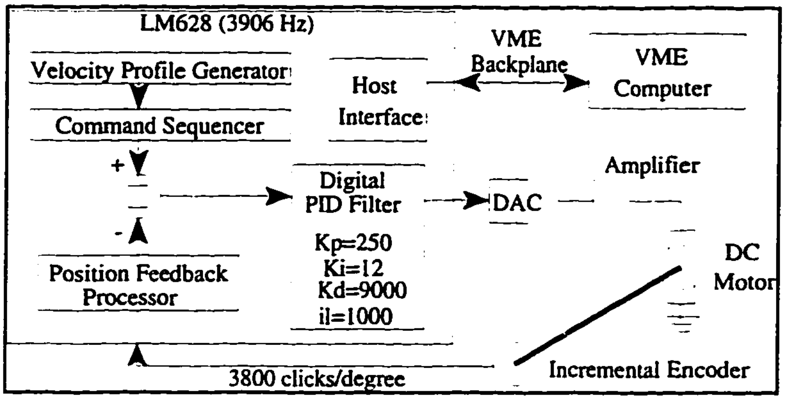

Figure 5.1. Low-level steering controller.

Figure 5.1 outlines the steps taken by the low-level controller in controlling the steering motor. The *low-level* program, which currently runs on a VME computer, communicates steering commands over the VME bus to the LM628 Precision Motor Controller of the servo module. Each time the host interface receives a command, a velocity profile generator calculates a profile of velocities that will accomplish the command.

As shown in Figure 5.2, the steering wheel will normally accelerate at 121.3 °/s^2 to a maximum velocity of 33.4 °/s, remain constant for a while, and then decelerate to a stop at the commanded position. The velocity profile is used to generate intermediate command positions at 3906 Hz. The difference in this position and the current steering position is the

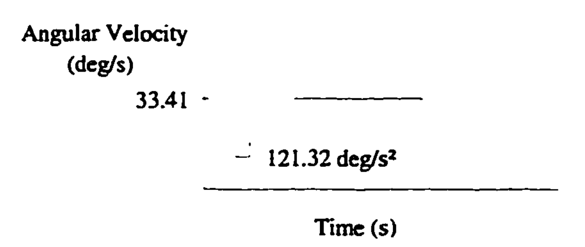

Figure 5.2. A velocity profile is generated for moving to the command position.

error the low-level PID controller works to drive to zero. The low-level PID controller was tuned by Mr. Phillip French [Fre94].

Figure 5.3 shows the steering motor's response to a step input of 28 °, as logged by the *low-level* program at 60 Hz. The top line is the step input, the middle line is the intermediate profile positions, and the bottom line is the feedback positions. The steering

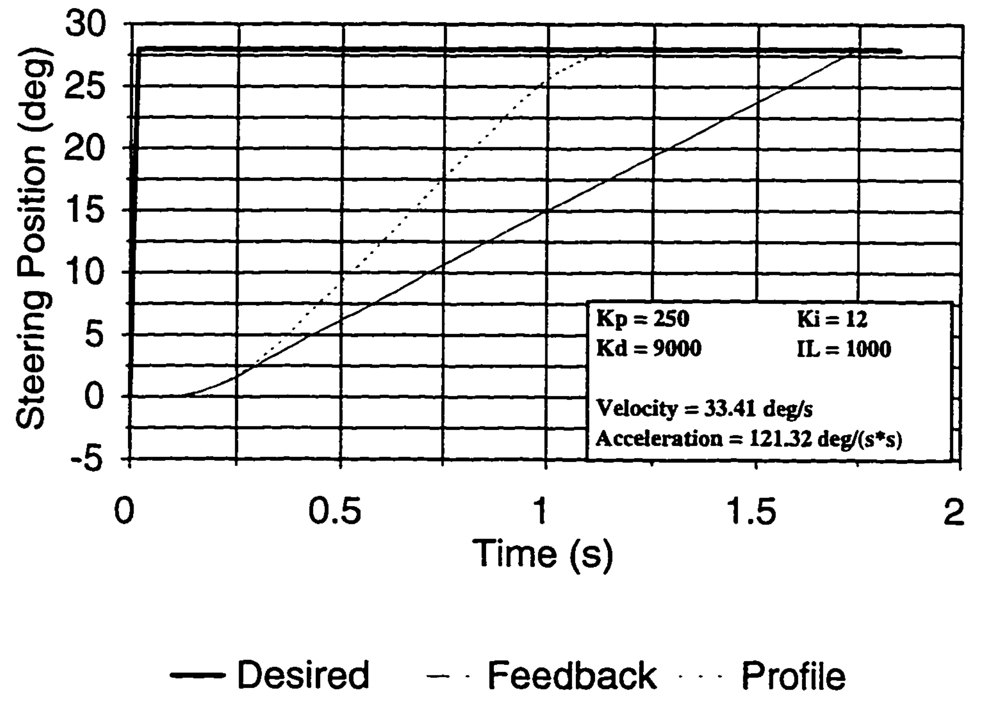

Figure 5.3. NTV steering response to a step input of 28 °.

motor turns at a maximum speed of 17.5 °/s and can't keep up with the command profile, which moves at a maximum speed of 33.4 °/s. The low-level gains and profile velocity and acceleration are adjustable parameters set in software.

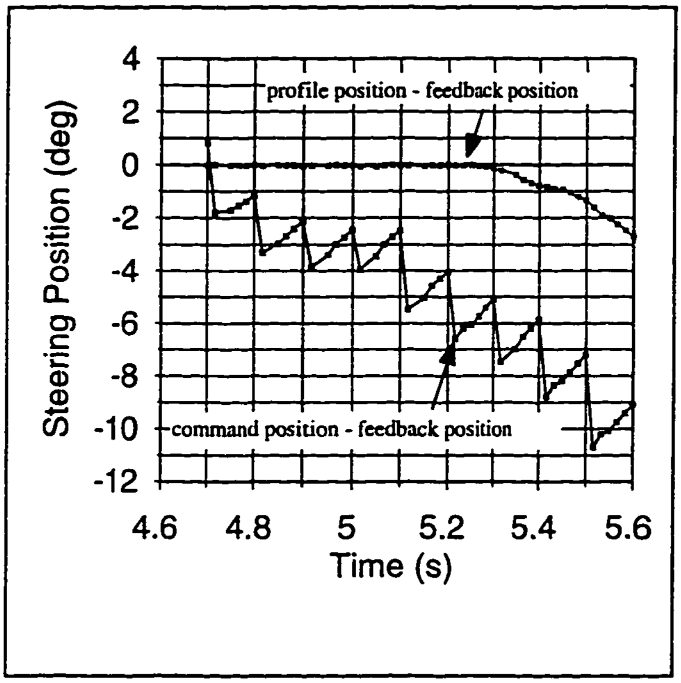

Figure 5.4. When the error is small, the feedback position is approximated as the profile position.

The author has observed that when the steering error (the difference between the command position and the feedback position) is small, then the difference between the profile and feedback position is negligible. This is shown in Figure 5.4. As was seen in Figure 5.3, when the error is large, the steering position will change a maximum of 0.292° every 60th of a second. This observation was tested during autonomous navigation.

A prerecorded path (near the Mechanical Engineering Building of the University of Florida) is shown in Figure 5.5. The path points are separated by 1.0 m. This path was autonomously executed at 1.34 m/s and the steering profile and feedback positions were recorded at 60 Hz. An analysis of the data showed that the maximum difference between the profile and feedback position was 0.052°. The difference between the two was less than 0.01° 91% of the time.

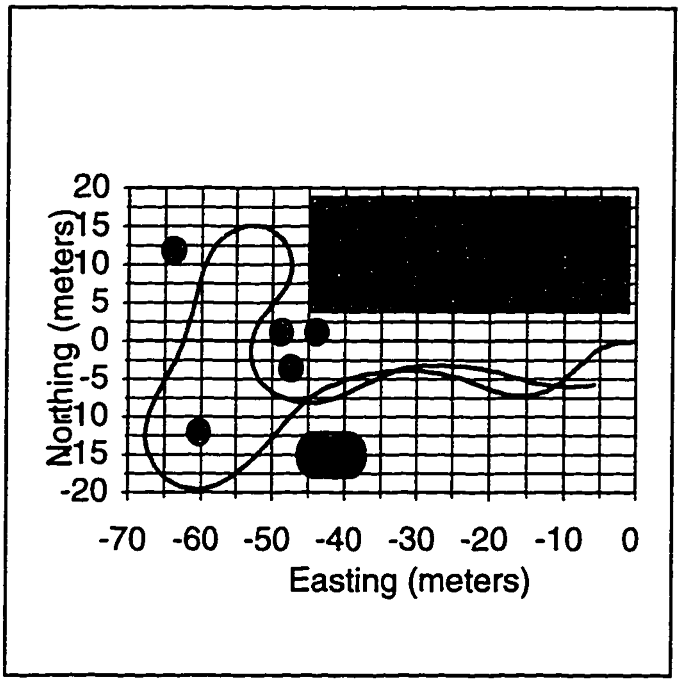

Figure 5.5. A prerecorded path autonomously executed by the NTV.

The *subgoal generator* ensures a smooth path. And when the path is smooth, the steering error remains small at slow speeds, even when the path curvature is large. In the simulation, a velocity profile is generated every time the *front-seat* program calculates a new command steer angle. The current steering angle is updated at 10 Hz using the equations

$$\Delta\phi = \phi_{current} - \phi_{profile}$$
$$\phi_{current} = \begin{cases} \phi_{profile}, & if(|\Delta\phi| < 1.75°) \\ \phi_{current} + 1.75°\, sign(\Delta\phi), & otherwise \end{cases} \tag{5.1}$$

Simulating Path Tracking

The following list describes the three phases involved in simulating the execution of a specific path by an animated robot vehicle and its trailer.

1) *Plan a safe and achievable path.*
2) *Initialize the current vehicle and trailer pose.*
3) *Draw the vehicle, trailer, planned path, actual path, and the environment until the vehicle is within some tolerance of the goal position and heading.*
   a) *Determine the control parameters from a look-up table.*
   b) *Calculate a command steering angle and vehicle velocity.*
   c) *Determine the new vehicle and trailer pose.*
   d) *Draw the environment.*
   e) *Draw the path to be executed.*
   f) *Draw the vehicle and its trailer at their new pose.*
   g) *Draw a path indicating the actual path of the vehicle or trailer.*

After planning a safe and achievable path and initializing the current vehicle and trailer pose, the steps under phase three are repeated at 10 Hz until the vehicle arrives at the goal position. If the method of steering control involves gains that change as a function of the current vehicle velocity, the gains can be calculated by interpolating the data in a predefined look-up table.

A command steering angle and vehicle velocity are then calculated using controlling equations specific to the method of vehicle control. A subroutine is then called that updates the vehicle and trailer pose, given that the vehicle was directed to use a command steering angle and command vehicle velocity, and assuming the vehicle has traveled 2.54 cm since the last update. The environment, planned path, actual path, vehicle, and trailer are all drawn in their proper prospective in three dimensions.

It should be noted that the steps under phase three may not in reality run at 10 Hz. No timer and delay loops are used here. The simulation is allowed to run as fast as the

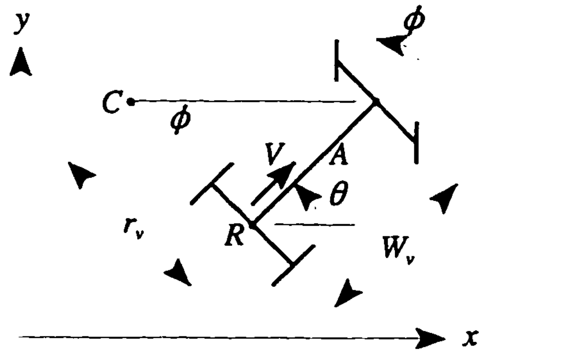

Figure 5.6. A car-like vehicle with velocity v and steering ange ϕ.

system can perform the calculations and draw all the objects. Thus, position updates "symbolically" occur at 10 Hz in the simulation. The focus of this section is the methodology used to determine the current vehicle and trailer pose at this rate.

Kinematic Constraints

Consider the car-like vehicle shown in Figure 5.6, where the controlled point is chosen to be R, the center of the rear axle of the vehicle. The pose of the vehicle is specified by the three parameters (x,y,θ), where (x,y) are the coordinates of point R and θ is the orientation of the vehicle with respect to the x-axis of a global coordinate system. Hence, the configuration space of the car-like vehicle is three dimensional. Assuming no slip, the instantaneous velocity of the control point R is along the main axis A of the vehicle. The space of achievable velocities at any vehicle configuration is thus two dimensional. The coordinate components of the instantaneous velocity v are

$$\frac{dx}{dt} = v\cos\theta, \quad \frac{dy}{dt} = v\sin\theta \tag{5.2}$$

The equations in (5.2) are combined to yield equation (5.3), a nonintegratible equation of motion referred to as a *nonholonomic equality constraint.*

$$-\sin\theta\, dx + \cos\theta\, dy = 0 \tag{5.3}$$

Nonholonomic constraints are generally caused by one or more rolling contacts between rigid bodies, where the relative velocity between the contact points is zero. Car-like robotic vehicles are thus typical nonholonomic mechanical systems. The only other kinematic constraint on the motion of a car-like vehicle is an upper bound on the steering

$$|\phi| \leq |\phi_{max}| < \frac{\pi}{2} \tag{5.4}$$

angle, ϕ, as expressed in equation (5.4). This equation can be rewritten as a *nonholonomic inequality constraint* involving the differential motions dx, dy, and dθ [Lat90].

Laumond [Lau86] has proved that in spite of the kinematic constraints, a car-like robot is fully controllable. In an environment which does not contain obstacles, a car-like robot can achieve any position in the *xy* plane with any orientation.

Assumptions

Due to the nonholonomic equality constraint, the space of differential motions (dx, dy, dθ) of the robot at any pose (x,y,θ) is a two dimensional space. The instantaneous motion of the point *R* is determined by two parameters: the linear velocity *v* of point *R*, and the vehicle's current steering angle, ϕ. Here, it is assumed that the feedback steering angle (obtained from a steering column encoder) can be approximated in simulation by modeling the vehicle as a tricycle. Although the steering angle constantly changes during vehicle execution of a curvilinear path, it is sufficient to assume the vehicle and its trailer travels on circular arcs when the incremental arc distance, *Δs*, is small. The steering angle is considered a constant over this interval. In determining the steering angle to use during the next incremental arc distance, the time *Δt* it takes to travel *Δs* is estimated and a new steering position is calculated using a steering angular velocity profile.

It is assumed that no wheel slippage occurs and that perfect velocity tracking occurs, i.e., once the command vehicle velocity is obtained, it is maintained. Over the incremental arc distance, *Δs*, the velocity *v* is considered a constant. In determining the vehicle velocity to use during the next time interval *Δt*, it is assumed that the acceleration/deceleration of the vehicle is constant when the current vehicle velocity is not equal to the command vehicle velocity.

Positioning Algorithm

The model discussed in the section *Modeling the NTV's Steering Behavior* of this chapter is used in updating the vehicle and trailer's position and heading in simulation. Equation (5.1) expresses the observation that the feedback steering position can be approximated by the profile position when the error is small. Otherwise, the steering wheel moves at a constant rate of 17.4 °/s. Equation (5.5) expresses the assumption that the vehicle acceleration/deceleration is a constant, k_v, when the current vehicle velocity does not equal the command vehicle velocity.

$$\dot{v} = \begin{cases} k_v \; for \; (v_{cur} \neq v_{cmd}) \\ 0 \; for \; (v_{cur} = v_{cmd}) \end{cases} \tag{5.5}$$

The algorithm for updating the vehicle and the trailer is given below.

```
updateVehicleTrailerPose($\phi_{cmd}$, $\phi_{cur}$, $v_{cmd}$, $v_{cur}$)
{
        time = 0, done = FALSE
        profile = getSteeringVelocityProfile($\phi_{cur}$, $\phi_{cmd}$)
        while(NOT done)
        {
                $\Delta t = 0.0254/v_{cur}$
                if(time + $\Delta t \geq 0.1$)
                {
                        done = TRUE
                        $\Delta t = 0.1$ - time
                        $\Delta s = v \cdot \Delta t$
                }
                else
                {
                        $\Delta s = 0.0254$
                        time = time + $\Delta t$
                }
                $\phi_{cur}$ = getNewSteerPosition(profile, $\phi_{cur}$, $\Delta t$)
                if ($v_{cur} < v_{cmd}$)          $v_{cur} = min(v_{cmd}, v_{cur} + k_v \Delta t)$
                else if ($v_{cur} > v_{cmd}$)     $v_{cur} = max(v_{cmd}, v_{cur} - k_v \Delta t)$
                else                              $v_{cur} = v_{cmd}$
                pose = getNewPose(pose, $\phi_{cur}$, $v_{cur}$, $\Delta s$)
        }
}
```

The subroutine *updateVehicleTrailerPose* is called 10 times per second. During each call, the position and heading of the vehicle and trailer is locally updated every Δs=2.54 cm. The time, velocity, angle, and distance units are respectively seconds, m/s, degrees, and meters. The next two sections will present the equations used in the *getNewPose* subroutine to update the position and heading of the simulated vehicle and its trailer.

Updating the Vehicle Pose

When the steering angle is nonzero, the vehicle moves around a circular arc centered at C and having a radius of r_v, as shown in Figure 5.6. The turning radius r_v of the vehicle is related to the steering angle ϕ by the equation

$$r_v = \left|\frac{W_v}{\tan\phi}\right|, \tag{5.6}$$

where W_v is the ***wheelbase***, the distance between the vehicle's front and rear axles. Over the time step, point A moves to point B along arc Δs at a constant velocity v, as shown in Figure

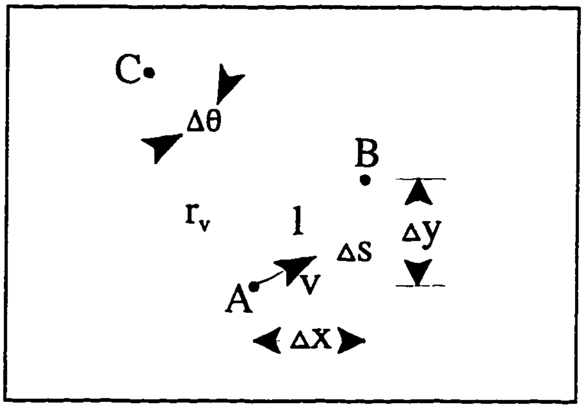

Figure 5.7. The vehicle moves from point A to point B along circular arc Δs during Δt.

5.7. The change in heading, $\Delta\theta$, can be expressed as a function of the arc length Δs,

$$\Delta\theta = \frac{\Delta s}{r_v}, \tag{5.7}$$

and the arc length Δs can be expressed as a function of the time interval and instantaneous velocity v.

$$\Delta s = v\Delta t \tag{5.8}$$

Substituting equations (5.6) and (5.8) into equation (5.7) yields the following expression for

$\Delta\theta$ in terms of known values.

$$\Delta\theta = \frac{v\Delta t \tan\phi}{W_v} \tag{5.9}$$

The length of the line segment, l, can be shown to be

$$l = 2r_v \sin(\frac{\Delta\theta}{2}) \tag{5.10}$$

When the steering angle is nonzero, the differential changes (Δx,Δy) in the vehicle's position (x,y) are

$$\Delta x = l\cos[\theta + sign(\phi)\frac{\Delta\theta}{2}], \qquad \Delta y = l\sin[\theta + sign(\phi)\frac{\Delta\theta}{2}] \tag{5.11}$$

When the steering angle is zero, $r_v = \infty$, $\Delta\theta = 0$, and the differential changes (Δx,Δy) in the vehicle's position (x,y) are

$$\Delta x = v\Delta t\cos\theta, \qquad \Delta y = v\Delta t\sin\theta \tag{5.12}$$

The current pose of the vehicle is updated locally inside the *getNewPose* subroutine at 2.54 cm intervals using the equations

$$\begin{aligned} \theta_{v,new} &= \theta_{v,old} + \Delta\theta \\ x_{v,new} &= x_{v,old} + \Delta x \\ y_{v,new} &= y_{v,old} + \Delta y \end{aligned} \tag{5.13}$$

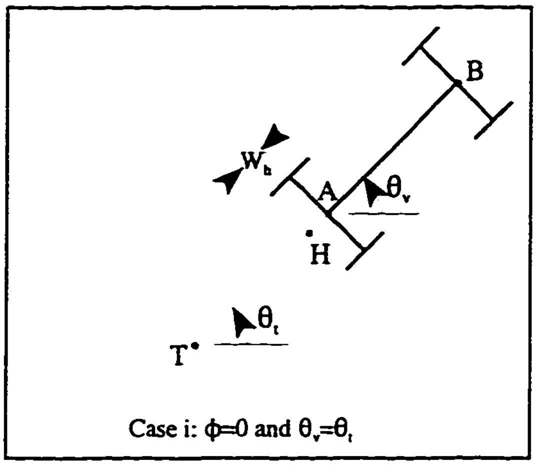

Figure 5.8. Vehicle/trailer configuration (*i*).

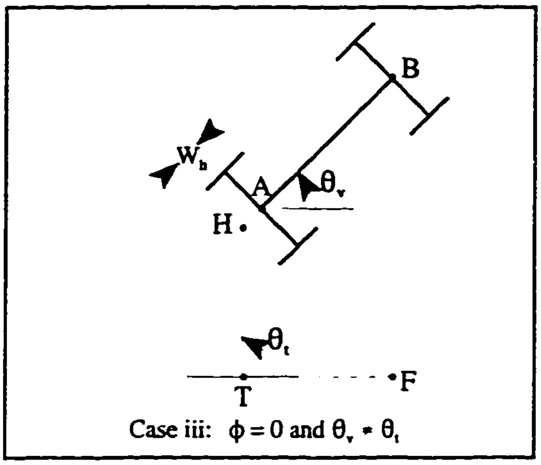

Figure 5.9. Vehicle/trailer configuration (*iii*).

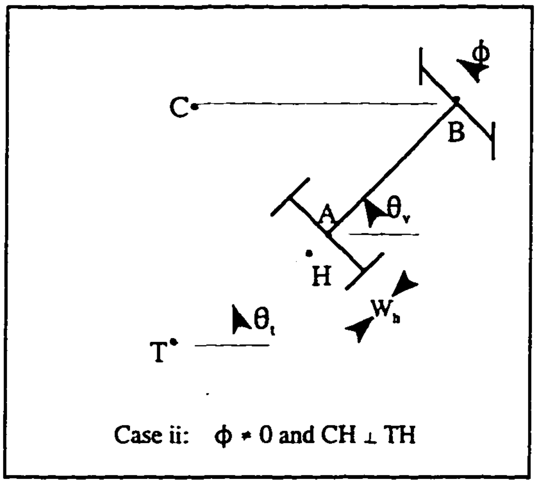

Figure 5.10. Vehicle/trailer configuration (*ii*).

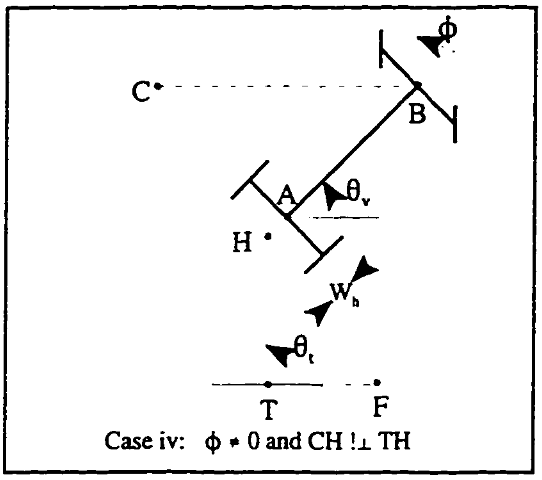

Figure 5.11. Vehicle/trailer configuration (*iv*).

Updating the Trailer Pose

If the vehicle tows a trailer, then a hitch point and some position on the trailer must be updated each time the vehicle's position (x,y) is updated. Figures 5.8-5.11 illustrate four general cases used to determine the motion of the trailer. For all of the cases, the updated hitch point can be calculated given the updated vehicle pose, as shown in equation (5.14).

$$\begin{aligned} x_{h,new} &= x_{v,new} - W_h \cos(\theta_{v,new}) \\ y_{h,new} &= y_{v,new} - W_h \sin(\theta_{v,new}) \end{aligned} \tag{5.14}$$

Case (*i*) is the simplest case. Here, the vehicle is traveling straight (ϕ=0) and the trailer heading coincides with the vehicle heading. In case (*ii*), the hitch point *H* moves along

$$\theta_t = \theta_v - sign(\phi)\tan^{-1}\left[\frac{W_h \tan\phi}{W_v}\right] \tag{5.15}$$

a circular arc centered at *C* and having a radius of *CH*. Here, *CH* is perpendicular to the main axis of the trailer (*TH*) and the instantaneous velocity of point *T* is in the direction of the main axis of the trailer. *CH* is perpendicular to *TH* when θ_t is equal to the expression in equation (5.15). For case (*i*) and (*ii*), the new trailer pose is calculated by the equations in (5.16).

$$\begin{aligned} \theta_{t,new} &= \theta_{t,old} \\ x_{t,new} &= x_{h,new} - W_t \cos(\theta_{t,new}) \\ y_{t,new} &= y_{h,new} - W_t \sin(\theta_{t,new}) \end{aligned} \tag{5.16}$$

In case (*iii*), the steering angle is zero, but the trailer heading does not coincide with the vehicle heading. Case (*iv*) is the most general case for a vehicle executing a turn. Here, the steering angle is nonzero and *CH* is not perpendicular to *TH*. It is assumed that in both case (*iii*) and (*iv*), the trailer point *T* moves about the circular arc centered at *F* and having radius *TF*. In case (*iii*), point *F* is found using the equations in (5.17).

$$
\begin{aligned}
k &= sign(\boldsymbol{HB} \times \boldsymbol{HT}) \\
x_f &= -\frac{kW_t \sin(\theta_{v,old})}{\sin(\theta_{t,old} - \theta_{v,old})} + x_{h,old} \\
y_f &= \frac{kW_t \cos(\theta_{v,old})}{\sin(\theta_{t,old} - \theta_{v,old})} + y_{h,old}
\end{aligned}
\tag{5.17}
$$

For case (*iv*), point *C* is given by the equations in (5.18),

$$
x_c = x_{v,old} - sign(\phi) r_v \sin(\theta_{v,old}), \qquad y_c = y_{v,old} + sign(\phi) r_v \cos(\theta_{v,old}) \tag{5.18}
$$

and point *F* can be found by determining the intersection of the line through *CH* and the line through the axle of the trailer.

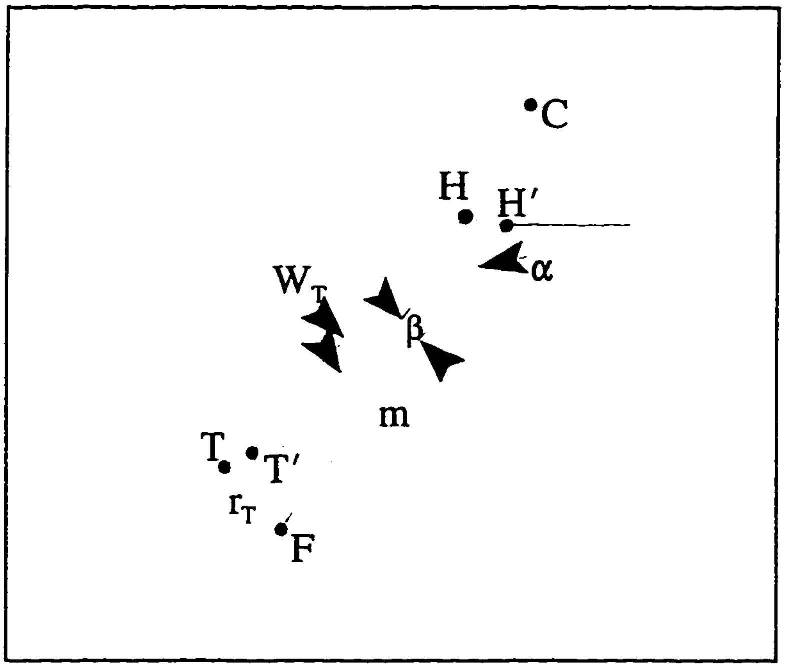

Figure 5.12. Hitch point H moves to H′ and trailer point T moves to T′

As shown in Figure 5.12, the hitch point H moves to the point H' and the trailer point T moves to the point T'. Once point F is found for case (*iii*) and (*iv*), the algorithm for finding the new trailer position, T', and its heading, is then as follows:

$$\begin{gathered}
m = \sqrt{(x_{h,new} - x_f)^2 + (y_{h,new} - y_f)^2} \\
r_t = \sqrt{(x_{t,old} - x_f)^2 + (y_{t,old} - y_f)^2} \\
\beta = \cos^{-1}\left[\frac{W_t^2 + m^2 - r_t^2}{2W_t m}\right] \\
\alpha = atan2(y_f - y_{h,new}, x_f - x_{h,new}) \\
k = sign(\boldsymbol{HF} \times \boldsymbol{HT}) \\
x_{t,new} = x_{h,new} + W_t \cos(\alpha + k\beta) \\
y_{t,new} = y_{h,new} + W_t \sin(\alpha + k\beta) \\
\theta_{t,new} = atan2(y_{h,new} - y_{t,new}, x_{h,new} - x_{t_{new}})
\end{gathered} \tag{5.19}$$

Simulating Auto-Tuning

The equations developed in the *Simulating Path Tracking* section of this chapter are incorporated into a simulation that auto-tunes the steering control parameters of a steered-wheeled robotic vehicle. The *Auto-Tuning Control Parameters* section of chapter 4 discusses the procedures used for auto-tuning the control parameters.

The auto-tuning procedure is a time consuming process, therefore, this simulation is nongraphical. First, the user is asked if he/she wants to tune the PID or pure pursuit controller. Next, the user is asked if he/she wants to use an oval shaped path, as shown in Figure 4.12, or a *figure 8* shaped path. A subroutine generates a *figure 8* given the length of the straight sections and the angle between the straight sections. A *figure 8* path is beneficial because it contains both a left and a right turn.

If the PID controller was chosen, then the [Ast95] relay tuner and the Ziegler-Nichols tuning rules [Zie42, Ast95] are used to generate an optimal proportional gain, integral time, and derivative time for 10 speeds between 0-4.5 m/s. Theses gains are written to a file. During path following, the gains are loaded to a look-up table. They are retrieved using linear interpolation, based on the user defined command velocity.

For both the PID and pure pursuit controller, the look-ahead distance is tuned using the golden section search (GSS) optimization technique for 10 speeds between (0-4.5 m/s). The performance index is the area between the planned path and the path traced out by the vehicle's control point. The vehicle's position is translated 1.54 m at the beginning of each lap to initially perturb the system.

These gains are also written to a file and loaded to a look-up table during path following. They are retrieved using linear interpolation, based on the user defined command velocity.

CHAPTER 6
RESULTS AND CONCLUSIONS

The high-level control strategy discussed in chapter 4 was demonstrated in simulation (as discussed in chapter 5) and implemented on the NTV. This chapter presents some results from tests run on the simulation programs and the NTV. It ends with some conclusions that can be drawn from this research.

Simulation Results

The model of the NTV's steering behavior that was discussed in chapter 5, was implemented for the simulation programs on a Silicon Graphics IRIS Crimson computer. Figure 6.1 shows the NTV and simulation steering response to a step input of 28.0°. Figure

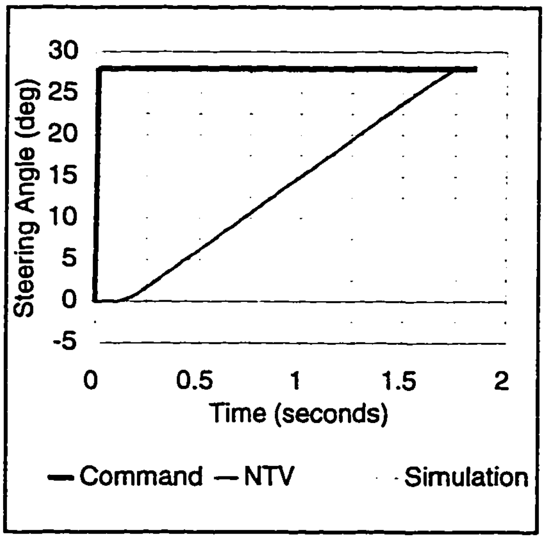

Figure 6.1. NTV and simulation response to a step input of 28°.

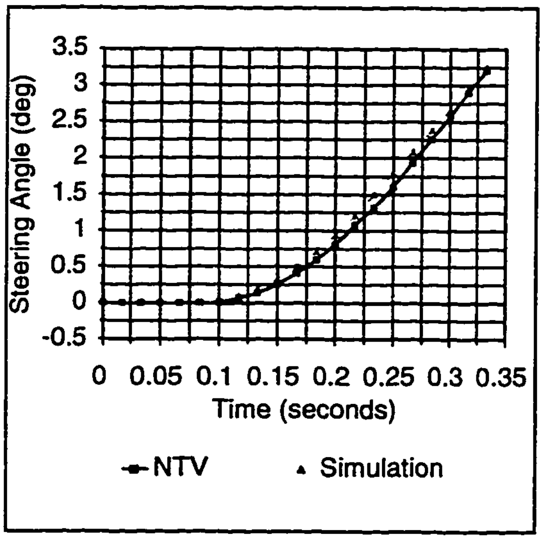

Figure 6.2. First 1/3 second in Figure 6.1.

6.2 shows the first 0.33 seconds following the command. As discussed in chapter 5, the steering position can be approximated by the profile position, calculated from a generated velocity profile, except when the profile velocity exceeds 17.52 °/s. This appears to be the maximum angular velocity of the steering servo motor.

The raw data from auto-tuning the pure pursuit look-ahead distance for 10 speeds (0-4.5 m/s) can be found in Appendix A. An implementation of the golden section search technique is used to progressively home in on the optimum look-ahead distance. For this set of data, the look-ahead distance was defined as L_a, as shown in Figure 4.2. At the beginning of each lap around the rectangular test path shown in Figure 4.12, an initial lateral error of 1.54 m is introduced.

The pure pursuit data shown in Appendix A is summarized in Figure 6.3. The upper line in the figure shows that the optimal look-ahead distance increases as the vehicle speed increases. The lower line shows that the lateral path error also increases as the vehicle speed does. Over the velocity range, the optimal look-ahead distance varied between 2-4 m and the average deviation varied between 10-25 cm.

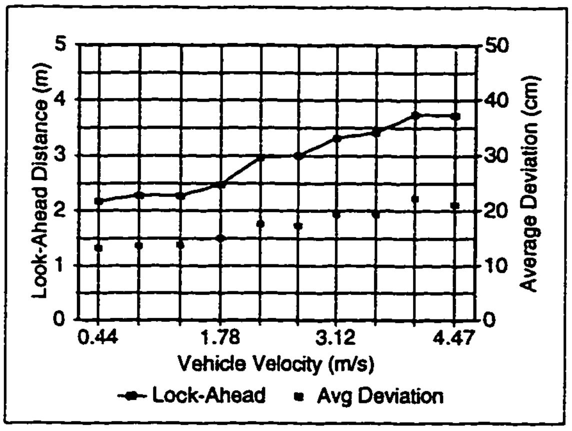

Figure 6.3. Pure pursuit auto-tuning results.

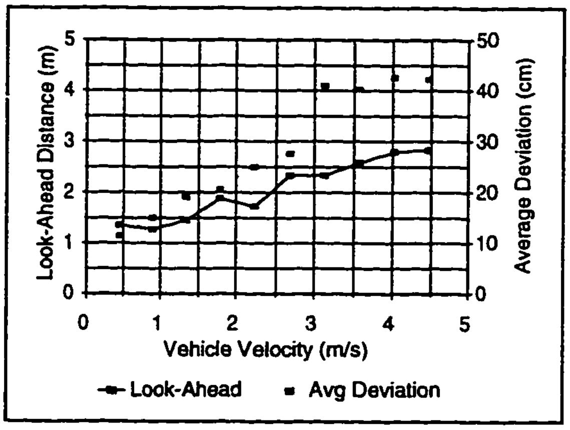

Figure 6.4. PID auto-tuning results.

The raw data from auto-tuning high-level PID controller is also listed in Appendix A. The proportional gain *K*, integral time T_i, and derivative time T_d, are shown in Table 6.1 for the 10 test speeds. It is noted that the proportional gain does not change in a uniform way. The integral and derivative times do not change much beyond 2.68 m/s.

Table 6.1. PID gains derived from auto-tuning.

Speed (m/s)	K	T_i	T_d
0.45	2.666	4.478	1.119
0.89	1.916	2.711	0.678
1.34	1.727	1.911	0.478
1.79	1.392	1.711	0.428
2.24	1.309	1.411	0.353
2.68	1.231	1.211	0.303
3.13	1.034	1.211	0.303
3.58	0.899	1.211	0.303
4.02	0.801	1.211	0.303
4.47	0.649	1.300	0.325

Results from auto-tuning the look-ahead distance for the PID controller is shown in Figure 6.4. The PID gains determined for each speed, as shown in Table 6.1, were used for this simulation test. As with the pure pursuit method, the optimal look-ahead distance increases with vehicle speed. The average deviation also increases with vehicle speed. The look-ahead distance varied between 1-3 meters and the average deviation varied between 10-45 cm.

Both the pure pursuit and PID steering controllers were easy to implement. The PID controller appears to be poorly tuned by the relay auto-tuner. Auto-tuning provides merely a starting point for further fine tuning. The author was able to improve on the path following accuracy by manual tuning using trial and error modifications. As tuned by the auto-tuning procedures, there are advantages and disadvantages associated with both of the steering controllers.

The PID controller is stable when the vehicle velocity is low or the lateral path error is large. (The look-ahead distance is large when the lateral path error is large). The disadvantages of the PID controller are 1) it is difficult to accurately auto-tune, 2) it is unstable when the vehicle velocity is high, and 3) even when it is accurately tuned, there is an inherent error that causes corners to be cut short.

Figure 6.5. PID control with short look-ahead at 1.34 m/s.

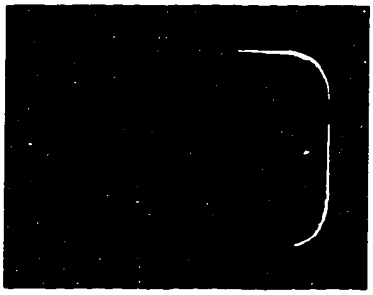

Figure 6.6. PID control with larger look-ahead at 1.34 m/s.

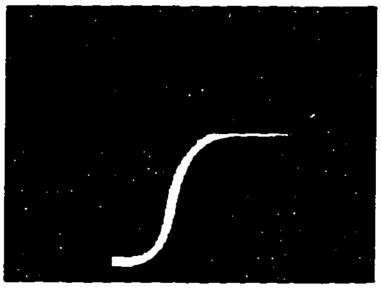

Figure 6.7. Off-path to on-path transition using PID control at 1.34 m/s.

As the look-ahead distance is shortened, the lateral error starts to decrease but control becomes jittery. Micro-oscillations are produced, as shown in Figure 6.5. As the look-ahead distance is increased, as shown in Figure 6.6, the lateral error begins to increase around corners but the control is smoother. When approaching a path from a distance, the transition from off-path to on-path travel is smooth, as shown in Figure 6.7.

The advantage of the pure pursuit method is that the steering control is stable and accurate when the lateral error is small, for all velocities tested. (The lateral error is initially small, and remains so, when the vehicle is started on-path). This controller is also easy to auto-tune if sufficient space is available. The disadvantage of this controller is that it can become unstable when the vehicle is making a transition from off-path to on-path travel.

Figure 6.8. Pure pursuit control with vehicle started on-path at 1.34m/s.

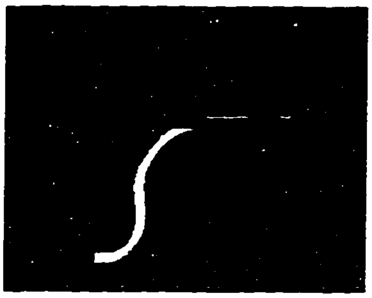
Figure 6.9. Off-path to on-path transition using pure pursuit control at 1.34 m/s.

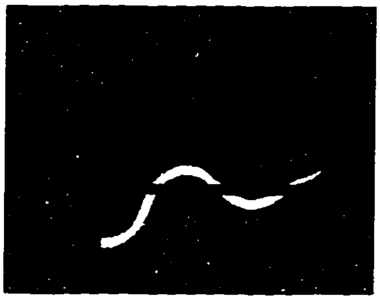
Figure 6.10. Off-path to on-path transition using pure pursuit control at 1.34 m/s.

Figure 6.8 illustrates the accuracy of path following using the pure pursuit method when the vehicle is started on path. When the vehicle approaches a path nearly perpendicular to it, as shown in Figure 6.9, the transition from off-path to on-path travel is fairly smooth. When the approach is from a smaller angle, the control can become unstable, as shown in Figure 6.10.

Even though the PID controller appears to be poorly tuned, it is the more stable method for approaching a path from off-path travel. On-path travel is more stable and accurate when the pure pursuit controller is used. An attempt was made to incorporate the

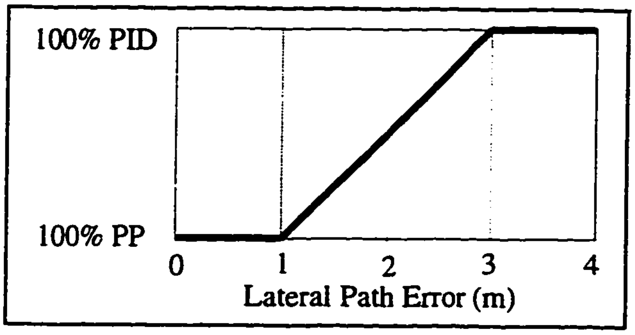

Figure 6.11. Weighted PID and pure pursuit control solution.

positive features of both control methods by using a weighted control solution. The weighted control solution is illustrated with Figure 6.11.

When the lateral path error is less than 1 meter, the recommended steering angle is taken to be the output from the pure pursuit controller. When the lateral path error is greater than 3 meters, the recommended steering angle is taken to be the output from the PID controller. And when the lateral path error is between 1-3 meters, the recommended steering angle is a combination of the pure pursuit and PID controllers. Figures 6.12 and 6.13 shows smooth transitions from off-path travel to on-path travel for a couple of different approach angles.

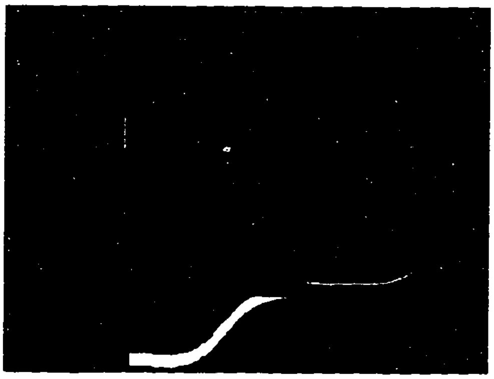

Figure 6.12. Off-path to on-path transition using weighted control at 1.34 m/s.

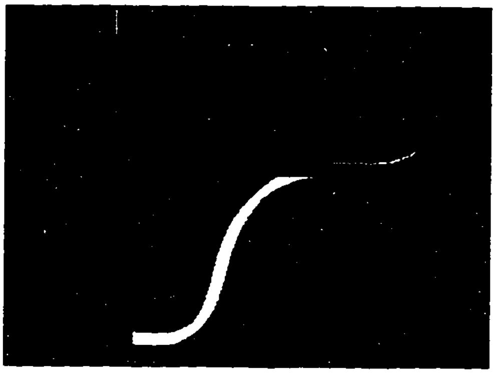

Figure 6.13. Off-path to on-path transition using weighted control at 1.34 m/s.

Navigation Test Vehicle Results

The high-level control strategy discussed in chapters 3 and 4 was implemented in simulation as well as on the NTV. The results from autonomous steering calibration, auto-tuning, and path following are provided in this section.

Autonomous steering calibration was performed to compensate for the effects of the Ackerman steering linkage, wheel slippage, and an inaccurate home steering position. The autonomous calibration procedure described in the *Autonomous Steering Calibration* section of chapter 4 was followed. Figure 6.14 shows the experimentally measured relationship between the command steering angle and the curvature of the turn for 40 samples taken on a grass surface while driving at 1.34 m/s. (Five samples were collected at each of eight steering positions.) Also shown in the figure is the expected relationship. The experimental data are fit with a line using the method of least squares and the slope (*m*) and y-intercept (*b*) are written to a calibration file.

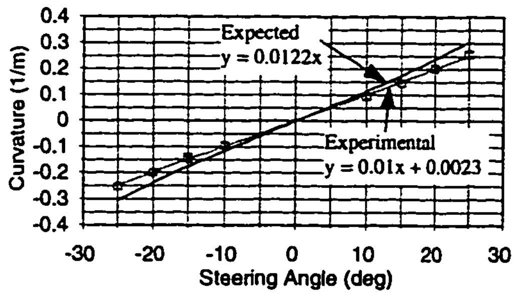

Figure 6.14. Relationship between the command steering angle and turn curvature based on 40 measurements.

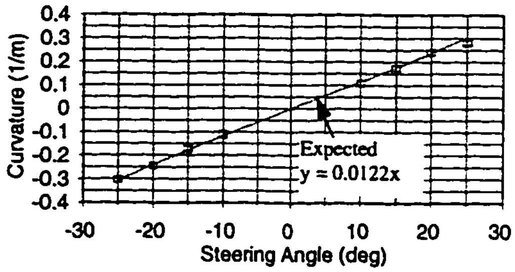

Figure 6.15. The adjusted steering angle yields the desired turn curvature for 40 measurements.

The relationship between the command steering angle ϕ_{cmd} that the high-level controller outputs and the expected curvature k_{cmd} is given by Equation 6.1,

$$k_{cmd} = \frac{\tan \phi_{cmd}}{W}, \tag{6.1}$$

where, W is the distance between the axles. To execute a turn with this curvature, the steering must be commanded to the position ϕ_{adj} as given by Equation 6.2.

$$\phi_{adj} = m k_{cmd} + b \tag{6.2}$$

The adjusted steering angle, as a function of the command steering angle, is given by Equation 6.3.

$$\phi_{adj} = \frac{m \cdot \tan \phi_{cmd}}{W} + b \tag{6.3}$$

The calibration procedure was run a second time for eight steering positions, five samples for each position. For each command steering position, the adjusted steer position in Equation 6.3 was executed. Figure 6.15 shows the measured curvature for each adjusted steer position. The measured curvature for all 40 tests are very close to the expected curvature.

The high-level PID controller was auto-tuned using the method outlined in the *PID Relay Auto-Tuning* section of chapter 4. This controller was previously tuned manually with no derivative gain (PI). In order to compare the results of the auto-tuned controller with the manually tuned controller, this controller was auto-tuned for PI control. The Ziegler-Nichols tuning rules for a PI controller are

$$\begin{aligned} K &= 0.4K_u \\ T_i &= 0.8T_u \end{aligned} \qquad (6.4)$$

Table 6.2. Auto-Tuned PID and PI gains at 1.34 m/s.

Sample	Characteristic Parameters		PID Gains			PI Gains	
	K_u	T_u	K_p	K_i	K_d	K_p	K_i
1	1.93	5.03	1.16	0.46	0.73	0.77	0.19
2	1.93	5.07	1.16	0.46	0.73	0.77	0.19
3	2.05	4.80	1.23	0.51	0.74	0.82	0.21
4	1.87	5.13	1.13	0.44	0.72	0.75	0.18
5	2.04	4.83	1.22	0.51	0.73	0.82	0.21
6	1.81	5.23	1.09	0.42	0.71	0.72	0.17
7	2.31	4.26	1.39	0.65	0.74	0.92	0.27
8	2.14	4.61	1.28	0.56	0.74	0.86	0.23
9	2.02	4.79	1.21	0.51	0.73	0.81	0.21
10	1.88	5.10	1.13	0.44	0.72	0.75	0.18
Average	2.00	4.89	1.20	0.50	0.73	0.80	0.20

The characteristic parameters (elicited through relay auto-tuning) and the corresponding PID and PI gains for 10 trials are shown in Table 6.2. These tests were performed on a fairly level grass surface at 1.34 m/s. For PID and PI control, the gains can be written in the form

$$\begin{aligned} K_p &= K \\ K_i &= \frac{K}{T_i} \\ K_d &= KT_d, \end{aligned} \tag{6.5}$$

where K_p is the *proportional gain*, K_i is the *integral gain* , and K_d is the *derivative gain.*

The average auto-tuned proportional and integral gains were 0.80 and 0.20, respectively. The manually tuned proportional and integral gains were 1.0 and 0.18, respectively.

The look-ahead distances for both the PI and pure pursuit controllers were tuned using the method outlined in the *Auto-Tuning the Look-Ahead Distance* section of chapter

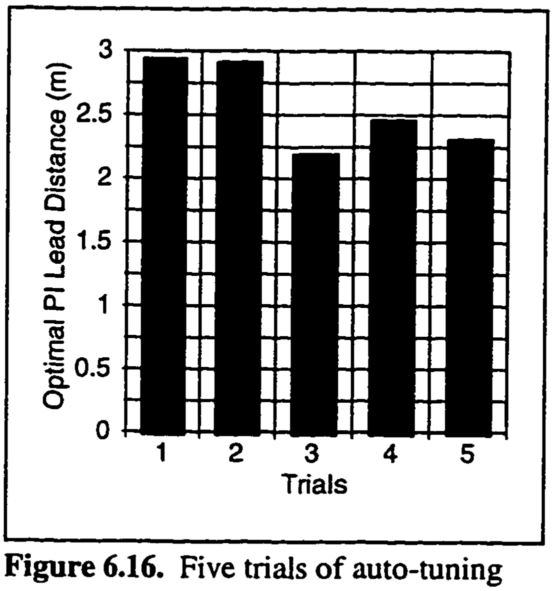

Figure 6.16. Five trials of auto-tuning the look-ahead distance for a PI controller.

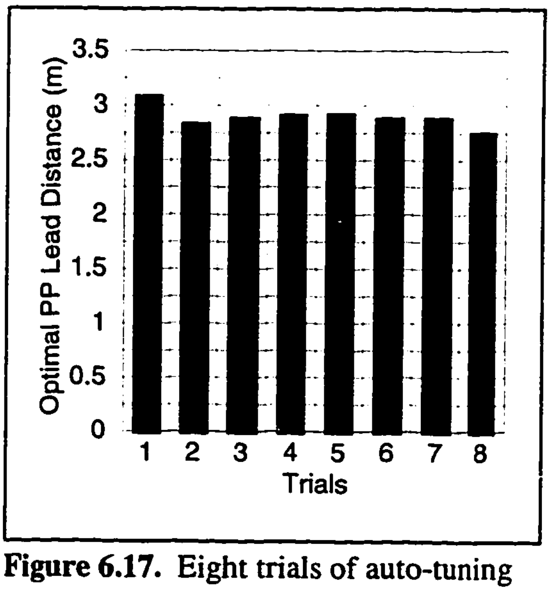

Figure 6.17. Eight trials of auto-tuning the look-ahead distance for a PP controller.

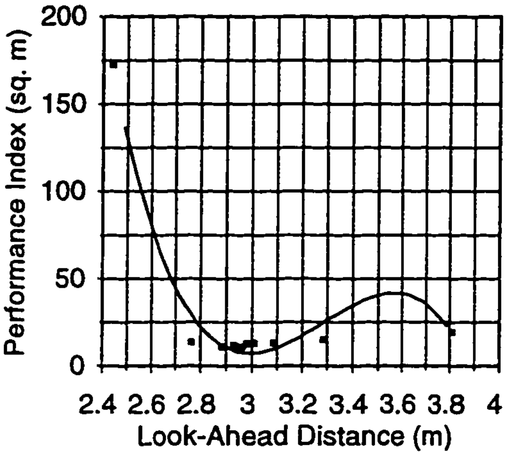

Figure 6.18. Typical data from auto-tuning the look-ahead distance for a PP contoller are fit with a cubic polynomial.

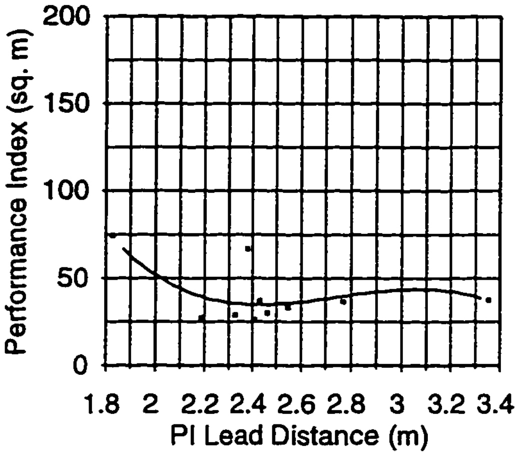

Figure 6.19. Typical data from auto-tuning the look-ahead distance for a PI controller are fit with a cubic polynomial.

4. The procedure was repeated eight times for the pure pursuit controller and five times for the PI controller. The results are shown in Figures 6.16 and 6.17.

The results obtained from applying this procedure to the pure pursuit controller are more repeatable than for the PI controller. Results from typical trials are shown in Figures 6.18 and 6.19. The performance index used to evaluate each run in a trial is the area between the planned path and the actual path traced out by the vehicle. The data in both graphs are fit with a cubic polynomial, using the method of least squares.

For the pure pursuit controller, the optimum look-ahead distance (when traveling at 1.34 m/s) is approximately 3 meters. Notice that there is a region about the optimum where the look-ahead distance doesn't change much. There is a region about the optimum where there is little improvement in path following accuracy.

The PI controller is more sensitive to fluctuations in the vehicle's velocity than the pure pursuit controller. When speed control is not robust, oscillations about the path can be induced (when using PI steering control). This can be seen in Figure 6.19, where there is a stray data point far above other data points. As a result, it is difficult to use this procedure to auto-tune the PI look-ahead distance and obtain repeatable results.

Once the optimal control paramcters were determined using the auto-tuning procedures, the NTV was commanded to follow a path while using these gains. A path was recorded at one meter intervals, having several turns of varying curvature in it. During path following, the average, maximum, and standard deviation from the path, and a performance index was calculated.

The performance index, *area*, is the area between the planned path and the path traced out by the vehicle. The look-ahead distance was changed as a linear function of the vehicle velocity, where λ is the slope of the line. Simulation tests for auto-tuning the pure pursuit controller suggests that λ=0.5 s is appropriate. The graphs that illustrate the path following results are too numerous to include here. Instead, they have been placed in Appendix B.

Figure B.1 illustrates path following (at 1.34 m/s) along the prerecorded path using the manually tuned gains. Figure B.2 shows how the lateral path error and the vehicle velocity changed during the trial. Figures B.3 and B.4 are the results from using the auto-tuned PI gains. The auto-tuned look-ahead distance was increased slightly (to 3 m) to improve stability. Notice that the auto-tuned PI controller performed as well as the manually tuned controller. The average deviations were 0.22 m and 0.21 m, respectively.

Figures B.5 and B.6 show the results from path following (at 1.34 m/s) the preplanned path using the auto-tuned pure pursuit controller. Path following at this speed is extremely accurate when there is no initial lateral path error. Figures B.7 and B.8 shows the results from varying the speed (0-3.75 m/s) during path following when the auto-tuned pure pursuit controller was used. The average deviations were 2 cm and 6 cm, respectively.

Figures B.9 and B.10 illustrate path following when the auto-tuned PI controller is used and there is an initial offset in the vehicle's position. The vehicle's actual path is smooth but convergence to the preplanned path is slow. Figures B.11 and B.12 show how the auto-tuned pure pursuit controller reacts to an initial offset in the vehicle's position. As discovered in the simulation tests, the pure pursuit controller becomes unstable when the lateral error is large (if the look-ahead distance is not increased adaptively and reduced slowly as the vehicle converges to the path).

Figures B.13-B.16 show path following results (at 1.34 m/s) when the weighted solution discussed in the *Weighted PID/Pure pursuit Steering Control* section of chapter 4 was applied. Notice that when there is an initial offset in the vehicle's position (Figures B.13 and B.14), or the vehicle's position and heading (Figures B.15 and B.16), convergence to the path is smooth and fast.

Conclusions

This research involved the development of a strategy that takes advantage of an accurate positioning system, and ensures accurate path following for any steered-wheeled vehicle over a range of speeds (0-4.5 m/s). Accurate path following was defined as an average deviation from the path (due to control error) of less than 0.1 m. This work is based on the technique of decoupling the steering and propulsion control and performing them independently. Here, only steering control was addressed. This section will summarize the author's research and specify what he believes to be unique about it.

The author's path tracking strategy involved standardizing and smoothing a wide range of acceptable paths, implementing several high-level steering controllers (PID, pure pursuit, and weighted PID/pure pursuit), developing tools for evaluating the accuracy of path tracking, generating in real-time a target position for the vehicle to track, and autonomously tuning the control parameters associated with the tested methods. The control strategy was implemented and demonstrated in simulation and on an all-terrain navigation test vehicle.

The path tracking software is flexible in that it is able to receive a variety of path types, each to be explicitly followed. The specified path can be computer generated (by a suite of four path planners implemented by the author), user generated, or prerecorded. The path is then smoothed by an algorithm that fits the path segments with circular arcs (having a radius equivalent to the minimum turning radius associated with the vehicle) or cubic polynomials. The smoothed path is placed in a standardized linked-list form and passed to an executioner program, where a local copy is made.

There exists a wall of separation between the work of the path specifier and the path executioner. The path specifier can be working on one path while the vehicle is tracking a local copy of the previous path. Subsequently, the vehicle can in mid-stream (without stopping) change from following one path to another. A shared memory flag can be set by the user if the path to be executed should be continuously repeated. A path that is currently being tracked can be translated on the fly in order to induce an error during auto-tuning.

The shortest distance from the vehicle's control point to the path (lateral path error) is calculated in real-time and used to update a running total on the average, maximum, and standard deviation from the path. It is also used to calculate the area in real-time between the planned path and the path traced out by the vehicle. Optimization decisions in auto-tuning the control parameters work to decrease this area. A smoothness index and oscillation index are also calculated in real-time, but these parameters were not incorporated into the auto-tuning performance index.

A target position on the path and ahead of the vehicle is also generated in real-time as a "carrot" for the vehicle to follow. The carrot is generated by a so called "backseat driver" program. At any given instant, the only path information the "front-seat driver" program requires is the location of the target position. The user can specify how far in front of the control point the target should be placed, or this parameter can be auto-tuned. This parameter can be changed as a linear function of the vehicle's speed. In principle, the backseat driver could be replaced by a program that uses vision data to generate a target point in the middle of a lane on a street or highway.

When the NTV executes a turn at a specific steering position, the measured curvature is not what is expected from the kinematic considerations. The measured curvature is a function of the commanded steering angle, surface conditions, vehicle speed, tire stiffness, etc. The author has developed an autonomous steering calibration procedure that approximates the relationship between the commanded steering angle and the resulting curvature for constant speed. The commanded steering angle is adjusted to compensate for the established behavior during auto-tuning and path following.

Three high-level steering control algorithms that are responsible for tracking the target position (PID, pure pursuit, and weighted PID/pure pursuit) were implemented and evaluated. The PID controller was difficult to manually tune on the NTV so this controller was degraded to a PI controller. In order to be able to compare the manual and auto-tuned controllers, a PI controller was implemented on the NTV. (An observation made by the author is that the auto-tuned PI controller performed smoother than the auto-tuned PID controller). A PID and PI controller was evaluated in simulation. At high speed, the average deviation from the path associated with the PID controller was significantly smaller.

The (simulation) PID and (NTV) PI gains were auto-tuned using a relay auto-tuner. This tuner, normally applied to industrial plants, oscillates the system (as in the frequency response method) so that the characteristic parameters can be calculated. The Ziegler-Nichols tuning rules are applied to determine the control gains. Here, the vehicle's heading oscillates (12 cycles) about some default value by alternating the command steering angle between a positive and negative value (15°). These controllers are in fact being used to control the vehicle's heading by alternating the steering angle. The amplitude and period of the oscillations are measured and averaged in software.

The manually tuned NTV PI gains (K_p=1.0, K_i=0.18) are close to the auto-tuned gains (K_p=0.8, K_i=0.2). In simulation, the relay auto-tuner was applied at ten speeds between 0-4.5 m/s. The resulting PID gains did not uniformly change. The author found that using the parameters determined at each speed did not improve path following accuracy above using the parameters determined at 1.34 m/s for path following at all speeds.

The look-ahead distance to the real-time target position is the only gain that needs to be tuned for the pure pursuit controller. It is also a gain that needed to be tuned for the PID and PI controllers. A golden section search was performed at several speeds between 0-4.5 m/s in simulation and at 1.34 m/s on the NTV. An educated guess is made on the bounds of the region suspected to contain the optimum gain. Each iteration narrows the search until it is smaller than the user defined tolerance (0.03 m) for stopping the search. The author predefined a search region of about 1.4 meters, which required 10 iterations (20 minutes) to accomplish the search on the NTV.

Auto-tuning the look-ahead distance for the pure pursuit method produced results that were repeatable. This was not the case with auto-tuning the look-ahead distance for the NTV PI controller. This controller appeared to be more susceptible to fluctuations in the vehicle velocity. It was observed that when the vehicle speed drifted above the commanded value, oscillations about the path could be induced.

Using the GSS algorithm to auto-tune a PID or PI controller appears to be useless, unless propulsion control is accurate. (The author used a slightly higher look-ahead distance of 3.0 m to improve stability for path following trials). An attempt was made to smooth the data by fitting it with a cubic polynomial, using the least squares method, but this did not improve repeatability. The lead distance being tuned was also allowed to change as a linear function of the current velocity, but this did not improve repeatability either.

Two simulation programs were written as a part of this research. A nongraphical simulation was used to implement the procedures necessary to test the feasibility of auto-tuning and gain scheduling. A graphical simulation was implemented to demonstrate the path following accuracy of the three high-level controllers that were evaluated. Kinematic equations necessary to update the vehicle and trailer's position and heading were derived, based on the assumptions that the terrain has a constant elevation and there is no wheel slippage. A future application for the graphical simulation is to develop a controller that keeps the trailer on the path during forward and reverse motion (while avoiding a jackknife).

To improve the simulation's representation of reality, the NTV's steering behavior was modeled. It was determined that the current steering angle can be approximated using a trapezoid shaped velocity profile, when the difference between the commanded steering position and the current steering position is small. The steering position is allowed to change by a maximum of 17.5 °/s.

The graphical simulation revealed several advantages and disadvantages of the PID and pure pursuit controllers. The PID and PI controllers are designed to eliminate heading error. As a result, an inherent lateral path error is inevitable around curved sections of the path. The pure pursuit controller can become unstaole when there are large changes in the error (such as approaching the a path from an off-path position).

The PID and PI controllers are stable when the lateral path error is large. (The look-ahead distance is large when the lateral path error is large). The pure pursuit controller provides accurate path following when the vehicle is started on-path. The average control error on the NTV was 2 cm at 1.34 m/s when the vehicle was started on the test path shown in Appendix B.

A weighted PID/pure pursuit solution was tested in an effort to combine the positive features of both controllers. At a constant speed of 1.34 m/s, a lateral error of greater than

3 m uses 100% of the PID (or PI) recommendation. A lateral error of less than 1 m uses 100% of the pure pursuit recommendation. Between these error values, a linear combination of the recommendations are used. The same values developed in simulation were shown to work well on the NTV.

These results indicate that the combination of the PI and pure pursuit controllers in a weighted solution performs better than the independent controllers tested. Further, it is sufficient to use the auto-tuned pure pursuit lead-distance for both controllers (at 1.34 m/s). The weighted solution meets the accuracy criteria set in chapter 1.

There are several aspects of this path tracking strategy that the author believes are unique. The author is not aware of other research that smooths out the rough edges in a planned or user defined path as this work does. There are other implementations that use a carrot following approach to track an explicit path. But none of these methods calculate the carrot in real-time. A static (and usually large) set of subgoals is searched for one that is approximately the look-ahead distance away from the vehicle. This implementation is memory efficient.

A PID controller has previously been applied to the minimize the lateral path error, but not the heading error, for a mobile robot. In simulation, the author has experimented with a new technique of rotating the carrot off-path in an effort to keep a trailer's control point on-path. This is one area that need further exploration.

Auto-tuning the control parameters for an ALV has not received any significant attention in the past. In particular, the application of relay auto-tuning to a mobile plant is new. Gain scheduling was used effectively in simulation for the pure pursuit controller.

Finally, the author has endeavored to quantify the quality of path tracking in real-time, an ingredient missing in other research reviewed by the author.

APPENDIX A
NONGRAPHICAL AUTO-TUNING SIMULATION

Auto-tuning the lead distance is an iterative process. Each row of data represents one lap around a rectangular test path. The optimum lead distance for a given vehicle speed is found by progressively eliminating search areas where it is not expected. The golden section search optimization technique was implemented. The units of the average deviation from the path (*dev*) is in cm. The lead distance used for each iteration is *value*. The optimum lead distance is expected to lie within the *lower* and *upper* limits.

The characteristic parameters for a PID controller, the ultimate gain K_u and the ultimate period T_u, are estimated using relay auto-tuning. The proportional gain K, integral time T_i, and derivative time T_d, are calculated using the Ziegler-Nichols tuning rules.

Pure Pursuit Auto-Tuning

speed = 0.45 m/s iterations = 11

count	value	dev*	lower	upper
1	1.98	203.7	1.98	3.05
2	3.05	22.2	1.98	3.05
3	2.39	13.7	1.98	3.05
4	2.64	15.6	1.98	3.05
5	2.23	13.4	1.98	2.64
6	2.14	13.3	1.98	2.39
7	2.08	14.3	1.98	2.23
8	2.17	13.3	2.08	2.23
9	2.20	13.3	2.14	2.23
10	2.16	14.0	2.14	2.20
11	2.18	13.3	2.16	2.20

lead = 2.17 m avgDev = 13.3 cm

speed = 0.89 m/s iterations = 12

count	value	dev*	lower	upper
1	1.87	204.1	1.87	3.09
2	3.09	23.7	1.87	3.09
3	2.33	13.9	1.87	3.09
4	2.62	15.3	1.87	3.09
5	2.16	14.4	1.87	2.62
6	2.44	14.1	2.16	2.62
7	2.27	13.9	2.16	2.44

8	2.38	14.3	2.27	2.44
9	2.31	13.8	2.27	2.38
10	2.29	13.8	2.27	2.33
11	2.28	13.7	2.27	2.31
12	2.28	13.9	2.27	2.29

lead = 2.28 m avgDev = 13.7 cm

speed = 1.34 m/s iterations = 12

count	value	dev*	lower	upper
1	1.98	191.3	1.98	3.20
2	3.20	26.3	1.98	3.20
3	2.44	14.7	1.98	3.20
4	2.73	15.8	1.98	3.20
5	2.27	13.9	1.98	2.73
6	2.16	15.6	1.98	2.44
7	2.33	14.3	2.16	2.44
8	2.22	15.3	2.16	2.33
9	2.29	14.0	2.22	2.33
10	2.25	14.7	2.22	2.29
11	2.28	13.9	2.25	2.29
12	2.28	13.9	2.27	2.29

lead = 2.28 m avgDev = 13.9 cm

speed = 1.79 m/s iterations = 12

count	value	dev*	lower	upper
1	1.97	176.7	1.97	3.19
2	3.19	21.4	1.97	3.19
3	2.44	15.8	1.97	3.19
4	2.72	16.2	1.97	3.19
5	2.26	180.2	1.97	2.72
6	2.55	20.7	2.26	2.72
7	2.37	17.6	2.26	2.55
8	2.48	15.1	2.37	2.55
9	2.50	15.8	2.44	2.55
10	2.46	15.2	2.44	2.50
11	2.49	15.2	2.46	2.50
12	2.47	15.3	2.46	2.49

lead = 2.48 m avgDev = 15.1 cm

speed = 2.24 m/s iterations = 12

count	value	dev*	lower	upper
1	2.17	200.3	2.17	3.39
2	3.39	23.0	2.17	3.39
3	2.64	18.8	2.17	3.39
4	2.93	18.4	2.17	3.39
5	3.10	18.7	2.64	3.39
6	2.82	18.6	2.64	3.10
7	3.00	18.3	2.82	3.10
8	3.04	18.5	2.93	3.10
9	2.97	17.7	2.93	3.04
10	2.95	18.0	2.93	3.00
11	2.98	18.0	2.95	3.00
12	2.96	17.8	2.95	2.98

lead = 2.97 m avgDev = 17.7 cm

speed = 2.68 m/s iterations = 12

count	value	dev*	lower	upper
1	2.66	25.4	2.66	3.88
2	3.88	27.1	2.66	3.88
3	3.13	19.1	2.66	3.88
4	3.42	20.9	2.66	3.88
5	2.95	18.6	2.66	3.42
6	2.84	20.1	2.66	3.13
7	3.02	18.0	2.84	3.13
8	3.06	18.5	2.95	3.13
9	2.99	17.7	2.95	3.06
10	2.98	18.0	2.95	3.02
11	3.00	17.4	2.98	3.02
12	3.0117.8		2.99	3.02

lead = 3.00 m avgDev=17.4 cm

speed = 3.13 m/s iterations = 12

count	value	dev*	lower	upper
1	2.70	133.1	2.70	3.92
2	3.92	32.5	2.70	3.92
3	3.16	20.6	2.70	3.92
4	3.45	20.6	2.70	3.92
5	2.99	22.9	2.70	3.45
6	3.27	19.9	2.99	3.45
7	3.34	19.5	3.16	3.45
8	3.38	19.6	3.27	3.45
9	3.32	19.5	3.27	3.38
10	3.30	19.7	3.27	3.34
11	3.33	19.7	3.30	3.34
12	3.31	19.8	3.30	3.33

lead = 3.32 m avgDev=19.5 cm

speed = 3.58 m/s iterations = 12

count	value	dev*	lower	upper
1	3.01	33.2	3.01	4.23
2	4.23	29.6	3.01	4.23
3	3.48	21.0	3.01	4.23
4	3.77	23.0	3.01	4.23
5	3.30	30.3	3.01	3.77
6	3.59	21.6	3.30	3.77
7	3.41	19.9	3.30	3.59
8	3.37	21.7	3.30	3.48
9	3.44	19.7	3.37	3.48
10	3.45	19.8	3.41	3.48
11	3.43	19.6	3.41	3.45
12	3.42	19.5	3.41	3.44

lead = 3.42 m avgDev = 19.5 cm

speed = 4.02 m/s iterations = 12

count	value	dev*	lower	upper
1	3.11	152.0	3.11	4.33
2	4.33	30.9	3.11	4.33
3	3.58	101.2	3.11	4.33
4	3.87	25.1	3.11	4.33
5	4.05	25.3	3.58	4.33
6	3.76	22.5	3.58	4.05
7	3.69	22.8	3.58	3.87
8	3.80	23.0	3.69	3.87
9	3.73	60.1	3.69	3.80
10	3.77	23.1	3.73	3.80
11	3.75	22.4	3.73	3.77
12	3.74	60.4	3.73	3.76

lead = 3.75 m avgDev = 22.4 cm

speed = 4.47 m/s iterations = 12

count	value	dev*	lower	upper
1	3.44	160.9	3.44	4.66
2	4.66	46.6	3.44	4.66
3	3.91	22.3	3.44	4.66
4	4.20	26.3	3.44	4.66
5	3.73	21.2	3.44	4.20
6	3.62	27.4	3.44	3.91
7	3.80	24.1	3.62	3.91
8	3.69	79.3	3.62	3.80
9	3.76	25.9	3.69	3.80
10	3.72	26.6	3.69	3.76
11	3.74	61.0	3.72	3.76
12	3.73	25.1	3.72	3.74

lead = 3.73 m avgDev =21.2 cm

PID Controller Auto-Tuning

speed = 0.45 m/s d = 0.075 m

T_u = 8.956 s K_u = 4.443

K = 2.666 T_i = 4.478 T_d = 1.119

iterations = 11

count	value	dev*	lower	upper
1	1.22	17.9	1.22	1.98
2	1.98	15.5	1.22	1.98
3	1.51	15.1	1.22	1.98
4	1.69	16.1	1.22	1.98
5	1.40	13.0	1.22	1.69
6	1.33	12.0	1.22	1.51
7	1.29	12.3	1.22	1.40
8	1.36	11.4	1.29	1.40
9	1.37	12.7	1.33	1.40
10	1.35	12.6	1.33	1.37
11	1.36	12.6	1.35	1.37

lead = 1.36 m avgDev = 11.4 cm

curve fit lead = 1.78 m

speed = 0.89 m/s d =0.104 m

T_u = 5.422 s K_u = 3.193

K = 2.666 T_i = 4.478 T_d = 1.119

iterations = 11

count	value	dev*	lower	upper
1	1.20	19.7	1.20	1.97
2	1.97	20.0	1.20	1.97
3	1.50	17.9	1.20	1.97
4	1.68	20.1	1.20	1.97

5	1.38	16.3	1.20	1.68
6	1.32	16.0	1.20	1.50
7	1.27	15.1	1.20	1.38
8	1.25	15.2	1.20	1.32
9	1.29	15.5	1.25	1.32
10	1.26	15.9	1.25	1.29
11	1.28	15.4	1.26	1.29

lead = 1.27 m avgDev = 15.1 cm

curve fit lead = 1.83 m

speed = 1.34 m/s d = 0.116 m

T_u = 3.82 s K_u = 2.879

K = 2.666 T_i = 4.478 T_d = 1.119

iterations = 11

count	value	dev*	lower	upper
1	1.12	20.3	1.12	1.88
2	1.88	20.8	1.12	1.88
3	1.41	24.2	1.12	1.88
4	1.59	22.1	1.12	1.88
5	1.70	22.4	1.41	1.88
6	1.52	20.7	1.41	1.70
7	1.48	19.3	1.41	1.59
8	1.45	19.2	1.41	1.52
9	1.44	20.2	1.41	1.48
10	1.46	19.2	1.44	1.48
11	1.45	19.8	1.44	1.46

lead =1.45 m avgDev = 19.2 cm

curve fit lead = 1.72 m

speed = 1.79 m/s d = 0.144 m

T_u=3.422 s K_u = 2.319923

K = 2.666 T_i = 4.478 T_d = 1.119

iterations = 11

count	value	dev*	lower	upper
1	1.30	20.9	1.30	2.06
2	2.06	28.7	1.30	2.06
3	1.59	23.1	1.30	2.06
4	1.77	23.0	1.30	2.06
5	1.88	20.6	1.59	2.06
6	1.95	26.3	1.77	2.06
7	1.84	26.6	1.77	1.95
8	1.91	26.8	1.84	1.95
9	1.8	33.2	1.84	1.91
10	1.89	29.9	1.87	1.91
11	1.88	26.1	1.87	1.89

lead = 1.88 m avgDev = 20.6 cm

curve fit lead = 1.94 m

speed = 2.24 m/s d = 0.153 m

T_u = 2.822 s K_u = 2.182

K = 2.666 T_i = 4.478 T_d = 1.119

iterations = 11

count	value	dev*	lower	upper
1	1.73	25.1	1.73	2.49
2	2.49	30.3	1.73	2.49
3	2.02	51.8	1.73	2.49
4	2.20	35.8	1.73	2.49
5	2.31	34.6	2.02	2.49
6	2.38	36.1	2.20	2.49
7	2.27	36.6	2.20	2.38
8	2.34	38.1	2.27	2.38
9	2.30	38.9	2.27	2.34
10	2.32	36.5	2.30	2.34
11	2.31	39.9	2.30	2.32

lead = 1.73 m avgDev = 25.1 cm

curve fit lead = 2.37 m

speed = 2.68 m/s d = 0.162 m

T_u = 2.422 s K_u = 2.052

K = 2.666 T_i = 4.478 T_d = 1.119

iterations = 11

count	value	dev*	lower	upper
1	1.58	31.6	1.58	2.34
2	2.34	27.7	1.58	2.34
3	1.87	43.7	1.58	2.34
4	2.05	78.2	1.58	2.34
5	1.76	33.6	1.58	2.05
6	1.69	42.7	1.58	1.87
7	1.80	55.2	1.69	1.87
8	1.73	41.0	1.69	1.80
9	1.78	62.9	1.73	1.80
10	1.75	27.9	1.73	1.78
11	1.74	30.1	1.73	1.76

lead = 2.34 m avgDev = 27.7 cm

curve fit lead = 1.74 m

speed = 3.13 m/s d = 0.193 m
T_u = 2.422 s K_u = 1.723
K = 2.666 T_i = 4.478 T_d = 1.119
iterations = 11

count	value	dev*	lower	upper
1	2.19	71.6	2.19	2.95
2	2.95	44.2	2.19	2.95
3	2.48	52.7	2.19	2.95
4	2.66	60.8	2.19	2.95
5	2.37	44.6	2.19	2.66
6	2.30	45.9	2.19	2.48
7	2.41	66.5	2.30	2.48
8	2.34	41.1	2.30	2.41
9	2.33	69.0	2.30	2.37
10	2.35	51.1	2.33	2.37
11	2.34	69.3	2.33	2.35

lead = 2.34 m avgDev = 41.1 cm
curve fit lead = 2.39 m

speed = 3.58 m/s d = 0.223 m
T_u = 2.422 s K_u = 1.498
K = 2.666 T_i = 4.478 T_d = 1.119
iterations = 11

count	value	dev*	lower	upper
1	2.19	45.9	2.19	2.95
2	2.95	47.0	2.19	2.5
3	2.48	87.6	2.19	2.95
4	2.66	110.2	2.19	2.95
5	2.37	104.4	2.19	2.66
6	2.55	61.0	2.37	2.66
7	2.59	40.3	2.48	2.66
8	2.62	78.7	2.55	2.66
9	2.58	99.6	2.55	2.62
10	2.60	71.5	2.58	2.62
11	2.59	84.5	2.58	2.60

lead =2.59 m avgDev = 40.3 cm
curve fit lead = 2.80 m

speed = 4.02 m/s d = 0.250 m
T_u = 2.422 s K_u = 1.335
K = 2.666 T_i = 4.478 T_d = 1.119
iterations = 11

count	value	dev	lower	upper
1	2.44	55.1	2.44	3.20
2	3.20	59.2	2.44	3.20
3	2.73	76.6	2.44	3.20
4	2.91	104.8	2.44	3.20
5	2.62	103.9	2.44	2.91
6	2.80	42.6	2.62	2.91
7	2.84	77.9	2.73	2.91
8	2.77	87.4	2.73	2.84
9	2.82	76.7	2.77	2.84
10	2.79	92.5	2.77	2.82
11	2.81	104.9	2.79	2.82

lead = 2.80 m avgDev = 42.6 cm
curve fit lead = 3.04 m

speed = 4.47 m/s d = 0.308 m
T_u = 2.600 s K_u = 1.082
K = 2.666 T_i = 4.478 T_d = 1.119
iterations = 11

count	value	dev	lower	upper
1	2.65	112.6	2.65	3.41
2	3.41	55.1	2.65	3.41
3	2.94	61.6	2.65	3.41
4	3.12	66.3	2.65	3.41
5	2.83	48.9	2.65	3.12
6	2.76	117.0	2.65	2.94
7	2.87	115.5	2.76	2.94
8	2.80	70.5	2.76	2.87
9	2.84	42.3	2.80	2.87
10	2.85	82.2	2.83	2.87
11	2.84	56.1	2.83	2.85

lead = 2.84 m avgDev = 42.3 cm
curve fit lead = 3.24 m

* The units of average deviation (*dev*) is cm.

APPENDIX B
NTV PATH FOLLOWING RESULTS

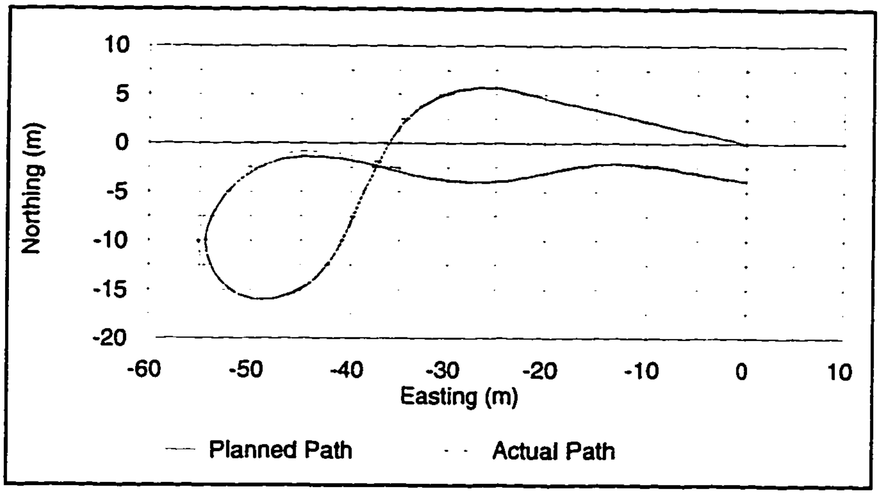

Figure B.1. Path following using PI control and manually tuned gains (K_p=1.0, K_i=0.18, L=3.24 m) at 1.34 m/s, with no initial offset and λ=0.5.

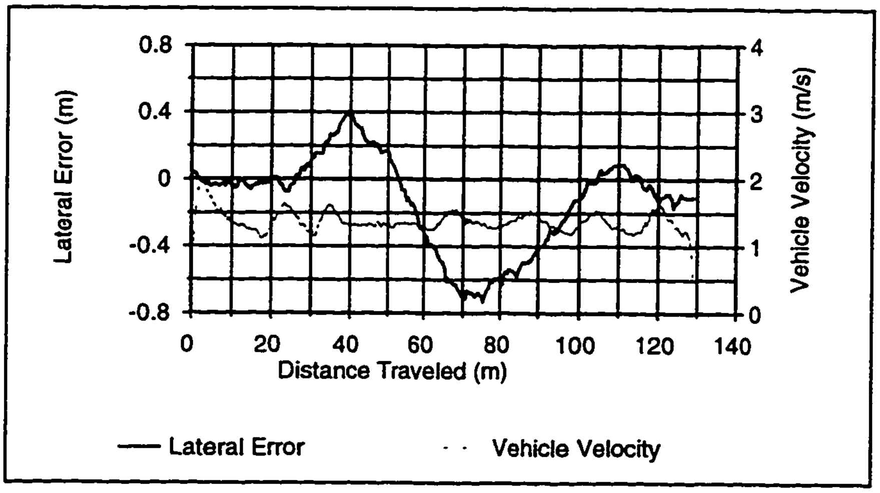

Figure B.2. The lateral error and vehicle velocity during the test in Figure B.1. (Avg dev=0.22 m, max dev=0.74 m, std dev=0.22, area=29.5 m^2).

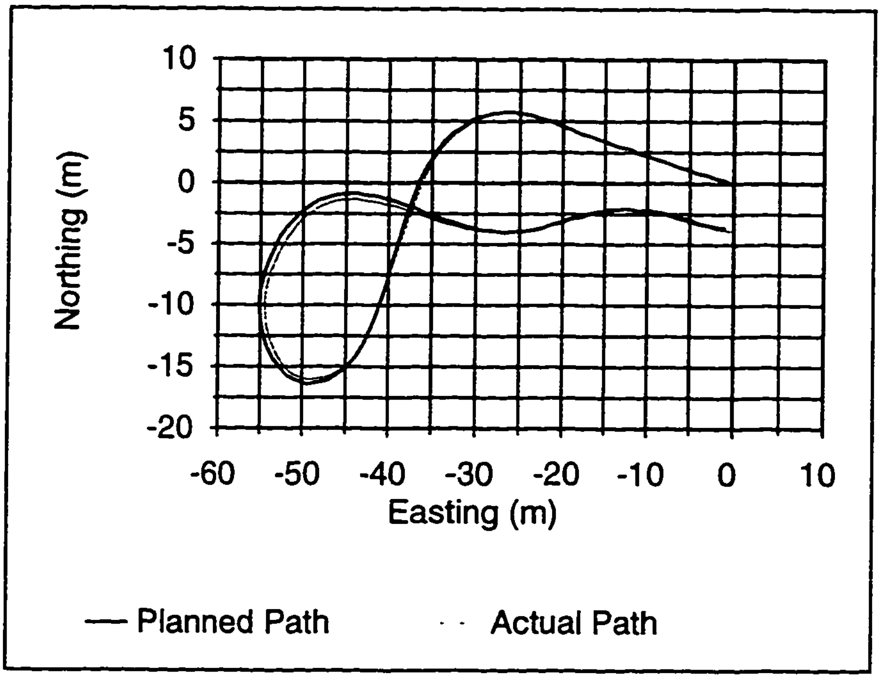

Figure B.3. Path following using PI control and auto-tuned gains (K_p=0.82, K_i=0.21, L=3 m) at 1.34 m/s, with no initial offset and λ=0.5.

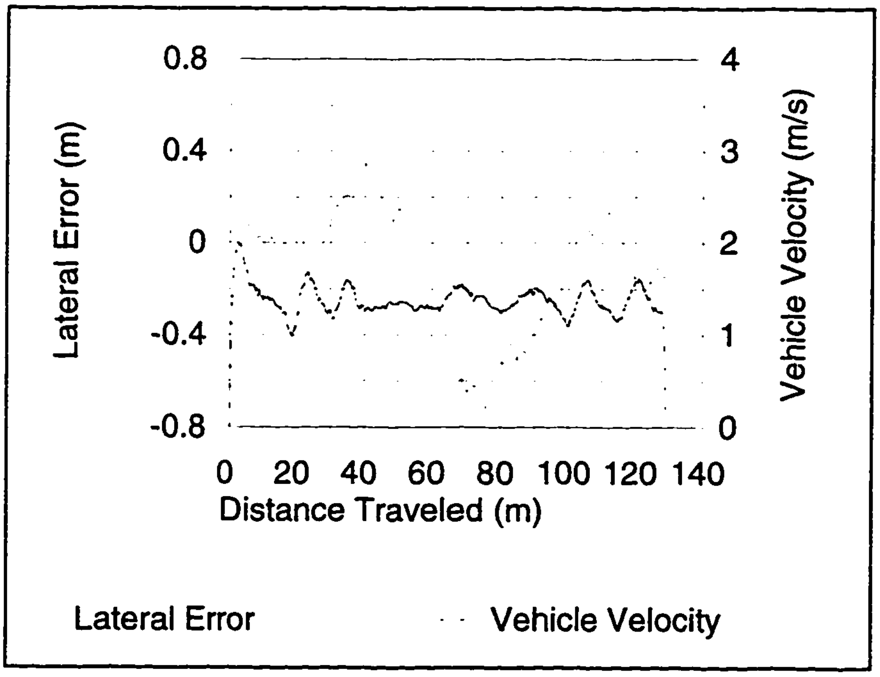

Figure B.4. The lateral error and vehicle velocity during the test in Figure B.3. (Avg dev=0.21 m, max dev=0.73 m, std dev=0.19, area=27.6 m^2).

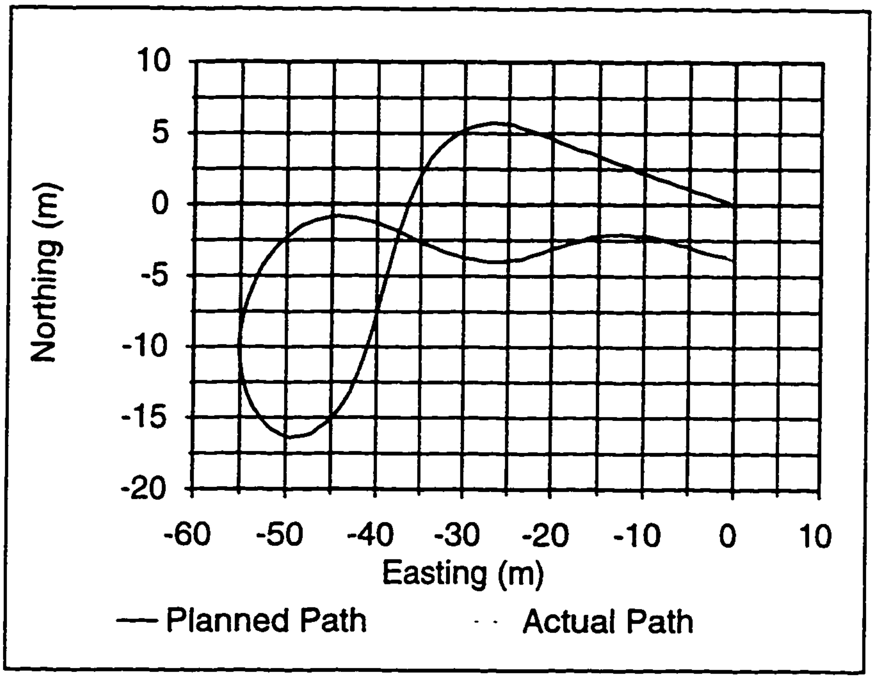

Figure B.5. Path following using pure pursuit control and an auto-tuned gain (L=3 m) at 1.34 m/s, with no initial offset and λ=0.5.

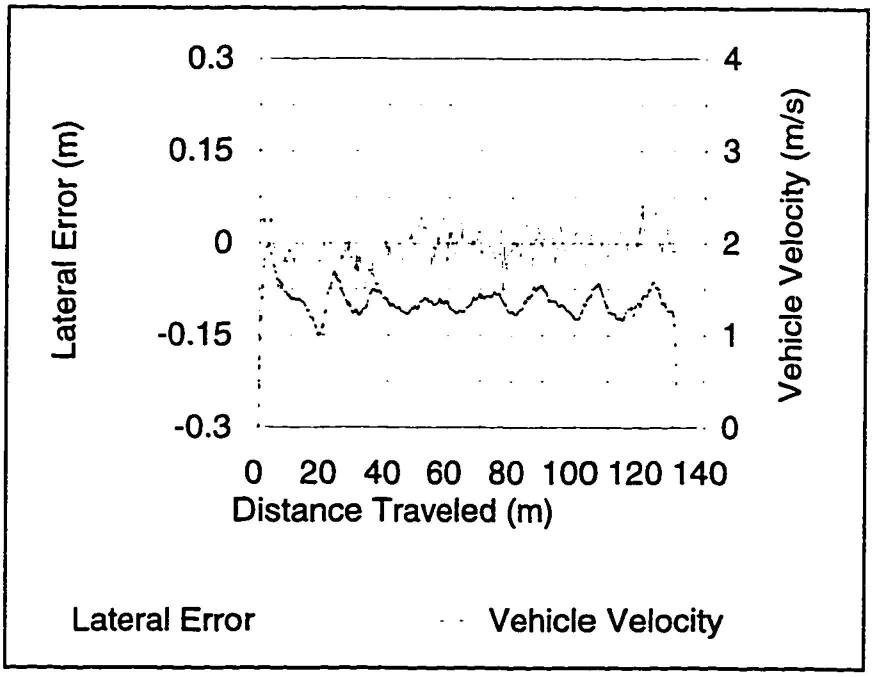

Figure B.6. The lateral error and vehicle velocity during the test in Figure B.5. (Avg dev=0.02 m, max dev=0.08 m, std dev=0.02, area=2.66 m^2).

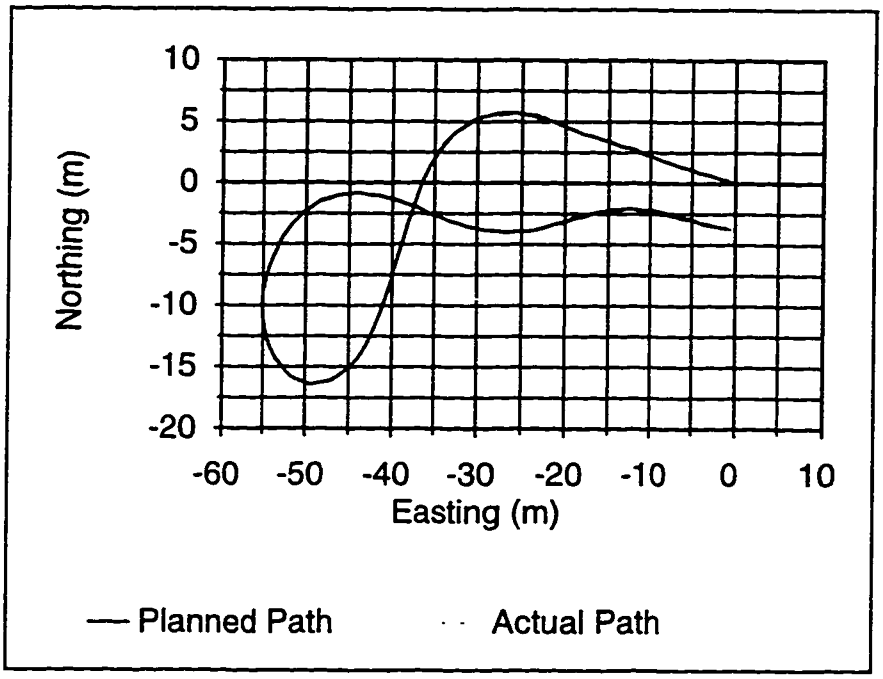

Figure B.7. Path following using pure pursuit control and an auto-tuned gain (L=3 m) at varied speed, with no initial offset and λ=0.5.

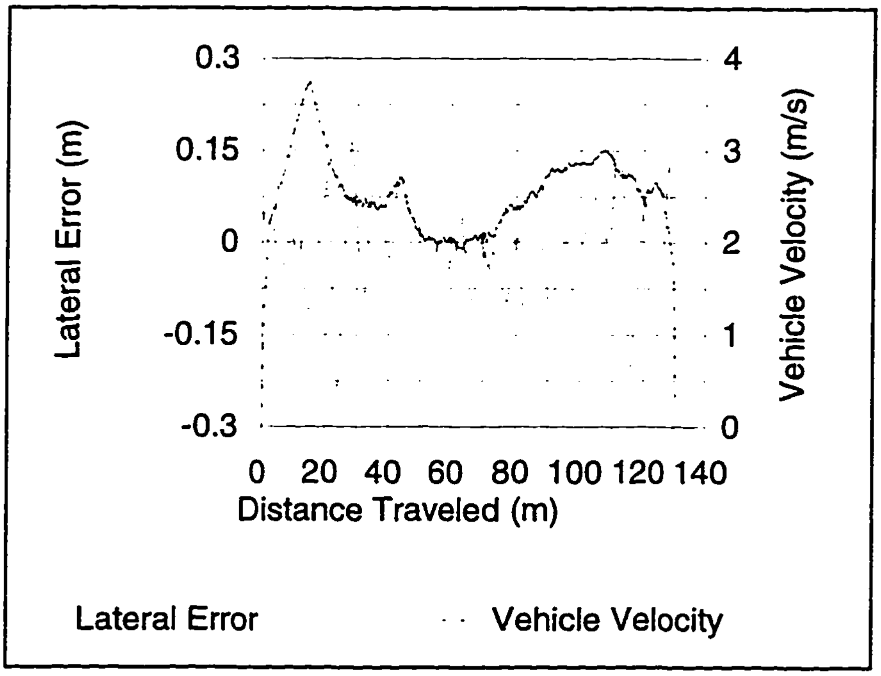

Figure B.8. The lateral error and vehicle velocity during the test in Figure B.7. (Avg dev=0.06 m, max dev=0.25 m, std dev=0.05, area=7.4 m^2).

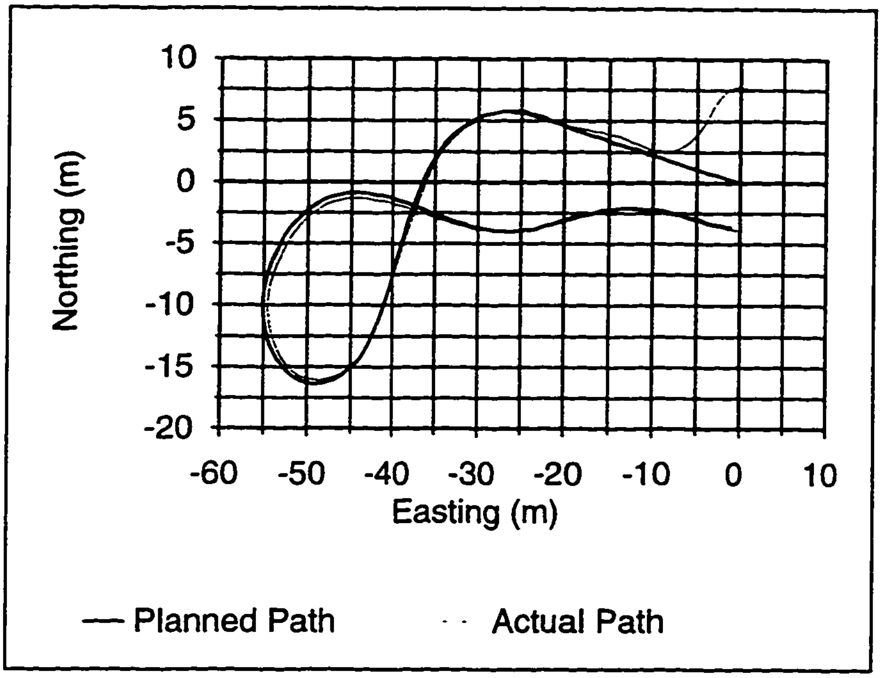

Figure B.9. Path following using PI control and auto-tuned gains (K_p=0.82, K_i=0.21, L=3 m) at 1.34 m/s, with an initial offset and λ=0.5.

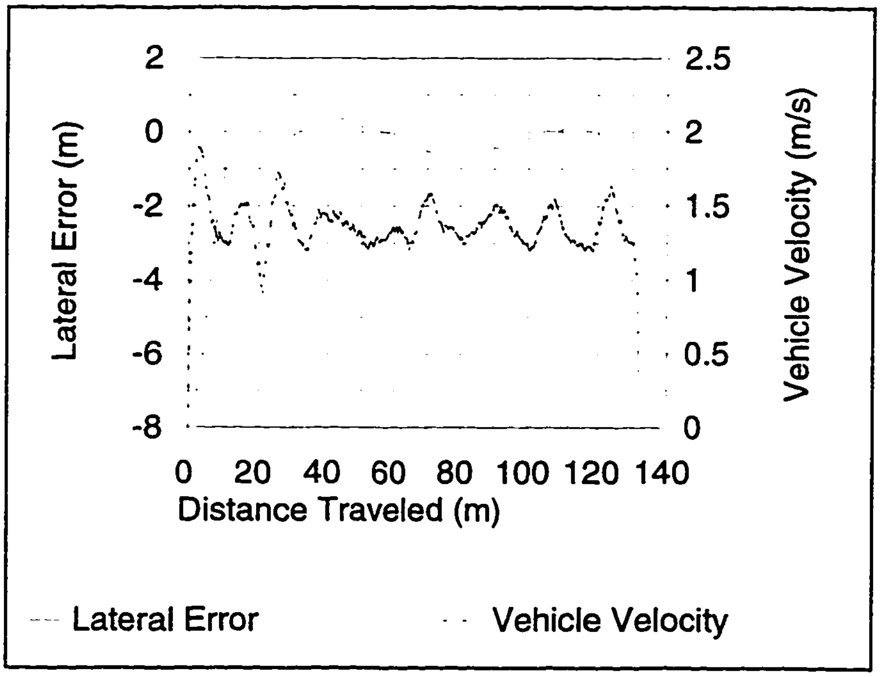

Figure B.10. The lateral error and vehicle velocity during the test in Figure B.9. (Avg dev=0.63 m, max dev=7.67 m, std dev=1.45, area=65.2 m^2).

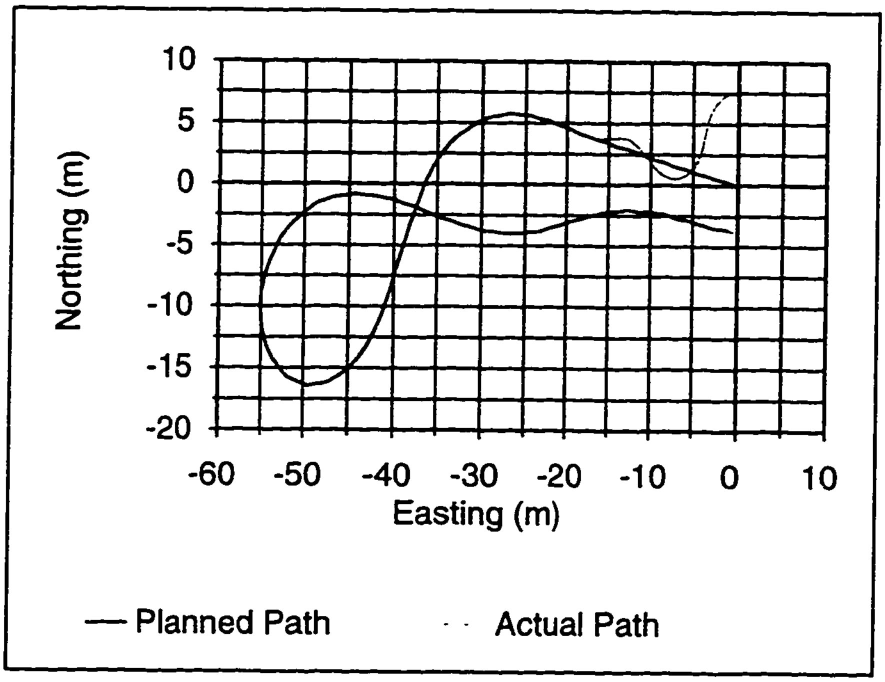

Figure B.11. Path following using pure pursuit control and an auto-tuned gain (L=3 m) at 1.34 m/s, with an initial offset and λ=0.5.

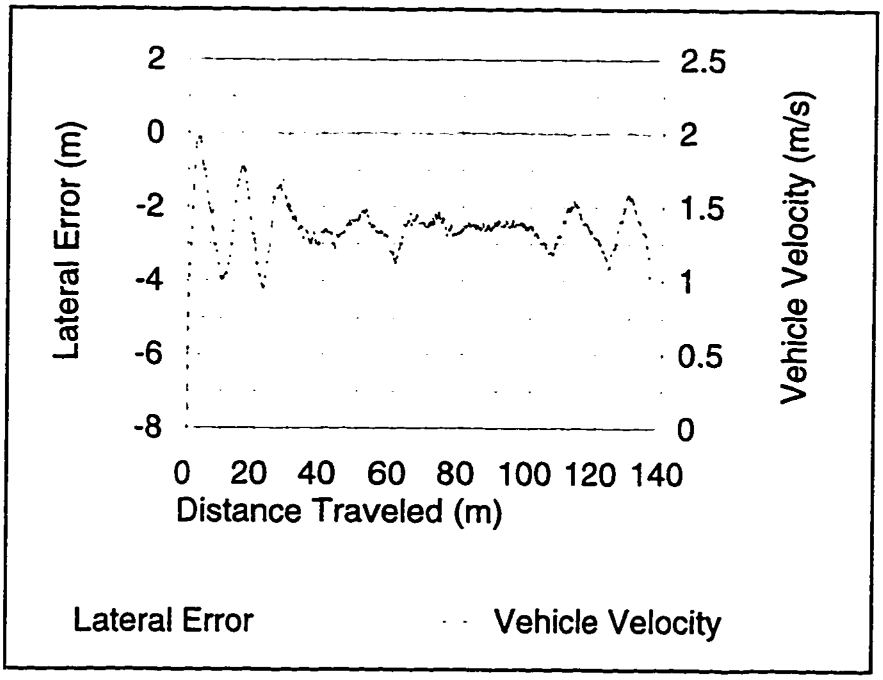

Figure B.12. The lateral error and vehicle velocity during the test in Figure B.11. (Avg dev=0.43 m, max dev=7.51 m, std dev=1.22, area=35.0 m^2).

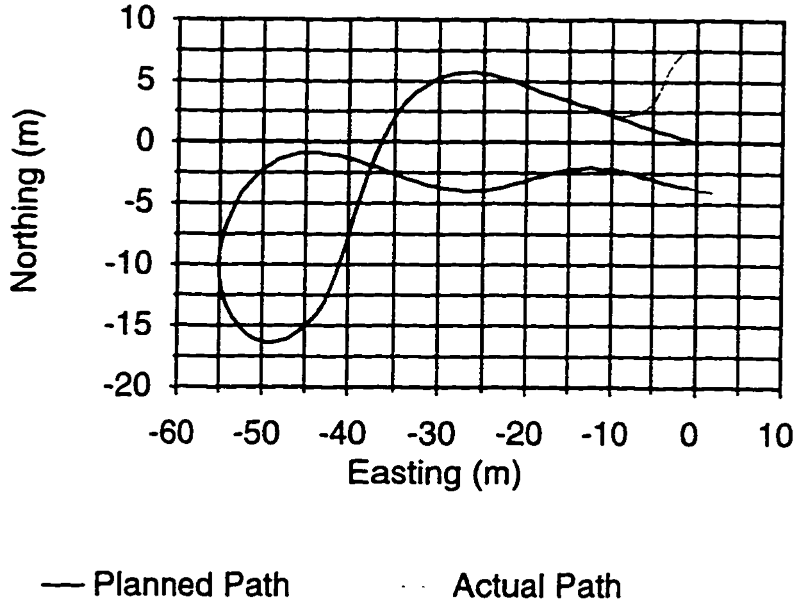

Figure B.13. Path following using weighted PI/PP control and auto-tuned gains (K_p=0.82, K_i=0.21, L=3 m) at 1.34 m/s, with an initial offset and λ=0.5.

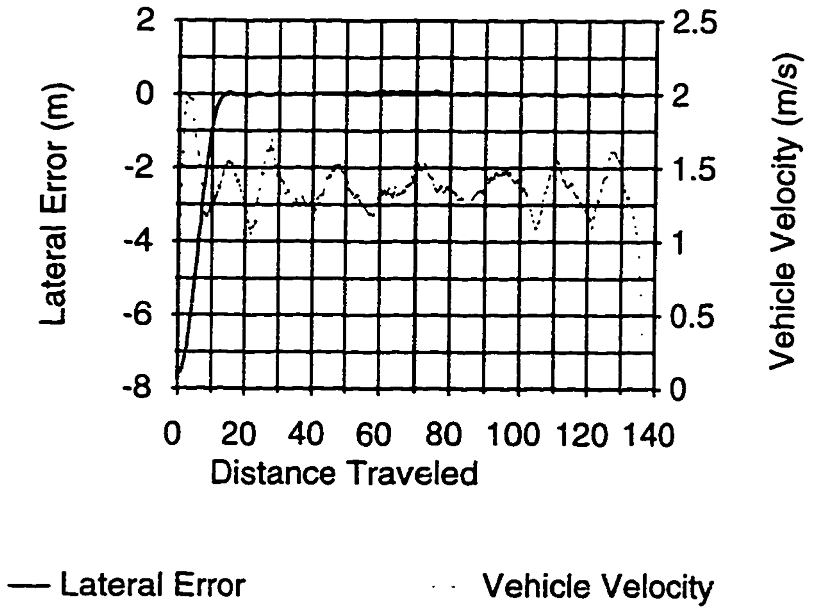

Figure B.14. The lateral error and vehicle velocity during the test in Figure B.13. (Avg dev=0.41 m, max dev=7.59, std dev=1.45, area=35.43 m^2).

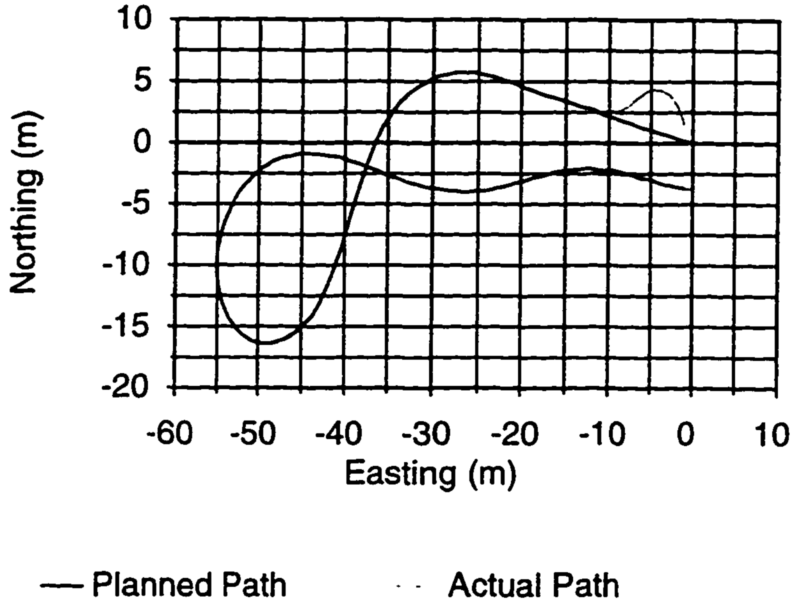

Figure B.15. Path following using weighted PI/PP control and auto-tuned gains (K_p=0.82, K_i=0.21, L=3 m) at 1.34 m/s, with an initial offset and λ=0.5.

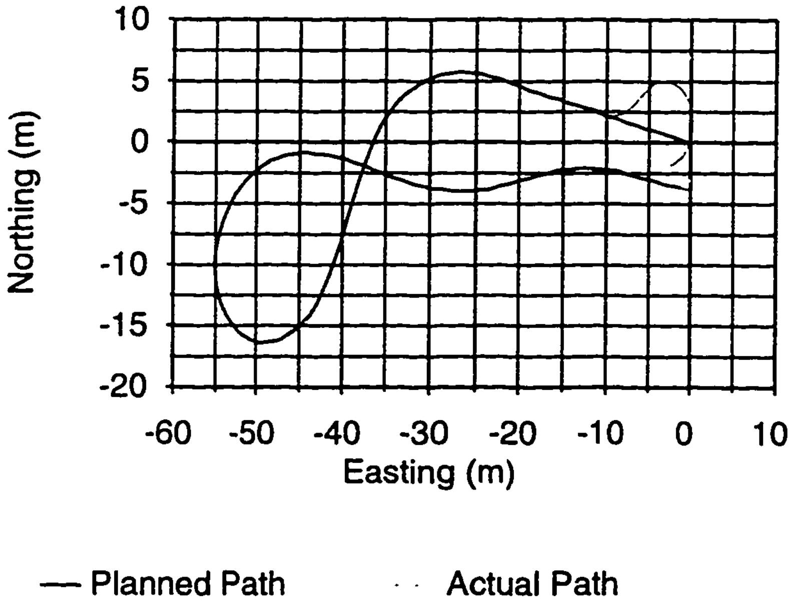

Figure B.16. Path following using weighted PI/PP control and auto-tuned gains (K_p=0.82, K_i=0.21, L=3 m) at 1.34 m/s, with an initial offset and λ=0.5.

REFERENCE LIST

Ack94 Ackermann, J., Guldner, J., and Utkin, V.I., “A Robust Nonlinear Control Approach to Automatic Path Tracking of a Car,” Proceedings of the IEEE International Conference on Control, 1, 389(1994), 196-201.

Ale88 Alexander, J.C. and Maddocks, J.H., "On the Maneuvering of Vehicles," SIAM Journal of Applied Mathematics 48, 1(1988), 38-51.

Ale89 Alexander, J.C. and Maddocks, J.H., "On the Kinematics of Wheeled Mobile Robots," International Journal of Robotics Research 8, 5(1989), 15-27.

Ami90 Amidi, O. and Thorpe, C., “Integrated Mobile Robot Control,” Proceedings of SPIE, 1388, (1990), 505-523.

Ast95 Astrom, K. and Hagglund, T., PID Controllers: Theory, Design, and Tuning, Instrument Society of America, Research Triangle Park, NC, 1995.

Bou92 Boukas, E.K. and Kenne, J.P., “Autonomous Vehicle: A Comparative Study of Control Algorithms,” Engineering Systems, Design, and Analysis, PD-Vol. 47-4, (1992), 183-187.

Can91 Canudas de Wit, C. and Sordalen, O.J., "Exponential Stabilization of Mobile Robots with Nonholonomic Constraints," IEEE Conference on Decision and Control, Brighton, England (1991), 692-703.

Che87 Chen, C., “Navigation of an Autonomous Land Vehicle,” Proceedings of the International Conference on Robotics and Automation, Raleigh, NC (1987), 140-144.

Cox90 Cox, I.J. and Wilfong, G.T., eds., Autonomous Robot Vehicles, Springer-Verlag, New York, 1990.

Cra95 Crane, C.D., Rankin, A.L., Armstrong, D.G., Wit, J.S., and Novick, D.K., “An Evaluation of INS and GPS for Autonomous Navigation,” Proceedings of the 2nd IFAC Conference on Intelligent Autonomous Vehicles, Espoo, Finland (1995), 208-213.

dAn91 d'Andrea-Novel, B., Bastin, G., and Campion, G., "Modelling and Control of Non-Holonomic Wheeled Mobile Robots," Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA (1991), 1130-1135.

dAn92 d'Andrea-Novel, B., Bastin, G., and Campion, G., "Dynamic Feedback Linearization of Nonholonomic Wheeled Mobile Robots," Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France (1992), 2527-2532.

DeS93 DeSantis, R.M., "Path-Tracking for a Car-like Mobile Robot," Proceedings of the American Control Conference, San Francisco, CA (1993), 64-68.

Dep58 Department of the Army, "Universal Transverse Mercator Grid," Technical Manual TM5-241-8, National Technical Information Center, Springfield, VA, Document #ADA176624 (1958).

Fen90 Feng, D. and Krogh, B.H., "Dynamic Steering Control of Conventionally-Steered Mobile Robots," Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH (1990), 390-395.

Fen94 Feng, L., Borenstein, J., and Everett, H.R., "Where Am I? Sensors and Methods for Mobile Robot Positioning," Technical Report UM-MEAM-94-21, University of Michigan, 1994.

Fre95 French, P.W., "Automation of an All-Terrain Tow Vehicle," Master's Thesis, University of Florida, 1995.

Gri90 Griswold, N.C., "Control for Mobile Robots in the Presence of Moving Obstacles," IEEE Transactions on Robotics and Automation, 6, 2(1990), 263-268.

Hed91 Hedrick, J.K., McMahon, D., Narendran, V., and Swaroop, D., "Longitudingal Vehicle Controller Design for IVHS Systems," Proceedings of the American Control Conference, 3, (1991), 3107-3112.

Hes91 Hessburg, T., Peng, H., and Tomizuka, M., "An Experimental Study on Lateral Control of a Vehicle," Proceedings of the American Control Conference, 3, (1991), 3084-3089.

Hom91 Hommertzheim, D., Huffman, J., and Sabuncuoglu, I., "Training an Artificial Neural Network the Pure Pursuit Method," Computer Operations Research, 18, 4(1991), 343-353.

Kan88 Kanayama, Y., Nilipour, A., and Lelm, C.A., "A Locomotion Control Method for Autonomous Vehicles," Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA (1988), 1315-1317.

Kan90 Kanayama, Y., Kimura, Y., Miyazaki, F., and Noguchi, T., "A Stable Tracking Control Method for an Autonomous Mobile Robot," Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH (1990), 384-389.

Keh91 Kehtarnavaz, N. And Sohn, W., "Steering Control of Autonomous Vehicles by Neural Networks," Proceedings of the American Control Conference, 3, (1991), 3096-3101.

Kel94a Kelly, A., "A Feedforward Control Approach to the Local Navigation Problem for Autonomous Vehicles," Technical Report CMU-RI-TR-94-17, Carnegie Mellon University, 1994.

Kel94b Kelly, A., "An Intelligent Predictive Controller for Autonomous Vehicles," Technical Report CMU-RI-TR-94-20, Carnegie Mellon University, 1994.

Kro85 Krogh, B.H., "Guaranteed Steering Control," Proceedings of the American Control Conference, Boston, MA (1985), 950-955.

Kro86 Krogh, B.H. and Thorpe, C.E., "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles," Proceeding of the IEEE International Conference on Robotics and Automation, San Francisco, CA (1986), 1664-1669.

Laf91 Lafferiere, G. and Sussmann, H.J., "Motion Planning for Controllable Systems without Drift," Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA (1991), 1148-1153.

Lat90 Latombe, J., Robot Motion Planning, Kluwer Academic Publishers, Boston, 1990.

Lau86 Laumond, J., "Feasible Trajectories for Mobile Robots with Kinematic and Environment Constraints," Proceedings of the International Conference on Intelligent Systems, Amsterdam (1986), 346-354.

Mui87 Muir, P.F. and Neuman, C.P., "Kinematic Modeling of Wheeled Mobile Robots," Journal of Robotic Systems, 4, 2(1987), 281-340.

Mul92 Muller, N., "Feedforward Control for Curve Steering for an Autonomous Road Vehicle," Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France (1992), 200-205.

Mur90 Murray, R.M. and Sastry, S.S., "Steering Nonholonomic Systems Using Sinusoids," IEEE Conference on Decision and Control, Honolulu (1990), 2097-2101.

Mur94 Murphy, K.N., "Analysis of Robotic Vehicle Steering and Controller Delay," In: Robotics and Manufacturing: Recent Trends in Research, Education, and Applications, ASME Press, New York, 1994, 631-636.

Nel88 Nelson W.L. and Cox, I., "Local Path Control for an Autonomous Vehicle," Proceedings of the IEEE International Conference on Robotics and Automation, Philadelphia, PA (1988), 1504-1510.

Nel89 Nelson W., "Continuous Curvature Paths for Autonomous Vehicles," Proceedings of the IEEE International Conference on Robotics and Automation, Scottsdale, AZ (1989), 1260-1264.

Oll91 Ollero, A. and Amidi, O., "Predictive Path Tracking of Mobile Robots," Proceeding of the 5th International Conference on Advanced Robotics, 2, (1991), 1081-1086.

Oll94 Ollero, A., Garcia-Cerezo, A., and Martinez, J.L., "Fuzzy Supervisory Path Tracking of Mobile Robots," Control Engineering Practice, 2, 2(1994), 313-319.

Oll95 Ollero, A. and Heredia, G., "Stability Analysis of Mobile Robot Path Tracking," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 3, (1995), 461-466.

Pay91 Payton, D.W. and Bihari, T.E., "Intelligent Real-Time Control of Robotic Vehicles," Communications of the ACM, 34, 8(1991), 48-63.

Pen91 Peng, H. and Tomizuka, M., "Preview Control for Vehicle Lateral Guidence in Highway Automation," Proceedings of the American Control Conference, 3, (1991), 3090-3095.

Ran96a Rankin, A.L., Crane, C.D., Armstrong, D.G., Brown, H.E., and Nease, A.D., "Autonomous Path Planning Navigation System Used for Site

Characterization", In: Navigation and Control Technologies for Unmanned Systems, S.A. Speigle, Ed., Proceedings of SPIE, 2738, (1996), 176-186.

Ran96b Rankin, A.L., Crane, C.D., and Duffy, J., "Simulating the Motion of a Nonholonomic Robot and its Trailer", In: Recent Advances In Robot Kinematics, J. Lenarcic and V. Parenti-Castelli, eds., Kluwer Academic Publishers, Boston, (1996), 277-286.

Ran96c Rankin, A.L. and Crane, C.D., "A Multi-purpose Off-line Path Planner Based on an A* Search Algorithm", Proceedings of the ASME Design Engineering Technical Conferences, Irvine, CA, August 1996.

Rat87 Rathbone, R.R., Valley, R.Jr., and Kindlmann, P.J., "Guided Vehicle Path Specification: Well Behaved Dynamics with Economy of Representation," Proceedings of the IEEE International Symposium on Intelligent Control, Philadelphia, PA (1987), 129-134.

Sam91 Samson, C. and Ait-Abderrahim, K., "Feedback Control of a Nonholonomic Wheeled Cart in Cartesian Space," Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA (1991), 1136-1141.

She91 Sheikholeslam, S. and Desoer, C.A., "Longitudinal Control of a Platoon of Vehicles with no Communication of the Lead Vehicle Information," Proceedings of the American Control Conference, 3, (1991), 3102-3106.

Shi90 Shin, D.H. and Singh, S., "Path Generation for Robot Vehicles Using Composite Clothoid Segments," Technical Report CMU-RI-TR-90-31, Carnegie Mellon University, 1990.

Shi91 Shin, D.H., Singh, S., and Shi, W., "A Partitioned Control Scheme for Mobile Robot Path Tracking," Proceedings of the IEEE Conference on Systems Engineering, Fairborn, OH (1991), 338-342.

Shi92 Shin, D.H., Sanjiv, S., and Lee, J.J., "Explicit Path Tracking by Autonomous Vehicles," Robotica, 10, (1992), 539-554.

Shi95 Shin, D.H. and Ollero, A., "Mobile Robot Path Planning for Fine-Grained and Smoothed Path Specifications," Journal of Robotic Systems, 12, 2(1995), 491-503.

Sin91 Singh, S., Feng, D., Keller, P., Shaffer, G., Shi, W.F., Shin, D.H., West, J., and Wu, B.X., "A System for Fast Navigation of Autonomous Vehicles," Technical Report CMU-RI-TR-91-20, Carnegie Mellon University, 1991.

Smi91 Smith, D.E. and Starkey, J.M., "Overview of Vehicle Models, Dynamics, and Control Applied to Automated Vehicles," Advanced Automotive Technologies, DE-Vol. 40, (1991), 69-87.

Sor92 Sordalen, O.J. and Canudas de Wit, C., "Exponential Control Law for a Mobile Robot: Extension to Path Following," Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France (1992), 2158-2163.

Tak90 Takano, M., "Mechanism of an Autonomous Mobile Robot and its Control," Advanced Robotics, 4, 3(1990), 187-202.

Van94 Vandoren, V. J., "The Challenges of Self-Tuning Control," Control Engineering, February (1994), 77-79.

van92 van Turennout, P., Hondered, G., and van Schelvin, L.L., "Wall-following Control of a Mobile Robot," Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France (1992), 280-285.

Wil90 Wilfong, G.T., "Motion Planning for an Autonomous Vehicle," Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH (1990), 529-533.

Zha92 Zhao, Y. and BeMent, S.L., "Kinematics, Dynamics and Control of Wheeled Mobile Robots," Proceedings of the IEEE International Conference on Robotics and Automation, Nice, France (1992), 91-96.

Zie42 Ziegler, J.G. and Nichols, N.B., "Optimum Settings for Automatic Controllers," Transactions of the AMSE, 64, (1942), 759-768.

BIOGRAPHICAL SKETCH

Arturo Lionel Rankin was born January 10, 1963, in Takoma Park, Maryland. He was awarded the degree of Bachelor of Science in Engineering, with an emphasis in biomedical applications, from the Catholic University of America, Washington, D.C., in May 1987. In August 1988, he began graduate studies at the University of Florida.

In December 1990, he was awarded the degree of Master of Science in food science and human nutrition. He has since returned to the field of engineering and in August, 1993, he was awarded the degree of Master of Science in mechanical engineering. He intends to work in industry upon completion of a doctoral degree in mechanical engineering.

Arturo has four siblings and his wife, Donna Marie Samuels Rankin, is an entrepreneur. In 1987, he was awarded the C.C. Chang award, presented each year to a graduating senior in biomedical engineering. He was recognized by Who's Who Among Students in American Universities and Colleges in 1986 and is a lifetime member of Tau Beta Pi. He is also a member of the worldwide Seventh-Day Adventist Church.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Carl D. Crane III, Chairman
Associate Professor of Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Joseph Duffy
Graduate Research Professor of Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Gary K Matthew

Gary Matthew
Associate Professor of Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Paul Mason
Assistant Professor of Mechanical Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Sencer Yeralan

Sencer Yeralan
Associate Professor of Industrial and Systems
Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 1997

Winfred M. Phillips
Dean, College of Engineering

Richard

Karen A. Holbrook
Dean, Graduate School