GROUND VEHICLE PLATOONING CONTROL AND SENSING IN AN ADVERSARIAL ENVIRONMENT

by

Samuel A. Mitchell

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Engineering

Approved:

Ryan Gerdes, Ph.D. Major Professor	Rajnikant Sharma, Ph.D. Committee Member
Tam Chantem, Ph.D. Committee Member	Mark R. McLellan, Ph.D. Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2016

ABSTRACT

Ground Vehicle Platooning Control and Sensing
in an Adversarial Environment

by

Samuel A. Mitchell, Master of Science
Utah State University, 2016

Major Professor: Ryan Gerdes, Ph.D.
Department: Electrical and Computer Engineering

The highways of the world are growing more congested. People are inherently bad drivers from a safety and system reliability perspective. Self-driving cars are one solution to this problem, as automation can remove human error and react consistently to unexpected events. Automated vehicles have been touted as a potential solution to improving highway utilization and increasing the safety of people on the roads. Automated vehicles have proven to be capable of interacting safely with human drivers, but the technology is still new. This means that there are points of failure that have not been discovered yet.

The focus of this work is to provide a platform to evaluate the security and reliability of automated ground vehicles in an adversarial environment. An existing system was already in place, but it was limited to longitudinal control, relying on a steel cable to keep the vehicle on track. The upgraded platform was developed with computer vision to drive the vehicle around a track in order to facilitate an extended attack. Sensing and control methods for the platform are proposed to provide a baseline for the experimental platform.

Vehicle control depends on extensive sensor systems to determine the vehicle position relative to its surroundings. A potential attack on a vehicle could be performed by jamming the sensors necessary to reliably control the vehicle. A method to extend the sensing utility

of a camera is proposed as a countermeasure against a sensor jamming attack. A monocular camera can be used to determine the bearing to a target, and this work extends the sensor capabilities to estimate the distance to the target. This provides a redundant sensor if the standard distance sensor of a vehicle is compromised by a malicious agent. For a 320×200 pixel camera, the distance estimation is accurate between 0.5 and 3 m.

One previously discovered vulnerability of automated highway systems is that vehicles can coordinate an attack to induce traffic jams and collisions. The effects of this attack on a vehicle system with mixed human and automated vehicles are analyzed. The insertion of human drivers into the system stabilizes the traffic jam at the cost of highway utilization.

(115 pages)

PUBLIC ABSTRACT

Ground Vehicle Platooning Control and Sensing
in an Adversarial Environment
Samuel A. Mitchell

In the past few years, automated cars have ceased to be part of science fiction, and have instead become a technology that has been implemented, with partially automated systems currently available to customers.

One benefit of automated vehicle technology is the consistent driving patterns due to automation, instead of the inconsistency of distractible humans. Passengers of automated vehicles will be exposed to much less danger than the passengers of human-driven vehicles.

These statements will only be true as automated vehicle systems are scrutinized by experts to find flaws in the system. Security enthusiasts have already hijacked control of an automated car remotely with a cell phone [1]. As security flaws are exposed and removed, the vehicle automation community will become safer.

This research investigates the design of an automated vehicle and potential methods to protect the security of such a system. Vehicle control and guidance algorithms are analyzed and presented.

In the event of sensor failure, which could be naturally caused or performed by a malicious entity, automated vehicles cease to operate correctly, either crashing or returning control to people. Minimizing the risks of sensor failure can be achieved in multiple ways. One potential solution is to use available sensors to detect additional information about parameters of interest, such as using two radio antennas to triangulate the origin of a communicated signal. A method to extend the capability of a standard monocular camera by determine inter-vehicle distance from an image. This is one example of a countermeasure to an attack on a vulnerable system.

The focus of this work was to create a testbed for evaluating attacks and countermeasures against automated vehicle systems. A set of automated vehicle was designed to provide a platform to evaluate the security and reliability of individual cars and cars as a group. The development of this system is presented in this work. Vehicle guidance requirements and algorithms are discussed. A method to determine distance using a single camera is proposed. Finally, the vehicle system was evaluated as a viable method to validate an attack on a highway system.

For Sara, who kept me going when I wanted to throw in the towel.

ACKNOWLEDGMENTS

Many thanks to my committee members who have pushed me to accomplish more than I saw was possible.

In August 2013, I enrolled in a microcontrollers course taught by Dr. Ryan Gerdes. Through the semester, I was given the tools to develop embedded projects, but it felt as if I were given the keys of the city. Since then, I've had the privilege to explore hardware security and embedded development with this artist.

I am grateful for the guidance of Dr. Raj Sharma and his focus on getting the job done. He showed me that I know more than I realized — you can complete a task without 100% mastery of the subject.

When school and projects got hard, Dr. Tam Chantem reminded me how much fun engineering is. Every time I worked with her, she radiated her love of tackling an engineering problem.

My parents, Alan and Liz Mitchell, have shaped the way I approach work. My mom taught me that any job can be accomplished with some spit, baling wire, and an extra hour working on the project. My dad exhibits a desire to search for more knowledge in the simplest tasks.

Dan and Deann Hegsted for believing that my project would work, even when I didn't.

The IEEE USU Student Branch for giving me opportunities to share my work with other students.

Most of all, accomplishing the work in this thesis would not have been possible without my wife, Sara. She encouraged me to work on my projects when I wanted to quit. She waited up to buoy my spirits after I worked all night with no apparent progress. Sara, thanks for helping me through this adventure.

This project was funded in part by NSF Award#1410000.

Samuel A. Mitchell

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Background

In America, a total of 7 billion hours and 3 billion gallons of fuel were wasted due to congested traffic in 2014 [2]. The National Highway Traffic Safety Administration reported that 1.09 % of licensed drivers were injured in traffic accidents in 2013 [3]. This remarkable waste of resources due to the current highway system has prompted much research into the field of automated ground vehicles. By issuing the DARPA Grand Challenge in 2004, the US Department of Defense encouraged researchers to move forward in the field of autonomous ground vehicles.

Automated ground vehicles have been used for years in controlled situations [4], but modern computing capabilities have allowed for substantial progress in vehicle technology [5], for example, image processing can be performed at sufficiently high rates to allow control at highway speeds and it is relatively simple to switch control methods based on the situation, such as highway and city street driving. The transition to automated vehicles is one potential solution to the problems with the current highway system. The formation of a coordinated group of cars, or vehicle platoons, can substantially improve highway throughput and fuel efficiency [6]. In addition to the efficiency improvement, vehicle platoons increase the safety of each vehicle when compared to independent ground vehicles [7].

Due to the increased viability of autonomous vehicles, researchers started scrutinizing the security of these vehicle systems [8]. Researchers have started investigating the security of individual vehicles, a stream of vehicles, and vehicles behaving as a platoon; i.e. a group of vehicles which coordinate control laws. Dadras et al. revealed how a single vehicle could destabilize a platoon of vehicles [9]. This destabilization causes vehicles to oscillate at a resonant frequency, causing collisions that could result in injury or death.

The results found by the Dadras are substantial, but a collision tolerant system to evaluate the security vulnerabilities doesn't exist. In order to physically validate the findings of the SATS Group, a testbed to assess platooning technologies in an adversarial environment was developed. The control algorithms select the steering and velocity commands, thereby providing the most basic requirement in vehicle automation. Control algorithms calculate commands based on sensing the surrounding environment. In a security breach, an attacker could jam or spoof the sensors and communication which would validate the sensor inputs. This testbed was validated through the evaluation of lateral and longitudinal control algorithms, externally sensing control inputs without communication, and executing an attack on the vehicle platoon. The validated experimental testbed will continue to be used to demonstrate security flaws and countermeasures for autonomous ground vehicles.

1.2 Vehicle Control Algorithms

Autonomous vehicles determine what path to follow and safe velocity on that path via control algorithms. The poor selection of a control algorithm can cause the vehicle to induce a traffic jam, drive off the road, or collide with other vehicles. An effective control algorithm is essential to the viability of a vehicle system. One of the first problems of developing the testbed was determining the vehicle control algorithms that would be used in the system. The algorithms must reflect the methods used in mass-manufactured driver-assist technologies, but the system needed to operate with fewer sensing capabilities. Not all vehicle guidance algorithms calculate control output using the same information about the vehicle and surroundings, or states. The observability of the system states dictates which control method is viable for each specific application. Guidance controllers that utilize more states, such as the heading angle of the preceding vehicle, to determine control input may achieve more accurate target tracking [10].

In this work, we approach two vehicle control challenges, the lateral (steering) and longitudinal controllers. It is common for longitudinal control algorithms to be presented with the assumption that lateral guidance is managed separately from the longitudinal controller [11–15], but lateral guidance papers tend to present the longitudinal controller

bundled into the guidance problem [4,16–19]. This nonuniformity makes it difficult to compare longitudinal controllers with one lateral guidance law. The bundled lateral-longitudinal control laws were compared with a controller that controlled the lateral and longitudinal aspects of motion separately.

1.3 Expanding the Utility of a Monocular Camera

One of the major problems in ground vehicle automation is the ability to sense the region around the vehicle [20–23]. Various methods have been proposed, such as installing a magnetic reference track for the vehicles to follow, utilizing camera-based approaches [21,22], LIDAR rangefinders, and GPS or differential GPS to determine absolute position [23]. The problem with each of these methods is that the sensors could be jammed or spoofed [24], resulting in system instability.

System reliability can be improved by maintaining a redundant source of the input data required by the vehicle control system [25]. A major factor in the viability of an automated vehicle is the cost of the final system [26], which drives the need for inexpensive sensors. Monocular cameras are currently used in autonomous systems for determining the angle to a vehicle, which can be used to determine the distance to an object if the camera is angled toward the ground and the angle of observation is externally measured [22]. A method is proposed to determine the distance to a preceding vehicle without prior knowledge of the vertical position or orientation of the camera. This will provide a backup sensor for an automated ground vehicle, which increases the security via redundancy at low cost.

1.4 Platform Design and Incorporation in Platooning

This work describes the development of a vehicle system that behaves similarly to a full-scale vehicle. The hardware and software development are described. The goal of the effort was to provide an environment which would allow a rapid transition from simulation to implementation on ground vehicles. The controller and guidance system were implemented on a set of ten scale vehicles.

Design reliability is best achieved when researchers can verify simulated results on

physical systems, but the development of full-scale self-driving vehicles is cost prohibitive for many academic institutions. The cost barrier results in two categories of research for autonomous vehicles: inexpensive scale vehicles [10, 27, 28] or a limited number of full-scale vehicles [29–31]. Many vehicles are needed in order to study interactions between large numbers of vehicles in an automated highway system, but most scale vehicles do not drive at speeds sufficient to compare well with full-scale vehicles. Attacks performed by researchers could undermine vehicle stability, causing collisions. The use of full-scale vehicles in such applications would be cost-prohibitive. A collision-tolerant scale vehicle capable of traveling up to $10\,\mathrm{m\,s^{-1}}$ was developed in order to facilitate experimental validation for the SATS Group.

The proposed platform makes use of a split-controller, proposed by Rajamani [32], to allow the platooning-level 'high level' control algorithms to be separated from individual vehicle dynamics, which are managed by a 'low level' controller. Separating the two systems causes the development of a single control algorithm to work on a scale vehicle as well as a sports car.

Software systems to execute the controllers discussed in Section 1.2 were examined. The desired system would incorporate the split-controller management of a vehicle system in real time. A software package which fit the requirements was CarSim, which delivers accurate methods for simulating ground vehicles and exporting the project to run on target computers. Many automobile manufacturers use this system, but the system was cost-prohibitive for us to use. No open-source system provided an interface which would accept an arbitrary longitudinal and lateral controller, so I developed the *sats_car_ros* project as an extension of the ROS Libraries. This project is open-source and freely available at `github.com/smitchell7/sats_car_ros`.

1.5 Security Considerations for Automated and Semi-Automated Vehicles

The leading cause of traffic accidents is human error [3]. Automated highway systems have been proposed as one solution to this safety hazard. However, vehicles within an automated highway system can cause traffic jams or accidents if the control algorithms do

not attenuate position error [11]. Yanakiev [11] described the importance of considering self-driving cars as a stream of vehicles. Poorly selected control algorithms or gains can cause traffic jams on a highway. Longitudinal control algorithms are string stable if the system does not amplify oscillations in inter-vehicle distance.

Dunn [33] presented an analysis on the stability of string stable control algorithms in the presence of malicious vehicles. A coordinated attack is performed by inserting vehicles with unstable gains into the vehicle stream. When presented against standard vehicle control laws, Dunn's attack causes the inter-vehicle distances within the system to oscillate, which leads to collisions.

The impact of Dunn's attack on a stream of vehicles with heterogeneous control laws is analyzed. In particular, the cost and benefit of inserting human drivers into the system is examined. Validation of Dunn's attack was also performed on a scale experimental platform.

1.6 Outline of Thesis

An analysis of existing lateral control algorithms is presented in Chapter 2. A method to determine the distance to a vehicle using a monoculare camera is presented in Chapter 3. The method relies on license plate detection, so license plate detection methods are discussed. In Chapter 4, the development of a vehicle platooning platform is described in detail. Design challenges are discussed. An attack that incites traffic flow instability was presented in [33]. Chapter 5 contains an analysis of the attack against a stream of heterogeneous vehicles, i.e. the vehicles don't all use the same control algorithm. In addition to this analysis, the experimental validation of the attack in [33] is presented. Chapter 6 contains a description of the experimental validation of the experimental platform as a two-vehicle platoon. Finally, Chapter 7 provides a brief summary of this work and details future work.

CHAPTER 2

Control of a Vehicle Platoon

2.1 Overview

Nationally, freeway congestion costs the economy over $1 trillion annually. Automated vehicles are part of a solution to preventing future costly and time-wasting traffic jams. Currently there is a great deal of research on autonomous vehicle control [32]. Longitudinal control has been extensively studied [14, 34–37], and vehicles utilizing longitudinal control are commercially available [38]. Lateral control requires more inputs than longitudinal control, and the necessary inputs can be difficult to obtain.

Full lateral and longitudinal control of automated ground vehicles allows for extended tests of control algorithms and security vulnerabilities. The target location for the test runs is the Utah State University Electric Vehicle & Roadway Research Facility & Test Track, shown in Figure 2.1.

Lateral control algorithms were evaluated as candidates for the experimental platform, with the successful candidate implemented on the experimental platform. Section 2.2 presents current solutions to the lateral control problem. Section 2.4 compares the efficacy of three existing controllers. Simulations are performed on multiple trajectories. Ackermann-steer vehicles with differential braking. The development and results of the lanekeeping algorithm are in section 2.5.

2.2 Literature Review

Two commonly used methods for lateral control are lanekeeping and vehicle following. Lanekeeping is effective for vehicles driving far from other vehicles as well as when the vehicles are operating in congested highway areas. Lane keeping breaks down when lanes aren't well marked due to construction, redirection of traffic, or poorly marked roads. Vehicle

Fig. 2.1: The Electric Vehicle & Roadway Research Facility and Test Track was designed for the development of electric automated vehicles. It is an oval 400 m long. The control algorithms presented in this chapter are tested for road use on this track.

following is effective in the previous situations.

There are a variety of methods to control a vehicle following system. Rajamani provides an effective overview of the control problem [32], giving a general overview of longitudinal and lateral control.

Chien analyses what parameters are required to effectively track another vehicle [13]. He also discusses some requirements to prevent individual vehicles from producing the "slinky effect," or position errors between vehicles being propagated. Yanakiev described the slinky effect further, describing a system that removes the slinky effect as string stable [11]. A system is string stable if the position error between vehicles is attenuated by the control algorithm. String stability is a basic requirement for an efficient automated highway system, as a string unstable system can result in traffic jams and collisions.

Various authors [13,19,39–41] have proposed lateral controllers which control the lateral and longitudinal behavior simultaneously. Intuitively, combining the lateral and longitudinal control of a vehicle system reduces system complexity, but the principle of separability states that the controllers can operate independently.

Petrov's nonlinear adaptive method [19] utilizes adaptation to determine the preceding vehicle velocity. The controller minimizes the error between two virtual targets: one ahead of the rear vehicle and behind the preceding vehicle. Petrov's method is novel due to its use of adaptive control to determine actual vehicle states, rather than system gains. The controller is fairly rigid, where the system can be tweaked only with the headway distance.

The impedance based controller by Yi [18] uses a virtual mass-spring damper system for longitudinal control. The lateral control is designed to follow the preceding vehicle with a straight line — the following vehicle is essentially dragged behind the preceding vehicle.

Pure pursuit guidance has been used for years in aviation applications. The pure pursuit path tracking algorithm computes the angular velocity required to hit a target (Figure 2.2). The algorithm is a lateral guidance algorithm, so it would need to be paired with a longitudinal controller for fully automated control.

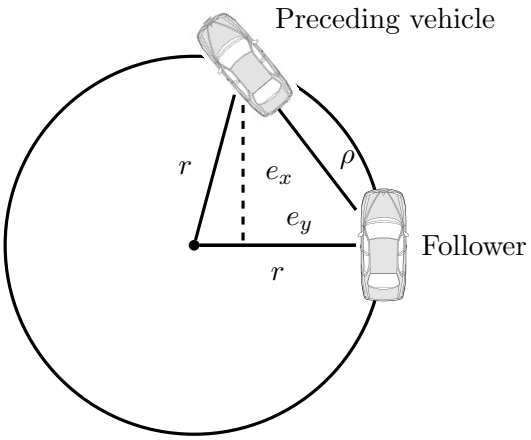

Fig. 2.2: Pure pursuit guidance turns the steering problem into basic geometry of a circle. The guidance law follows the circle of minimum radius that intersects with the target. This guidance law is not coupled with a longitudinal control law, so a string stable constant time headway law was utilized.

2.3 A Brief Overview of Modern Control Theory

The concepts in this chapter are presented assuming a basic understanding of modern control theory. This section briefly describes tools utilized in modern control theory and the differences between the subject and classical control theory.

The study of control systems boils down to one goal: to drive the states of a system to a desired final state. Within this goal, there are two major steps to the problem, (1) to model the system as a set of input and state vectors and (2) determine an input to drive the state vectors to a desired state.

Classical controls analyzes the relationship between one input and one output, and the system is made up of an integral-differential equation, see Equation 2.1. The integral-differential equation is simplified to a frequency response analysis using the Laplace transform, shown in Equation 2.2. For ease of analysis, the equation is generally represented as a transfer function between the input u and the state x in the Laplace domain, as seen in Equation 2.3.

$$u(t) = \frac{d}{dt}x(t) + x(t) - \int_0^t x(t)dt \tag{2.1}$$

$$\mathcal{L}\{x\}(s) = X(s) = \int_0^\infty e^{-st}x(t)\ dt \tag{2.2}$$

$$H(s) = \frac{X(s)}{U(s)} \tag{2.3}$$

Classical control theory is effective at modeling systems with one input and one output [42], but when a system has multiple inputs or multiple outputs, the coupled transfer functions grow increasingly complex. Instead of representing a multiple-input multiple-output system as a single transfer function, a system of coupled differential equations is used. Equation 2.5 shows the previously used integral-differential equation 2.1 as a state space representation of a single input multiple output system using the mapping in Equation 2.4, where x_i is the system state and u is the input to the system. By splitting the system, a controller can be designed to drive each state x_1 and x_2 to an arbitrary point.

$$\begin{aligned} x_1 &= x(t) \\ x_2 &= \int_0^t x(t)dt \end{aligned} \tag{2.4}$$

$$\begin{aligned} \frac{d}{dt}x_1 &= x_1 - x_2 + u \\ \frac{d}{dt}x_2 &= x_2 \end{aligned} \tag{2.5}$$

A simplified notation groups the system inputs and outputs as vectors, as seen in Equations 2.6 and 2.7.

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{2.6}$$

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{2.7}$$

State space representation simplifies the analysis of coupled differential equations and design of a controller. Each state of $\mathbf{x}$ is not necessarily easily measured, and the system input could skew the resulting observation, so a set of observation equations 2.8 with scalar coefficients c_i is used to describe the sensed system state.

$$\begin{aligned} y_1 &= c_1 x_1 + d_1 u_1 \\ y_2 &= c_2 x_2 \end{aligned} \tag{2.8}$$

$$\mathbf{y} = C\mathbf{x} + D\mathbf{u} \tag{2.9}$$

Once a system has been represented in state space form, future behavior is predicted from the structure of the coefficient matrices A, B, C, D. Stability, or whether the system states will come to rest at the origin, is measured by determining the eigenvalues of the A matrix.

A nonlinear system cannot be directly represented in a matrix with scalar values, so system stability is determined using linearization or Lyapunov stability analysis. Linearization is calculated by evaluating the Jacobian matrix at an equilibrium state, as shown Equation 2.10.

$$J = \left.\frac{d\mathbf{f}}{dx}\right|_{x_0} = \begin{bmatrix} \left.\frac{\partial f_1}{\partial x_1}\right|_{x_0} & \cdots & \left.\frac{\partial f_1}{\partial x_n}\right|_{x_0} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial f_m}{\partial x_1}\right|_{x_0} & \cdots & \left.\frac{\partial f_m}{\partial x_n}\right|_{x_0} \end{bmatrix} \tag{2.10}$$

Lyapunov stability analysis is comparable to analyzing the energy in a system. A Lyapunov candidate function $V(x,t)$ is selected such that it satisfies the constraints in Equation 2.11.

$$\begin{aligned} V(x,t) = 0 &\Leftrightarrow x = 0 \\ V(x,t) > 0 &\Leftrightarrow x \neq 0 \end{aligned} \tag{2.11}$$

The change in system energy $\frac{d}{dt}V(x,t)$ must be strictly negative to guarantee stability, while if the change in system energy is less than or equal to 0, the system is negative semidefinite, or it may result in oscillations about the origin.

Lyapunov stability analysis gives a sufficient but not necessary condition for the stability of a system to be determined. For example, one system can have two candidate functions

V_1 and V_2, where $\dot{V}_1 < 0$ but $\dot{V}_2 \geq 0$. In such a case, the V_1 candidate function showed system stability, and V_2 does not guarantee system instability.

Nonlinear control design generally relies on Lyapunov candidate functions to determine a controller which guarantees system stability.

This brief overview of control systems theory should give sufficient background to the reasoning and methods behind the remainder of this chapter.

2.4 Viability of Existing Lateral Control Algorithms

A suitable lateral controller for use in a vehicle platooning system was required for the experimental platform. Before a controller could be selected, a vehicle model had to be selected. The vehicle model used in simulation was required to correlate with both a standard highway vehicle and the differential steer vehicle intended for use in experimental validation. The states of a simplified vehicle are v velocity, ψ heading, and position p_x and p_y. These states are updated using Equation 2.12.

$$\begin{aligned} \dot{v} &= u_v \\ \dot{\psi} &= u_\psi \\ \dot{p}_x &= v\cos\psi \\ \dot{p}_y &= v\sin\psi \end{aligned} \tag{2.12}$$

Lateral controllers found in literature generally combined the lateral and longitudinal control into one nonlinear controller. The main purpose for the lateral control algorithm is to allow rapid testing of longitudinal controllers, which is simplified by separating the lateral and longitudinal controllers.

Three controllers were selected from Section 2.2 for further evaluation. The controllers were selected for the simplicity of implementation and the authors' claim of controller performance. The three controllers are shown in Table 2.1. The two major requirements of the system are low path error and longitudinal string stability [11]. The algorithms were compared via simulation according to these requirements.

Table 2.1: The three controllers selected for further evaluation. The controllers were selected to provide a wide selection of vehicle following algorithms.

Control name	Citation	Author
Impedance	[18]	Yi
Adaptive	[19]	Petrov
Pure pursuit	[13]	Chien

2.4.1 Nonlinear Adaptive Control

The nonlinear adaptive control presented by Petrov [19] utilizes error coordinates between target points. The preceding vehicle velocity v_1 and angular velocity ω_1 are unknown, while the lateral offset e_x, distance to vehicle e_y, and relative heading e_θ are known. The distance to the target point is L.

The control inputs are calculated in rotated coordinates (Equation 2.13), so they must be reversed to calculate velocity and angular velocity commands.

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos e_\theta & -L_2 \sin e_\theta \\ \sin e_\theta & L_2 \cos e_\theta \end{bmatrix} \begin{bmatrix} v_2 \\ \omega_2 \end{bmatrix} \tag{2.13}$$

The control inputs are calculated with estimated values of the preceding vehicle linear and angular velocity, $\hat{v}_1$ and $\hat{\omega}_1$, respectively.

$$\begin{aligned} u_1 &= -k_x e_x + \hat{v}_1 - \hat{\omega}_1 e_y \\ u_2 &= -k_y e_y - (L - e_x)\,\hat{\omega}_1 \end{aligned} \tag{2.14}$$

The adaptation law for the estimates is selected to provide a negative definite Lyapunov candidate function.

$$\begin{aligned} \dot{\hat{v}}_1 &= -\gamma_v e_x \\ \dot{\hat{\omega}}_1 &= \gamma_\omega L e_y \end{aligned} \tag{2.15}$$

This results in a system where velocity is directly controlled. For this evaluation, the velocity isn't controlled via acceleration $\dot{v} = k\,(v_d - v)$, but the variable is saturated at a maximum acceleration value.

2.4.2 Impedance Based Control

The impedance based control uses a bidirectional spring-damper model to determine desired acceleration and lateral control outputs. The inputs to the system are the position and velocity from a fixed vehicle frame. The position and velocity are represented in Cartesian coordinates using the 2 element vector p_n. These inputs could be obtained via GPS sensors with inter-vehicle communication. The outputs are the desired acceleration and angular velocity. The controller is shown in Equation 2.16.

$$\begin{bmatrix} \dot{v} \\ \omega \end{bmatrix} = \frac{1}{m} \begin{bmatrix} \cos\theta & \sin\theta \\ -\frac{\sin\theta}{v} & \frac{\cos\theta}{v} \end{bmatrix} U$$
$$U = \{k\left(p_{n+1} - p_n - d_n\right) - k\left(p_n - p_{n-1} - d_{n-1}\right) + c\left(\dot{p}_{n+1} - \dot{p}_n\right) - c\left(\dot{p}_n - \dot{p}_{n-1}\right)\} \tag{2.16}$$

2.4.3 Pure Pursuit with Longitudinal Control

The pure pursuit algorithm controls the target curvature (Equation 2.17).

$$\kappa = \frac{\cos\theta}{\rho} \tag{2.17}$$

Curvature κ is related to angular velocity $\dot{\psi}$ through a velocity multiplier v (Equation 2.18).

$$\dot{\psi} = v\kappa \tag{2.18}$$

The commanded acceleration is calculated using Yanakiev's constant time headway controller [11], Equation 2.19. The desired acceleration u_v is calculated from the inter-vehicle distance ρ, velocity v, time headway h, and the relative velocity between vehicles $\dot{\rho}$. Gains k_p and k_d must be selected to maintain string stability.

$$u_v = k_p\left(\rho + vh\right) + k_d\dot{\rho} \tag{2.19}$$

2.4.4 Simulation Design

The controllers were simulated using the basic vehicle dynamics model from Equation 2.12. The acceleration input was saturated at $5\,\mathrm{m\,s^{-1}}$ to make the vehicles more realistic. The controllers were simulated on a track with three maneuvers. The path is shown in Figure 2.3, with the maneuvers described in Table 2.2 and shown in 2.3. The lead vehicle traveled at $4\,\mathrm{m\,s^{-1}}$ in a clockwise circle with a 15 m radius, transitioned to $2\,\mathrm{m\,s^{-1}}$ in a counter-clockwise circle with a 10 m radius, then finished with a straight line at $5\,\mathrm{m\,s^{-1}}$. These maneuvers were considered sufficient to represent difficult driving conditions that the vehicle might encounter, thereby determining the control algorithm performance. The lead vehicle followed the trajectory without error, and the following vehicles used the states of the preceding vehicle as an input to its respective control algorithm. The observed position and angle to each vehicle had noise added to simulate the effects of sensor error.

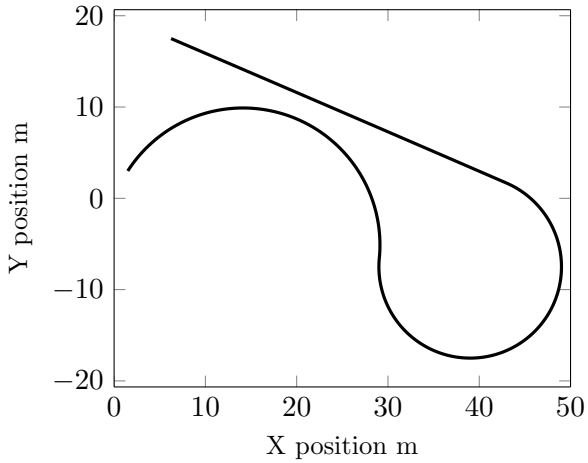

Fig. 2.3: The simulation trajectory for the lead vehicle.

Table 2.2: The curve radius and velocity of the lead vehicle. The path is shown in Figure 2.3.

	Turn radius (m)	Duration (s)	Velocity (m/s)
Left turn	15	10	4
Right turn	10	22	2
Straight	∞	8	5

2.4.5 Simulation Results

The controllers were each simulated with the recommended control gains from the respective papers, shown in Table 2.3. The lead vehicle followed the path described in 2.2, and it was followed by 9 following vehicles. The two requirements, lateral path error and string stability were compared. The cars are meant to drive on a road, so the maximum path error of any of the vehicles once the system reached steady state was used to determine the viability of the controller.

The string stability requirement can be mathematically proven, but if position error is attenuated through the system of vehicles, that is sufficient to show string stability.

The results from the simulation of the three control algorithsm is shown in Table 2.4. The three algorithms were determined to be useful in their own respects. Yi's Impedance controller quickly dampened any position error in the system, but the path error was extremely large. The control algorithm essentially dragged the string of vehicles behind the leader without any active steering.

Petrov's Adaptive controller followed very accurately, but the fixed inter-vehicle distance caused the system to become unstable. The desired distance could be modified to add a velocity-based headway distance, i.e. $L = L_0 + vh$, but the controller is not string stable.

The pure pursuit algorithm had a small path error was small, and the separate longitudinal controller allowed the selection of a string stable controller. This algorithm performed the best, and was selected for the implementation described in Section 4.

2.5 Lanekeeping for Ackermann-Steer Vehicles

In this section, a path-following controller was proposed. The controller is based on

Table 2.3: Gains used for simulation of the control algorithms from Table 2.1.

Algorithm					
Petrov	$k_x = 8$	$k_y = 20$	$\gamma_v = 5$	$\gamma_m = 5$	L =4 m
Impedance	m =1300 kg	k =15 000 kg s^{-2}	c =18 000 kg s^{-1}		
Pure pursuit	$k_p = 1$	$k_d = 2$	$h = 0.5$	L =1 m	

Table 2.4: Simulation results of the three lateral controllers. The pure pursuit controller had the minimum path error, and it was string stable.

	Petrov	Impedance	Pure pursuit
Max path error cm	12.4	852.0	4.5
String stable		×	×

optimal control, with the model for an Ackermann-steer vehicle with differential braking taken from [16].

2.5.1 Vehicle Dynamics

The Ackermann-steer model with differential braking as presented in [16]. The paper utilizes backstepping to provide three input vectors which can arbitrarily control the vehicle states. The variables used by [16] are used in this section, and are defined in 2.5.

The equations of motion are shown in 2.20. A linearized tire model was used, and the lateral forces are calculated using the stiffness coefficient and the desired turn angle, as seen in Equation 2.21. C_f and C_r are the front and rear cornering stiffness constants, respectively. α_f and α_r are the front and rear lateral slippage, which is calculated in Equations 2.22 and 2.23.

$$\begin{aligned} m\dot{u}_x &= F_{xr} + F_{xf}cos\left(\delta\right) - F_{yf}sin\left(\delta\right) + mru_y \\ m\dot{u}_y &= F_{yr} + F_{xf}sin\left(\delta\right) + F_{yf}cos\left(\delta\right) - mru_x \\ I_z\dot{r} &= aF_{xf}sin\left(\delta\right) + aF_{yf}cos\left(\delta\right) - bF_{yr} + \frac{d}{2}\left(\Delta F_{xr} + \Delta F_{xf}cos\left(\delta\right)\right) \end{aligned} \tag{2.20}$$

$$\begin{aligned} F_{yf} &= -C_f\alpha_f \\ F_{yr} &= -C_r\alpha_r \end{aligned} \tag{2.21}$$

$$\alpha_f = tan^{-1}\left(\frac{u_y + ra}{u_x}\right) - \delta \tag{2.22}$$

Table 2.5: Variables used through Section 2.5.

m	mass of the vehicle
I_z	moment of inertia
a	distance from center of gravity to the front wheel
b	distance from center of gravity to the rear wheel
d	distance between the two wheels (front or rear)
F_{xf}	longitudinal front tire force
F_{xr}	longitudinal rear tire force
F_{yf}	lateral front tire force
F_{yr}	lateral rear tire force
u_x	longitudinal velocity
u_y	lateral velocity
r	yaw rate
δ	front wheel turning angle

$$\alpha_r = tan^{-1}\left(\frac{u_y - rb}{u_x}\right) \tag{2.23}$$

Now, neglecting the longitudinal forces, substituting for the lateral forces in Equations 2.20, and making small angle approximations we get resulting equations of motion in Equation 2.24.

$$\begin{aligned} m\dot{u_x} &= mru_y + C_f\left(\frac{u_y - rb}{u_x}\right)\delta \\ m\dot{u_y} &= mru_x - C_r\left(\frac{u_y - rb}{u_x}\right) - C_f\left(\frac{u_y + ra}{ux}\right) + C_f\delta \\ I_z\dot{r} &= -aC_f\left(\frac{u_y + ra}{ux}\right) + bC_r\left(\frac{uy - rb}{u_x}\right) + aC_f\delta \end{aligned} \tag{2.24}$$

These equations can be rewritten in state space form for simplicity.

$$\ddot{q} = M^{-1}\left(f\left(\dot{q}\right) + g\left(\dot{q}, u_c\right)\right) \tag{2.25}$$

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2.26}$$

$$f(\dot{q}) = \begin{bmatrix} mru_y \\ -C_r\left(\frac{u_y - rb}{u_x}\right) - C_f\left(\frac{u_y + ra}{u_x}\right) - mru_x \\ -aC_f\left(\frac{u_y + ra}{u_x}\right) + bC_r\left(\frac{u_y - rb}{u_x}\right) \end{bmatrix} \tag{2.27}$$

$$g(\dot{q}, u_c) = \begin{bmatrix} Cf\left(\frac{u_y + ra}{u_x}\right)\delta \\ C_f\delta \\ aC_f\delta \end{bmatrix} \tag{2.28}$$

Where $\dot{q} = [u_x,\ u_y,\ r]$ The local frame coordinates of the vehicle are described by Equations 2.29, 2.30, and 2.31.

$$\dot{p_x} = u_x \tag{2.29}$$

$$\dot{p_y} = u_y \tag{2.30}$$

$$\dot{\psi} = r \tag{2.31}$$

For general control purposes, the vehicle position ω is represented as a vector of global coordinates s, e, and ψ, which represent the distance along the roadway, distance of the vehicle from the road centerline, and the heading angle. These coordinates can be computed using the update law in Equation 2.32.

$$\begin{bmatrix} \dot{s} \\ \dot{e} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ \psi \end{bmatrix} \tag{2.32}$$

The system state vector x uses the local position coordinates p_x and p_y, heading ψ, longitudinal velocity u_x, lateral velocity u_y, and the global coordinates ω, as seen in Equation 2.33.

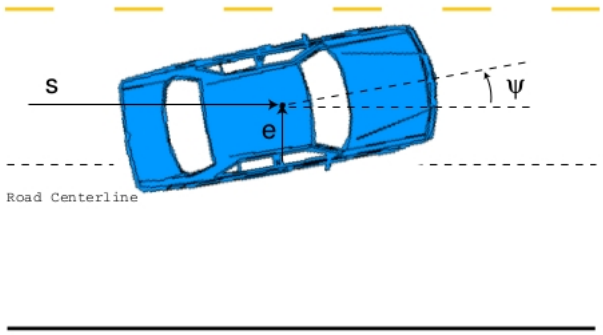

Fig. 2.4: The vehicle is controlled on a straight line. The global coordinates used as s and e, which correlate to the distance along the roadway, distance of the vehicle from the road centerline, and the heading angle.

$$x = \begin{bmatrix} p_x, & p_y, & \psi, & u_x, & u_y, & r, & s, & e \end{bmatrix}^T \tag{2.33}$$

2.5.2 Model Linearization

The vehicle model is linearized about a certain trajectory so that the stability analysis and the controller design process become easier. The equilibrium points are a constant longitudinal velocity and all other states are zero. Feedback linearization and Jacobian linearization methods were both used.

We begin with the vector states $\dot{q}$. Feedback linearization was used to develop the update laws. Three control laws were proposed for the terms in $g\left(\dot{q}, u_c\right)$ matrix terms as follows.

$$\frac{C_f}{m}\left(\frac{u_y + ra}{u_x}\right)\delta = -ru_y + u_1 \tag{2.34}$$

$$\left(\frac{C_f}{m}\right)\delta = C_r\left(\frac{u_y - rb}{u_x}\right) + C_f\left(\frac{u_y + ra}{u_x}\right) + ru_x + u_2 \tag{2.35}$$

$$\left(\frac{aC_f}{I_z}\right)\delta = \frac{aC_f}{I_z}\left(\frac{uy + ra}{ux}\right) - \frac{bC_r}{I_z}\left(\frac{u_y - rb}{u_x}\right) + u_3 \tag{2.36}$$

The velocities in each direction are controlled directly.

$$\begin{bmatrix} \dot{u}_x \\ \dot{u}_y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ r \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{2.37}$$

Global coordinates are updated based on the directional velocities and heading angle.

$$\begin{aligned} \dot{s} &= u_x cos\left(\psi\right) - u_y sin\left(\psi\right) \\ \dot{e} &= u_x sin\left(\psi\right) + u_y cos\left(\psi\right) \end{aligned} \tag{2.38}$$

These two equation were linearized using Jacobian method and the same equilibrium points.

$$\dot{x} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{2.39}$$

2.5.3 Controller Design

In order to design a controller, we need first to know which states are controllable and which are not. For that, the controllability matrix was computed by Equation 2.40.

$$\mathcal{C} = \begin{bmatrix} B & AB & A^2B & \dots & A^nB \end{bmatrix} \tag{2.40}$$

The resultant matrix is found to be of rank six, which means that six of the system states are controllable.

By looking back at the linearized matrix A, it is clear that the first and last two rows represent the same positions in different coordinate frames. Since the extra states are used for visualization purposes, the final states were included in control design. The new states of x are shown in 2.41. The controllability was recomputed with an A that had six states, and the rank was found to be six.

$$x_c = \begin{bmatrix} \psi, & u_x, & u_y, & r, & s, & e \end{bmatrix}^T \tag{2.41}$$

Using the new vector of controllable states, the proposed control law is shown in Figure 2.42, where x_{cd} is the vector of desired states values, and K is a gain matrix.

$$u = -K\left(x_c - x_{cd}\right) = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{2.42}$$

The only desired states are the global coordinates. The remaining terms of the desired values vector are selected as zero since we do not focus on their control. Next, the gains matrix is computed using the LQR method which is

$$J = \int_0^\infty \left(x^T Q x + u^T R u\right) dt \tag{2.43}$$

The elements of the Q matrix were chosen so that more weight is given to the global coordinates.

2.5.4 Path Planning Methods

When the system is controlled by strictly proportional methods, the system strictly uses skidding to steer the vehicle. This is not reasonable for most vehicle systems, so a path planning approach is required to realistically steer the vehicles. There are many methods

to estimate a path to a target. In this work, the authors investigate four potential path planning methods: proportional control, proportional derivative, pure pursuit, and cubic spline interpolation.

Proportional Control

The most basic path planning method is to draw a straight line from the vehicle to its target, and use that angle as the control input, as shown in 2.44.

$$r_{des} = \arctan \frac{e_y}{e_x} \tag{2.44}$$

Proportional Derivative Control

An improvement to the proportional control is to incorporate the desired final direction of the system in the path planning method. The resulting equations are:

$$\psi_{des} = e_\psi \tag{2.45}$$

$$r_{des} = \arctan \frac{e_y}{e_x} \tag{2.46}$$

Cubic Spline Interpolation

Cubic spline interpolation is a method to draw a cubic function through two points. The method utilizes the position and derivative of two points. In spatial coordinates, $\tan \psi$ is the derivative of the point. The output of the cubic spline interpolation is of the form

$$y = c_3 x^3 + c_2 x^2 + c_1 x + c_0. \tag{2.47}$$

The curvature of a curve at any point is

$$\kappa = \frac{\ddot{y}}{(1 + \dot{y}^2)^{3/2}}. \tag{2.48}$$

Spatial curvature relates to r through current velocity

$$r = v\kappa. \tag{2.49}$$

If the cubic spline is calculated using error coordinates, the value and slope of x_1 are 0. The resulting value for r is

$$r_{des} = v\frac{c_2}{\left(1 + c_1^2\right)^{3/2}} \tag{2.50}$$

Pure Pursuit

Pure pursuit control is a basic missile trajectory algorithm [43]. A circle is drawn tangent to the vehicle and through the target, as shown in Figure 2.2.

The radius of the circle is found using trigonometric identities, resulting in 2.51.

$$R = \frac{e_x^2 + e_y^2}{2e_y} \tag{2.51}$$

This result is transformed to yaw rate using 2.49 and 2.52.

$$r_{des} = v\frac{2e_y}{e_x^2 + e_y^2} \tag{2.52}$$

2.5.5 Results

The control system was simulated using a target that followed a path 2.53, shown in Figure 2.5.

$$\begin{aligned} x &= t + \cos\left(0.1t\right) \\ y &= \sin\left(0.1t\right) \end{aligned} \tag{2.53}$$

Initial conditions are shown in 2.54. Values for Q and R were set to values in 2.55.

$$x_0 = \begin{bmatrix} 0.5 & 3 & 10^\circ & 2 & 0 & 0 & 0.5 & 3 \end{bmatrix}^T \tag{2.54}$$

$$
\begin{aligned}
Q &= I \begin{bmatrix} 2 & 1 & 1 & 5 & 50 & 500 \end{bmatrix}^T \\
R &= I \begin{bmatrix} 1 & 5000 & 1 \end{bmatrix}^T
\end{aligned} \tag{2.55}
$$

Each of the four path planning methods were simulated. Each method was compared using the mean squared error and required lateral control input u_y. These metrics are selected to ensure accuracy to the path, and u_y requires the vehicle to skid to achieve the desired state. The results from the simulation are shown in Table 2.6. The pure pursuit control outstrips the other methods in both the error and control input.

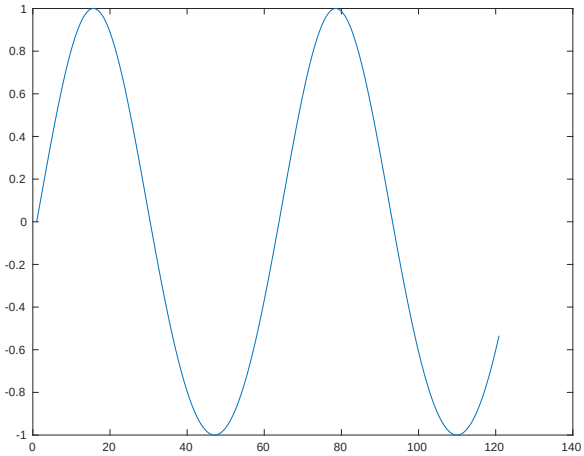

Fig. 2.5: The simulation path as seen in Eqn. 2.53.

Table 2.6: The resulting error for the lanekeeping control methods.

	Path Error	Maximum u_y
Proportional	0.939	0.681
Proportional Derivative	0.618	0.682
Pure Pursuit	0.088	0.643
Cubic Spline Interp	0.385	0.796

2.6 Conclusion

Lateral control is essential to lengthening the duration of test runs for the experimental vehicle platform. Three existing lateral control algorithms were simulated to determine the

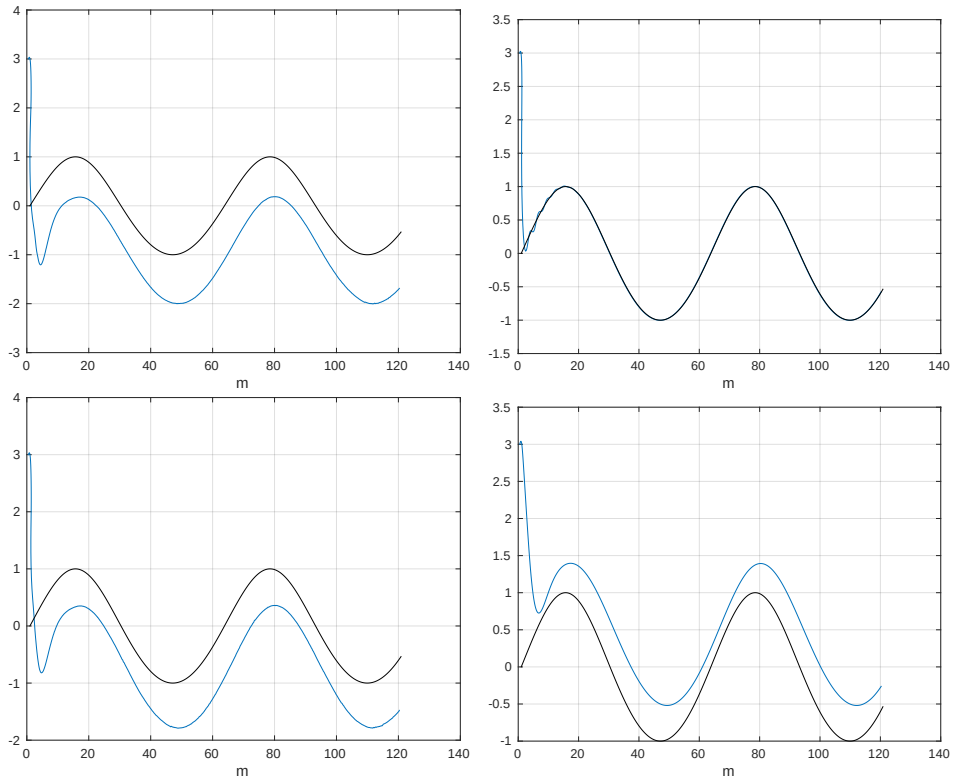

Fig. 2.6: Each controller (blue) compared to path (black), in raster scan order. (a) Proportional control, (b) Proportional Derivative control, (c) Pure Pursuit, (d) Cubic Spline Interpolation.

best candidate for use on the test track. Pure pursuit guidance achieved the lowest path error while maintaining string stable operation.

A lanekeeping algorithm for Ackermann-steer vehicles with differential braking was developed. The algorithm converged to the path quickly, but the underlying model of differential braking was incompatible with the experimental platform.

In order to implement the control algorithms, the control inputs must be obtained by local sensors or communication. Chapter 3 discusses potential local sensing methods.

CHAPTER 3

Vehicle State Estimation with a Monocular Camera

3.1 Background

Lateral control of a ground vehicle requires control inputs that can be difficult to obtain. The difficulty to obtain the inputs can induce the manufacturer to utilize only one sensing system, thereby creating a single point of failure. Upon the event of sensor failure, the vehicle should seamlessly transfer control to the person driving the car. Potential negative outcomes could result in fatality. In one instance, a camera-based automated vehicle failed to distinguish another vehicle, which caused a fatal collision [44].

One of the problems in ground vehicle control is observing the states of the surrounding area. Current methods utilize a rotating LIDAR sensor which maps the immediate surroundings of the vehicle hundreds of times a second. While these sensor systems are sub-millimeter accurate, the high production costs ($30,000 to $85,000) of such systems restricts manufacturers from adopting the technology [45].

In systems with a single point of failure, malicious agents could intentionally disable or spoof one sensor, thereby rendering the control system useless. The inputs to potential control algorithms with sensing methods are shown in Table 3.1. This chapter focuses on a method to improve the reliability of sensing required control inputs.

The Subaru EyeSight system utilizes a stereoscopic camera to approximate the distance to surrounding objects [49]. This system determines the distance and relative velocity of the

	Distance ρ	Relative velocity $\dot{\rho}$	Bearing angle θ	Jamming / Spoofing Methods
GPS with communication	✓	✓	✓	[46]
Stereoscopic camera	✓	✓	✓	[47]
Monocular camera			✓	[47]
LIDAR	✓	✓	✓	[48]
Lidar-Lite	✓	✓		[48]

objects directly in front of the vehicle and provides lane-departure warnings. This system costs the consumer an extra $5000.

Consider an attack from a capable adversary which renders useless the standard sensors utilized in platooning: GPS, communication, LIDAR, one side of a stereo camera. We show that it is possible to maintain vehicle stability through such an attack.

The experimental platform under development utilizes a fixed LIDAR module to determine distance to the preceding vehicle. The distance measurement is a narrow beam with a limited field of view. We propose a method to use a monocular camera to determine the distance to a fixed-size object. The monocular detection method relies on vehicle detection, specifically license plate detection, to measure the distance to the vehicle. This method is effective in creating low-cost systems or as a failsafe method to provide system reliability under sensor failure.

3.2 Review

A review of state of the art methods of vehicle detection most closely related to our application is presented. The focus of this review is on the feasibility of utilizing vision-based approaches in controls-based applications.

Stein *et al.* [34] estimated distances to a preceding vehicle with a single camera. The work focused on using similar triangles to estimate the inter-vehicle distance. While the error increases quadratically, the percent error in inter-vehicle distance increases linearly, yielding a 10% error at 90 m. The authors demonstrated the possibility of utilizing a single-camera vision system for vehicle following, but an algorithm to do so was not provided. This section discusses potential vehicle detection methods.

3.2.1 Vehicle Detection

One method for vehicle detection is presented in [50]. The author described current methods for vehicle detection, recommending knowledge-based approaches during the day and multiple cue integration with a particle filter for low-light conditions. Particle filtering is widely used in target tracking. The presented particle filter utilizes four vision cues to

detect vehicle candidates: vertical edge, taillight, underneath shadow, and symmetry. Using this method, multiple vehicles can be tracked simultaneously during the day or night. This method would be computationally expensive on an embedded system.

A statistical approach for automatic vehicle detection based on local features [51] is one potential method for detection. The use of local features allows for detection despite vehicle geometric variance and partial occlusions. The PCA+ICA model is capable of computing one result in 8.7 seconds on a Pentium 4 3.2 GHz processor. These results are improved to 2 Hz when a weighted Gaussian Mixture Model was implemented. An autonomous vehicle driving at highways speeds should update the control algorithm much faster than this — a vehicle traveling at $20\,\mathrm{m\,s^{-1}}$ would travel 10 m between updates, which is sufficient to cause collisions.

3.2.2 License Plate Detection

Vehicles are required to mount a rear-facing license plate for law enforcement purposes. This is an easily identifiable feature on every vehicle, which can be used for vehicle detection. Some drawbacks to these methods are the fact that some recently bought vehicles do not have a license plate. Also, some vehicles do not mount the license plate in the center of the vehicle. Despite these exceptions, these methods provide a reliable method to detect the preceding vehicle.

A survey of license plate recognition was presented in [52]. We are concerned with license plate detection, but recognition requires detection as a step of the process. The authors note that the efficacy of the recognition algorithm relies on the quality of the acquired images. The main methods of license plate detection are done using boundary or edge information, connected component analysis, texture features, color features, or character features. Each method has its strengths and weaknesses, but the fastest methods are the ones relying on boundary or edge features. The biggest drawback to these methods is the sensitivity to unwanted edges in the image. The target application is on a 1 GHz processor, so the boundary algorithms are tested to determine viability of detection.

A basic example of edge-based license plate detection is [53]. Edge based methods

utilize two major steps: preprocessing and candidate verification. The focus of [53] is the preprocessing segment of an algorithm, because effective preprocessing can achieve high detection rates without any candidate verification. The algorithm initially detects vertical edges in the image; text contains a high density of vertical edges. An edge density map is generated then binarized, after which the image is dilated to remove stray lines. The authors found a detection rate of 96 % on a set of 478 images.

One boundary algorithm [54] claimed a significant speedup (10 ms computation time per image) with an accuracy increase when compared to other detection methods. The method employed down sampling, which reduced preprocessing time, or the time to determine candidate regions. The preprocessing was followed by extracting the covariance of 13 photo parameters; i.e. RGB values, detected horizontal / vertical edges. A support vector machine (SVM) was then used to determine the validity of the candidate region.

3.2.3 Sensor Reliability

Modern vehicles utilize various sensor fusion methods to extend system reliability. Luo described various sensor fusion methods [55]. These methods could be used to determine faulty or spoofed sensors. Gray and Vantsevich considered estimation and control strategies with unreliable sensors [56].

3.3 License Plate Detection

The algorithms obtained from Section 3.4 require a vehicle detector, and the distance estimator requires a license plate detector. Three vehicle detection methods were implemented to determine the algorithm viability for use in a resource constrained environment. Two of the detectors were license plate detection methods [53,54], and the last detector was an adaptive template detection algorithm [57].

The basic concept of a license plate detector comes from the concentration of vertical lines in the license plate region. Most algorithms create a density map of the vertical edges in the image. After grouping the densities together using morphological operations, the algorithm uses some selection criteria to determine the true candidate regions.

The fast license plate extraction method by Bai [53] (hereafter called Bai Fast) performed preprocessing on the image, then restricted the candidate regions by the width to height ratio. This method is simple and was performed quickly, but the system was likely to show false positives unless the preprocessing method was tuned precisely to the region.

The covariance based license plate detector by Feng [54] down sampled the image to determine candidate regions, then a 169 element covariance matrix was calculated for each candidate region. Some of the inputs to the support vector machine were the region width, height, edge density, and color. The covariance matrix was passed to the support vector machine for every candidate region. If the preprocessing produced many candidate regions, the system took that much longer to determine the best fit for the system.

The Kernelized Correlation Filter (KCF) [57] was implemented as an alternative to license plate detection methods. The code for the filter came from the kind people at PhotoRithm. The KCF was very effective at detecting and tracking a moving object. The authors claim sub-pixel accuracy of the detected object. In addition to fine resolution, the system performs its comparison in the frequency domain, which results in extremely fast performance. The filter adaptively updates its template, which will track the vehicle even if the car turns a corner, changing the image from its original template. Despite these benefits, the KCF tracks an object on a fixed window size. The algorithm can effectively detect the angle to a detected object, but no other parameters can be extracted.

3.3.1 Testing on Sample Images

Three detection methods were selected, and a set of sample images was obtained to determine the efficacy of the algorithms. Due to privacy constraints, there is no publicly available database of American license plate images, which makes it difficult to compare license plate detection methods [58]. The MediaLab LPR Database [59] was generated to fill the gap in available license plate images, but the dimensions are different from the standard American plates. Despite the differences, the images were deemed analogous to American plates, and the database served as a baseline to measure the accuracy of the detection algorithms.

Three vehicle or license plate detectors were implemented and compared. Both of the license plate detection methods required fine tuning for each dataset, which means that re-tuning would be required if the algorithms were used with a new camera. The two license plate detectors, accuracy rates of each detector were much lower than claimed in the original articles, and the authors didn't provide data sets to test against. Due to the

The vehicle detector tested was the KCF. The KCF didn't specifically detect license plates; instead it adaptively detected the target. Due to the tracking nature of the KCF, the image set of separate license plates could not be used to test the algorithm. Instead, the KCF was initialized with the target vehicle directly in front of a camera, and the filter was computed on the video in real time.

The KCF was very effective for determining the bearing to the target vehicle. The detected region adaptively fit the vehicle based on previously detected regions, which meant that the license plate could not be specifically detected.

As can be seen in Table 3.1, the KCF achieved the highest accuracy rates in addition to real-time operation.

3.3.2 Migration to Beaglebone Black

After comparing the detectors on desktop machines, two algorithms were ported to the target platform, the Beaglebone Black. The support vector machine–based algorithm was processing frames at roughly 0.25 Hz on an i7 processor, so that algorithm wasn't ported to the Beaglebone Black.

The Beaglebone Black utilizes a 1 GHz ARM Cortex-A8 processor with NEON hardware acceleration. OpenCV was compiled with enabled acceleration [60–62] to maximize the throughput of the processor. Despite these considerations, the Beaglebone Black could

Table 3.1: A comparison of vehicle detection methods.

	Bai Fast [53]	Feng Effective [54]	Henriques KCF [57]
Detection rate	75 %	20 %	N/A
Time Intel i7-4770K 3.50 GHz	33 µs	4–20 s	33 µs
Time ARM Cortex A8 1 GHz	1 s	N/A	125 µs

Fig. 3.1: Samples from the set of license plate images. The plates have different dimensions than American license plates, but the vertical edge densities are similar.

grab 640x420 pixel images from the USB webcam at 23 Hz.

The fast license plate detector was compared against the KCF. The KCF was implemented with two kernel sizes: 256x256 pixels and 32x32 pixels. The smaller kernel size proved to track the license plate as well as the 256x256 pixel kernel. The obtained frame rates are presented in Table 3.1. In addition to outperforming the other algorithms in accuracy, the KCF was the fastest algorithm tested, at 8 Hz.

Despite the good performance of the KCF, our application required a resolution of 10 Hz while simultaneously running other applications. Due to the timing constraints, we selected an alternative to onboard image processing, as discussed in Section 3.4.3.

3.4 Parameter Estimation

Consider a vehicle platoon with a malicious agent nearby. The control inputs required by each vehicle are the inter-vehicle distance ρ, inter-vehicle relative velocity $\dot{\rho}$, and the angle to the preceding vehicleθ. These parameters are generally obtained via GPS with

inter-vehicle communication, a 360° LIDAR unit, or stereo vision camera. The attacker jams all of the sensors except for one of the cameras on the stereo vision camera.

This section describes the methods used to determine the parameters required by the control algorithms using a single camera.

3.4.1 Angle to Target

When a camera snaps a picture, it records the wavelength of light at many observed points, or pixels. If the position of the camera is not known, the physical information that can be extracted from the image is the observed relative angles between two points. With some knowledge about the positioning of the camera, a bird's eye view can be generated using a technique called Inverse Perspective Mapping [63]. The angle to a target is needed for the control algorithms presented in Table 3.1. The Inverse Perspective Mapping method requires the vertical height and camera angle. The computation is computationally complex, so we tested a simpler method to measure the effectiveness of the method against the real-world.

The angle to the target θ is determined by determining the center of the image, then measuring the arctan of the pixel positions x and y, multiplied by a constant c.

$$\theta = c \arctan \frac{y}{x}. \tag{3.1}$$

This angle mapping is commonly used to determine the angle to a target. The accuracy of this method will be analyzed in Section 3.4.4.

3.4.2 Distance to Target with Fixed Size

Stereo vision is generally required to estimate the distance to a given object. If one lens of a stereo vision camera is obscured or disabled, accidentally or maliciously, the stereo camera is rendered useless. This gives rise to the need for a distance estimator that can be utilized with a monocular camera. The biggest drawback to using monocular cameras in vehicle detection is the lack of depth information. Despite this, monocular cameras are

very effective at detecting an object with low computational complexity. If an object has a known size, a monocular camera can be calibrated to determine the distance to the object.

In the United States, there are two sizes of license plates: motorcycle 4 in by 7 in and standard 6 in by 12 in. This work performs a feasibility study to determine the efficacy of license plate detection for distance estimation, so we analyze one style of license plate. The motorcycle license plate dimensions were used due to the limited size of the target platform. This method would not be effective under a variety of conditions, such as if an attacker changed the size of the license plate on the vehicle, removed the license plate, or placed the license plate in an atypical orientation.

The license plate detection algorithms described in Section 3.3 detect the height and width of the plate in pixels in addition to the pixel position of the license plate. This information can be utilized to approximate the distance to the preceding vehicle. The license plate is expected to be observable and within the range of the camera.

A geometry-based distance estimator is shown in Figure 3.2, with the math in Equation 3.2. This method relies on basic trigonometry to determine the altitude of a triangle. The ratio of distance ρ to the license plate height in meters l can be described as the opposite over hypotenuse of a right triangle with respect to the observed angle ϕ. The pixels h were related to the angle ϕ by a direct ratio. Tunable parameters were added to the height and pixel angles, as shown in Equation 3.3.

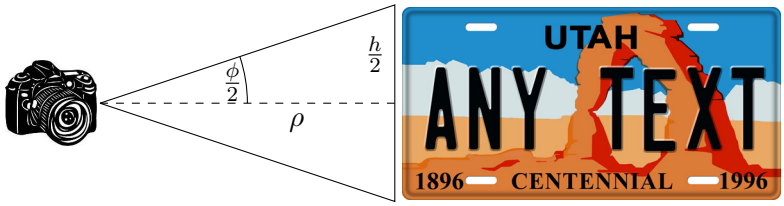

Fig. 3.2: The height of an image can be determined using the tangent operation. This operation is as accurate as the angle estimate from pixels.

$$\rho = \frac{l}{2 \tan \frac{\phi}{2}} \tag{3.2}$$

$$\rho = \frac{al}{2\tan\frac{bh}{2}} \tag{3.3}$$

The height measure was used because the observed license plate width can vary depending on the heading of the preceding vehicle, but the observed height remains relatively constant with variations in the preceding vehicle heading.

As the distance to an object increases, the observed size of the object decreases exponentially. Due to this, a second metric was proposed using the exponential decay the algorithm height, as shown in Equation 3.4. The tunable parameters of this method do not correspond directly to any physical measurement.

$$h = ae^{-b\rho} + c \tag{3.4}$$

Mapping the license plate height in pixels to distance is merely the inverse of Equation 3.4. This is shown in Equation 3.5.

$$\rho = \left| -\frac{1}{b}\log\frac{h-c}{a} \right| \tag{3.5}$$

The results of the geometry and exponential models are described in Section 3.4.4.

3.4.3 Data Collection

Testing the conjecture about using the area of license plates for distance measurements is contingent upon an effective license plate detector. The limited resources of the Beaglebone Black required image processing to be performed remotely.

The Pixy Camera [64] was utilized to provide a simple vehicle detector. The camera utilizes an on-board processor to detect color signatures, reliably providing a detected region at 50 Hz. The camera operates using the hue information taken from the RGB camera on-board. The processor is trained by the user to detect up to 12 distinct hues. The hues can then be used to detect a color signature of two adjacent colors, such as the red / blue plate in Figure 3.3. A license plate was created with dimensions 4 in by 7 in. The license plate was marked with a two-tone color code to reduce the number of false positives.

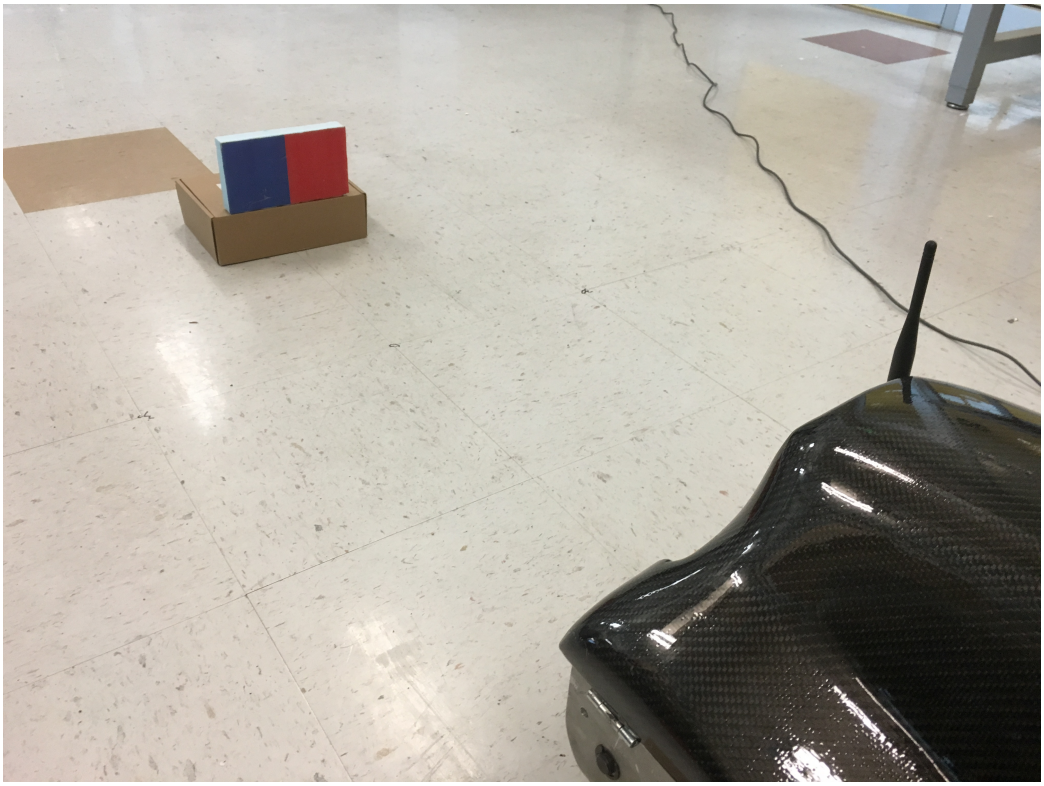

Fig. 3.3: Photo of the color coded license plate at 1 m distance.

3.4.4 Experimental Validation

A Pixy Camera was trained to detect a 2-color block. The block was detected and distances were recorded at various angles (0 rad, $\pi/2$ rad) and distances (0.3 m, 3 m) from the camera. The height, width, and area were utilized to determine the best fit to Equations 3.4 and 3.3, with the parameters for the fit shown in Table 3.2. The resulting curve and sample data points for the geometric fit and exponential fit are shown in Figures 3.5 and 3.6.

Table 3.2: The least squares fit for the parameters of the two distance estimators.

Equation	a	b	c
Geometric 3.3	0.2838	0.0014	N/A
Exponential 3.4	75.52	1.424	6.154

Once the three equations were determined, the accuracy of the algorithms were tested. The mean squared error is presented, shown in Table 3.3. In addition to varying distances, the algorithm was tested for resilience to difference in relative vehicle heading. The mean

squared error is presented for each of the cameras tested, with the number of data points noted in Table 3.3.

Table 3.3: The mean squared error in meters squared for both of the distance estimators. The exponential model outperformed the geometric based model.

Geometric 3.3	Exponential 3.4
0.13	0.09

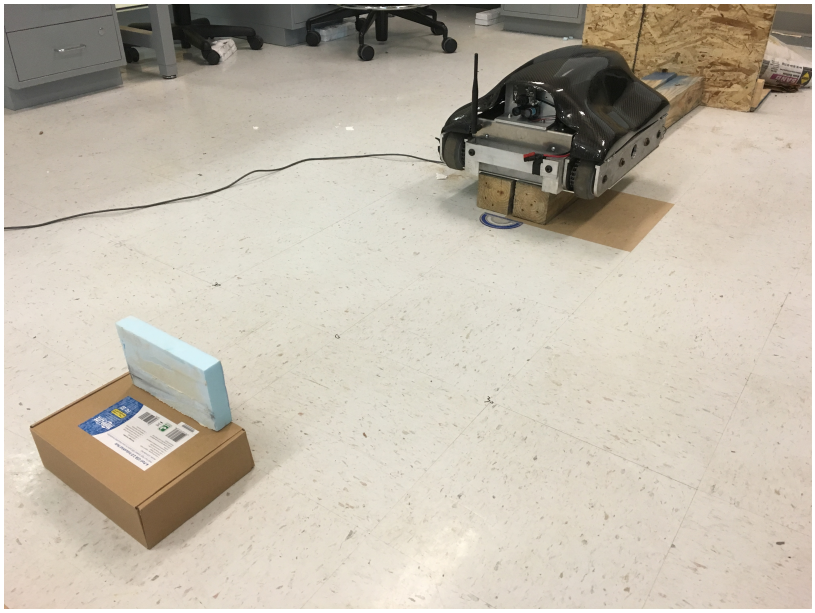

Fig. 3.4: The experimental setup for determining the distance and angle to the license plate. The distances were measured at various angles to account for any fish-eye effect from the camera.

3.5 Conclusion

One of the problems in ground vehicle control is observing the states of the surrounding area. Current professional methods utilize a rotating LIDAR sensor which maps the immediate surroundings of the vehicle hundreds of times a second. While these sensor systems are sub-millimeter accurate, the price of such systems restricts the purchasability for the average consumer.

This section contained a method to determine the angle and distance to a preceding vehicle that can be implemented inexpensively as a redundant sensor. This redundancy

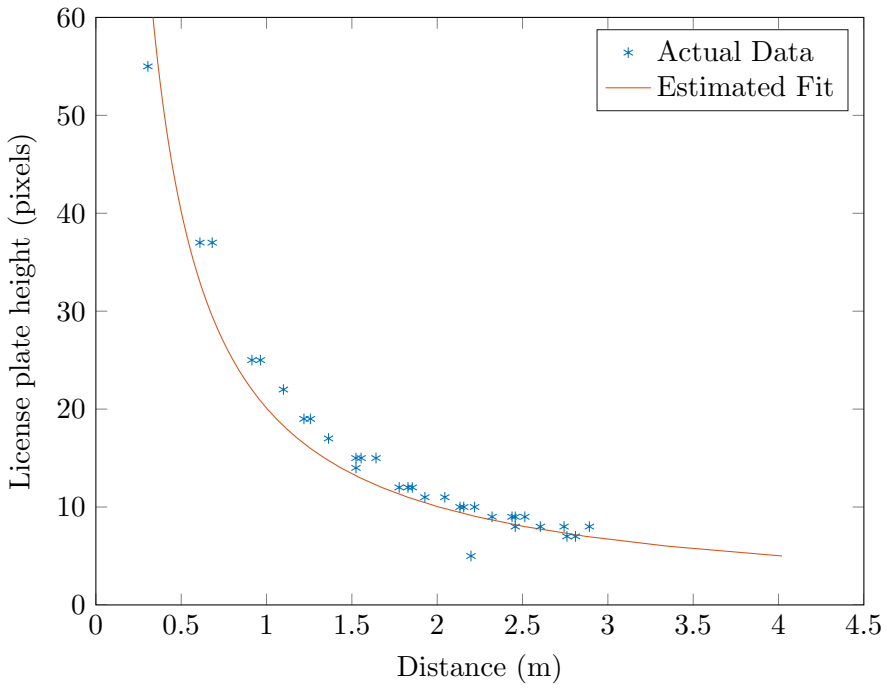

Fig. 3.5: A sample fitting of the geometry-based license plate distance estimator to sample points.

provides increased reliability of a vehicle under adversarial conditions.

A method to estimate the distance to a license plate was presented. The average error of this algorithm is 30 cm, which makes it a poor method for extended periods of control, but much better than no data.

Various license plate detection algorithms were evaluated as candidates for use in the distance estimator. None of the algorithms performed accurately or quickly enough to fit the needs of a vehicle at high speeds. In testing the distance estimator, a colored plate served as a replacement for the license plate.

The proposed distance estimator is effective at providing a low-cost sensor that can operate as a redundant system. The system was used in the SATS Group vehicle test platform as a redundant sensor for the LIDAR distance finder.

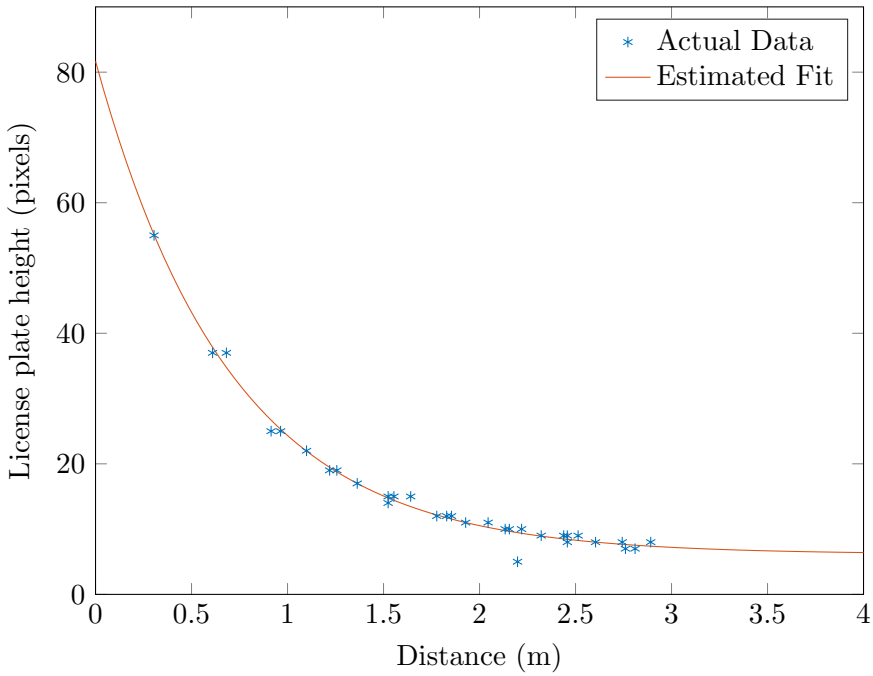

Fig. 3.6: A sample fitting of the exponential license plate distance estimator to sample points.

CHAPTER 4

Experimental Ground Vehicle Platoon Design and Development

The SATS Group has found vulnerabilities to multiple ground vehicle guidance control algorithms as well as methods to protect against attacks. Simulation shows the feasability of proposed methods, but experimental validation on a physical platform is needed to confirm that these methods are not merely academic exercises.

I led the development of a platform with which the researchers SATS Group could validate the results from simulation. The experimental platform consists of automated electrical vehicles that are approximately 1:10 scale of standard highway vehicles. The SATS car system was originally designed by Daniel Dunn and myself. After running a number of experiments on the platform, we saw that more flexibility was required for an upgraded version of the system. This system consists of two major segments, hardware and software. The remainder of Chapter 4 describes the design of the SATS cars.

The following sections describe the design of the hardware and software systems with which to test various control algorithms. Each of the subsystems were developed and tested separately to ensure reliability. Each subsystem was integrated into the prototype once it was operational.

4.1 Hardware

The focus of the SATS Group is to analyze and improve the security of autonomous vehicles, which necessitates the experimental platform to be analogous to full scale highway vehicles. In addition to this requirement, the platform must be able to withstand collisions with other cars at high relative velocity. Due to the second requirement, full-scale vehicles could not be used, so vehicle kits from Battlekits were selected [65].

Two styles of vehicles were purchased, standard and enhanced. The standard configuration corresponds to a nominal highway vehicle. The enhanced configuration uses a 5 in

wheelbase with larger motors instead of the standard 4 in wheelbase. The enhanced configuration allows the attacker to achieve greater velocity and acceleration than the standard vehicle. We consider it a reasonable assumption that an attack vehicle would have greater velocity and acceleration capabilities than a normal highway vehicle, hence the enhanced vehicle configuration. An in-depth description of the hardware is in Appendix A.

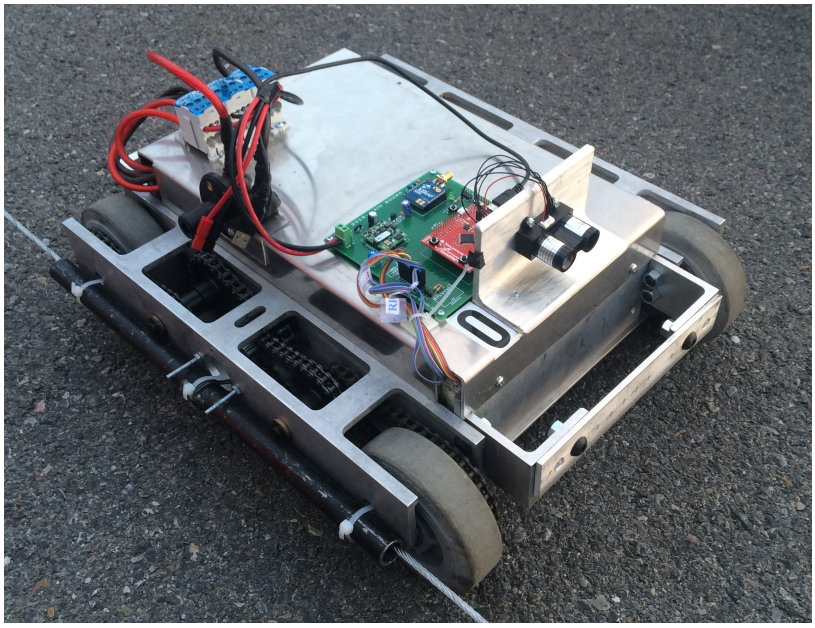

Fig. 4.1: The base SATS car system. The vehicle can travel at speeds up to $10\,\mathrm{m\,s^{-1}}$. In the pictured configuration, the vehicle is guided by a steel cable, which limited the duration of tests.

4.2 Software: Robot Operating System

The experimental platform has one basic purpose, which is to run a variety of lateral and longitudinal controllers with minimal effort to implement a new controller. In addition to this, the system must provide a method to record sensor and control data.

Control systems can be destabilized by sensing or actuation delay [66]. If delay is small enough, compensation can be made, but a variable delay could negate the effects of this compensation. The high level control algorithms in Chapter 2 were analyzed at 10 Hz which proved sufficient resolution to meet stability requirements. These simulations assumed that the system instantaneously responded to the control input, without vehicle dynamics.

Rajamani [32] described a split-controller configuration for usage in automotive applications. This method utilized a high-level controller to command angular velocity and longitudinal velocity / acceleration, and a low level controller to achieve the commanded angular velocity and longitudinal velocity. The low level controller runs at a much higher rate than the high-level controller in order to achieve the commanded velocities. The sensors report data at 50 Hz, which imposes a maximum rate of 50 Hz on the low level controller.

Robot Operating System (ROS) has been proposed as a framework to work with multiple tasks [67]. ROS is middleware written in Python which facilitates message passing and scheduled execution of tasks. The system allows flexibility in hardware and provides a simple interface for logging data, but it does not guarantee system deadlines. Due to the flexibility of the system and simple interface, ROS was selected as the system of choice. We created the *sats_car_ros* package as a collection of tasks to run. The vehicle software system diagram is shown in Figure 4.2.

The ROS system operates on a Beaglebone Black development platform [68] with the Ubuntu distribution of Linux. The processor is an AM335x 1 GHz ARM Cortex-A8 with a NEON floating-point accelerator. The Beaglebone Black communicates with a ground station via a WiFi connection.

4.2.1 Tasks

One purpose of utilizing an RTS is the ability to schedule and execute multiple tasks with high reliability. This section describes the tasks that are required for the system to operate correctly. In ROS, nodes are used to execute software tasks, such as calculating the desired control output or reading from a sensor. This document uses "node" and "task" interchangeably.

High Level Controller

The high level controller accepts information about a target point and provides control commands to reach that target. The high level controller executes code for longitudinal and lateral control. The controllers initially implemented are shown in Equations 4.1 and 4.2.

The acceleration is integrated to determine the command velocity, as the vehicle dynamics model we use accepts linear and angular velocity as commands.

$$\dot{v} = k_p(x_{i+1} - x_i + hv_i) + k_v(v_{i+1} - v_i) \tag{4.1}$$

$$\omega = v\frac{2\sin\theta}{x_{i+1} - x_i} \tag{4.2}$$

In order to determine the desired command output, the high level controller gathers the required information from the sensor hub and Pixy node. The lead vehicle follows a predefined GPS path, as described in Section 4.2.2.

The high level controller runs at a rate of 10 Hz. The velocity and angular velocity commands were originally sent to the low level controller via a message, but passing the message to the other node increased system delay. In order to reduce delay, the high and low level controllers were combined into one node, with the high level running once for every five executions of the low level controller. We do not increase the frequency of the high level controller due to the computationally intensive path calculation from Section 4.2.2.

Low Level Controller

The low level controller accepts control commands from the high level controller and calculates the required PWM output to send to each motor. The current control algorithm is a PID velocity controller, as shown in Equation 4.4.

$$e[t] = v^c - v[t] \tag{4.3}$$

$$u = k_p(e[t] - e[t-1]) + k_i e[t]\Delta t + \frac{k_d}{\Delta t}\left(e[t] - 2e[t-1] + e[t-2]\right) \tag{4.4}$$

The low level controller writes the calculated PWM value to the file register where the PWM duty cycle command is read (/sys/class/pwm/pwmchip0/pwmX/duty_cycle). The write was originally causing excessive delay (2 s because the file was being opened with a large buffer size. This issue was resolved by setting the file buffer size to 0, which caused the file to be written immediately upon receiving the command.

Sensor Node

The sensor node provides communication with the sensor hub (Tiva C), verifies the data, and passes the information to the high and low level controllers. The sensor node has the capability to read two types of messages from the sensor hub: sensors only or sensors with GPS.

The sensor hub transmits a message with three or four sections: start code 1 B, sensor message 24 B, optional GPS message 52 B, and cyclic redundancy check (CRC) 2 B. The CRC is calculated on the first 25 B or 79 B of the message, using the CRC-CCITT (XModem) method.

When the sensor node receives a correct message from the sensor hub, it relays the message to the controller node via a ROS message queue (topic). Upon receipt of an erroneous message, the sensor node discards the message and flushes the UART buffer. This method is used to prevent the system from using old data.

Image Processing

The original plan for the image processing system was to compute information about the distance to the target at the high level controller rate 10 Hz. The Beaglebone Black proved to have insufficient resources to perform this task, so a Pixy Camera was purchased. The Pixy Camera has an on-board processor which detects color-based candidate regions and reports the location and size of the candidate region at 50 Hz.

A ROS package for the Pixy Camera is available on github.com. The message that the pixy_node package transmitted had a variable length, up to 32 kB, and did not meet our specifications. Due to this, a team member reconfigured the pixy_node package to transmit the parameters of interest of the largest detected color code.

The image processing node reads information from the Pixy Camera, packages the information and sends it to the high level controller.

4.2.2 Design Challenges

The development of the system came with challenges. This subsection describes some of

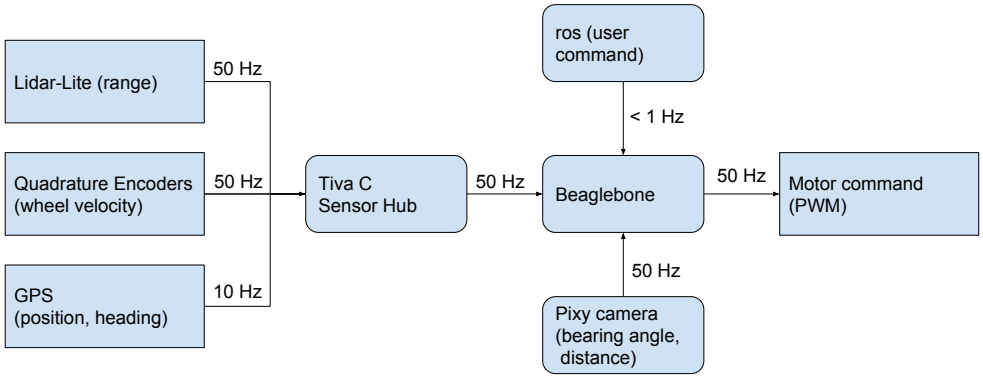

Fig. 4.2: Sensor and data flow of the SATS Group experimental vehicle platform. The Tiva C serves as a sensor hub and passes the data to the Beaglebone, which runs the *sats_car_ros* package. The Beaglebone writes the motor commands directly.

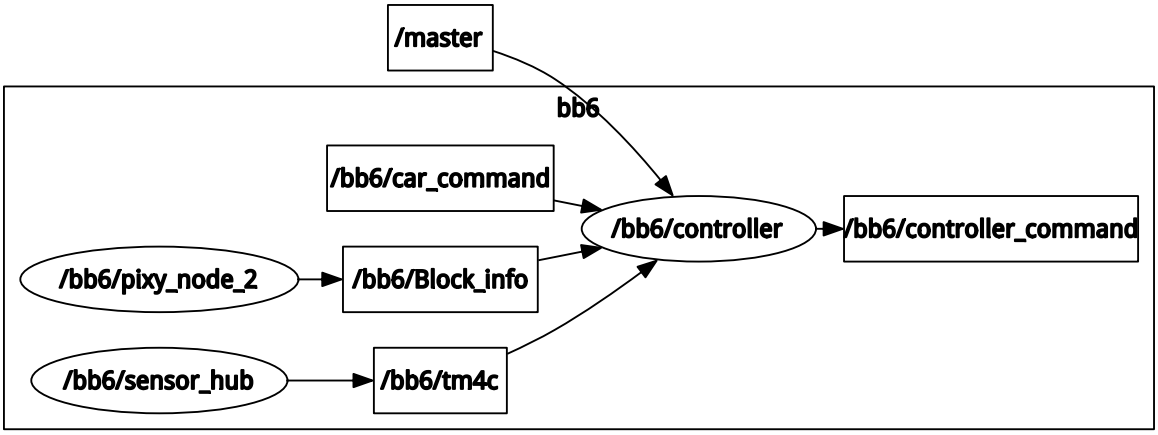

Fig. 4.3: The interconnection of the nodes on the SATS Cars. Each vehicle operates within its own namespace (e.g. /bb1), so each vehicle is effectively independent from the others. The ovals denote nodes (tasks), and the rectangles denote topics (message queues). The pixy_node_2, sensor_hub, and controller nodes run at 30 Hz, 50 Hz, and 50 Hz respectively.

the problems encountered and solutions developed during the development of the *sats_car_ros* package.

Emergency Shutdown

One safety requirement for the vehicle system was a master kill switch to stop all of the vehicles. To accomplish this task, each of the vehicles subscribed to a topic "/master". This master topic contains a bool to set the vehicles to on or off. When the master switch is off, the vehicle will calculate control gains but will apply a PWM value that turns off the motors.

Another safety issue appeared while programming the controller node. If the controller node encountered an error and exited, the motors would maintain their most recent velocity command. ROS contains the functionality to run commands when the node exits, so the PWM values were set to stationary upon shutdown. If the user halts ROS execution, the vehicle safely stops.

When the wireless connection is lost, the vehicle continues to operate without feedback from the ground station. A watchdog timer which monitors the wireless connection should be incorporated in a future release of the controller.

Path Selection

The following vehicles within the platoon utilize image processing to determine its desired path. The leader of the platoon determines its own path via following a GPS trajectory. Two methods to do so were presented: (1) playing a ghost leader to the platoon at a set velocity, (2) calculating the nearest point to the desired path then looking ahead a set distance.

Method 1 was implemented first due to the simplicity of the method. The position data of a vehicle driving around the test track was recorded. After the data was recorded, the lead vehicle subscribed to the position of the ghost leader and utilized that information to calculate the distance and angle for the high level controller. When this was tested on a physical system, the GPS position error was large enough to saturate the velocity of the

lead vehicle. If the lead vehicle was set to a static velocity, the ghost leader could very easily drive too slowly or too quickly, which would cause the leader to drive off the track. This method was deemed impractical for our application.

The guidance method for the lead vehicle was performed by calculating the nearest point to the track. The 400 m track was sampled at 1000 equally spaced points. The lead vehicle calculates the closest point, then looks ahead 6 m. The proper way to calculate the closest point to the track is to utilize the Haversine formula, which calculates the distance between two points on a sphere. This formula is not extremely complex, but it is too computationally expensive to compute at 10 kHz on the Beaglebone Black. For small differences in GPS coordinates, the meters per degree latitude and longitude is roughly constant, so the distance was calculated by scaling the difference in latitude and longitude points by the appropriate constants.

Calculating the closest point using the Euclidian distance sometimes returned points behind the vehicle, so the Manhattan distance metric was utilized to determine the closest point, as shown in Equation 4.5.

$$\min_i |p_{lat}^{self} - p_{lat}^{path}[i]| + |p_{long}^{self} - p_{long}^{path}[i]| \tag{4.5}$$

After the closest point was selected, the list was traversed to find an appropriate look ahead point. A look ahead distance of 6 m provided stability when the vehicle was traveling at $1\,\mathrm{m\,s^{-1}}$. Further work could be done to determine an appropriate velocity-dependent look ahead distance.

When implemented on the Beaglebone Black, finding the closest point executed in roughly $1\,\mathrm{m\,s^{-1}}$.

System Delay

A major issue in getting the system to work correctly was system delay. The ROS nodes are shown in Figure 4.3. When testing the algorithm, the high level task, low level task, and sensor hub task were not running at the desired rates. The basic task timing

analysis showed that the nodes should not overrun their execution times. The task timing was analyzed again, and the high level node took 50 ms to execute, instead of 2 ms. The offending function was located through further timing analysis, and the problem turned out to be a generic logging function. The conversion of the input to a string took much longer than expected. Due to this discovery, the generic logging function is used only in operations that are not time sensitive.

4.3 Command Station

A command station was needed to coordinate the operation of each vehicle and to visualize the data obtained from the vehicles. The command station is a MacBook Pro running Ubuntu Linux connected to a wireless access point. Each vehicle connects to the command station via the access point, as seen in Figure 4.4.

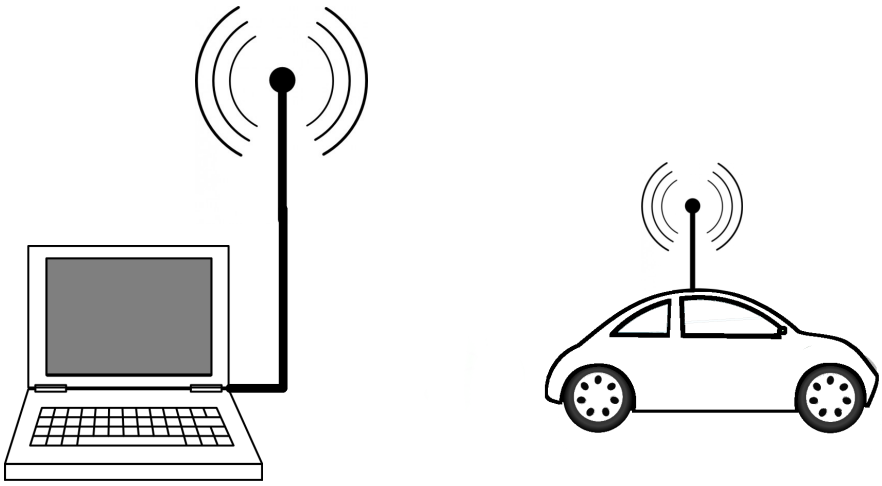

Fig. 4.4: The command station configuration. The laptop sends commands to each vehicle, collects data from the vehicles, and visualizes the data via RQT.

Each of the vehicles perform controller calculations on-board the vehicle. The vehicles receive configuration commands from a base station. This section describes the operation of the base station and connectivity challenges.

4.3.1 ROS Master

ROS requires one computer to coordinate communication between nodes. In order to operate correctly, the command station must be able to communicate with the target ip addresses without password verification. The hostname must also be a registered target on the network.

The ROS master executes a launch script which logs in to each vehicle and starts each node. The nodes obtain parameters, such as loop rate or control gains, from the master. When a node attempts to communicate via a message queue (topic), it obtains the address of the publisher(s) from the ROS master, then proceeds to communicate with the other nodes.

Two methods can be used to send control messages to the vehicles. RQT is a graphical interface that allows the user to create and publish messages [69]. Testing and development using RQT is sufficient when testing individual vehicles, but one message configuration can't be published to multiple topics, i.e. 3 vehicles operating in the same mode would require 3 separate configuration messages.

A web interface for the *sats_car_ros* package was developed by Gregory Vernon, shown in Figure 4.5. The web interface utilizes rosbridge to provide function handles for sending messages. The new interface allows the user to define one configuration and send it to an arbitrary number of vehicles. The web server is hosted on the ROS master.

4.3.2 WiFi Access Point

When working with the vehicles at the test track, we were originally using 802-11g wireless adapters with 7 dB antennas attached, but the signal strength was abysmal. On the South end of the test track, the wireless adapter received a 20 % signal strength at 8 m. At the North end of the test track, the system received better connectivity: 34 %.

802-11n wireless adapters were purchased to improve signal strength through beamforming. When the vehicles sat stationary, the signal strength was measured to be 85 %, but when the vehicles were in motion, the signal strength dropped to 35 %. We understood this

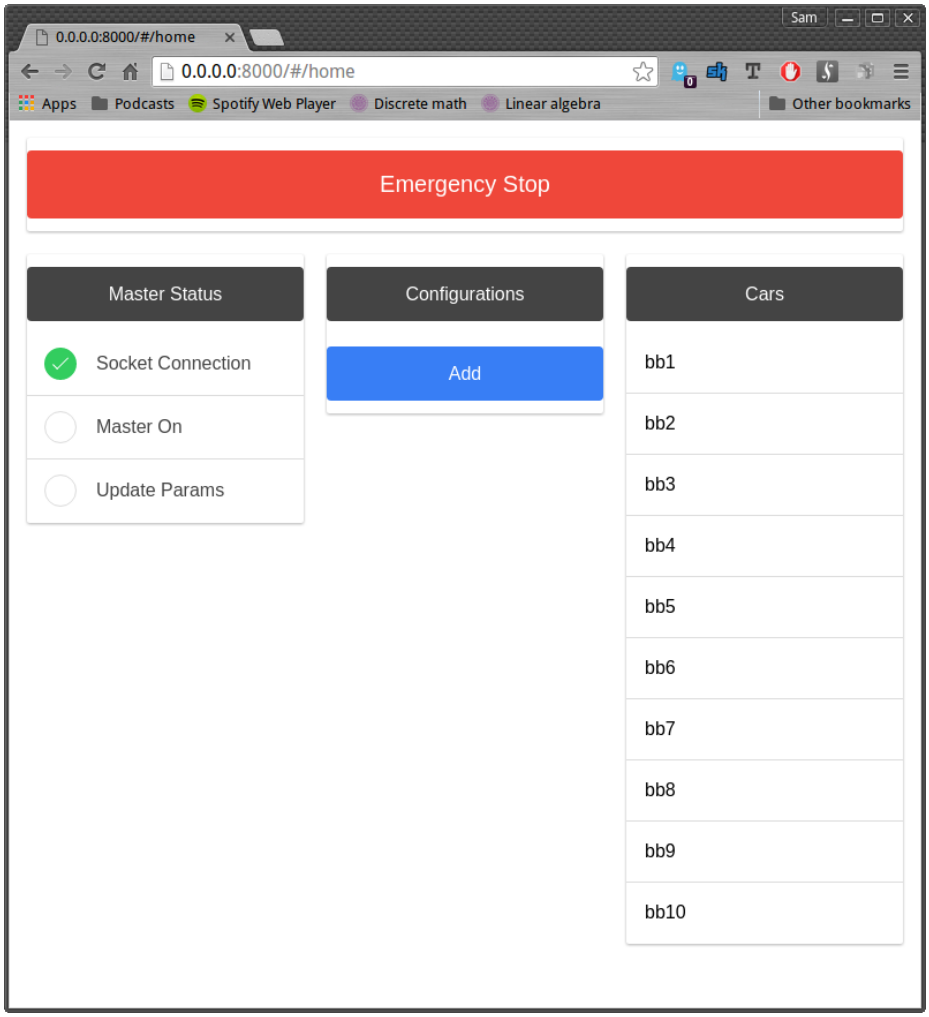

Fig. 4.5: The web interface allows one configuration to be applied to all vehicles. The emergency stop button shuts down all vehicles simultaneously.

to mean that the beamforming technology did not adapt to the changing channel sufficiently quickly for the vehicles in motion.

In the debugging process, the number of active wireless cells were determined to be 29, with 26 of those cells operating at 2.4 GHz. The system was reconfigured to operate solely in the 5 GHz band, but beamforming was disabled. After these changes, the wireless adapters read values of 92 % signal strength at 10 m distance.

4.4 Discussion

The *sats_car_ros* package was created to be a simple interface for the SATS Group scale

vehicles. Controllers that have been simulated in another system are easily implemented within the controller nodes. The sensor and control data from each vehicle is reliably reported to the ground station, where the data can be visualized real-time with the RQT interface. The vehicle system currently has three points of failure which are of concern, all are hardware-related.

The motor electrical system and the electronics power system are not isolated from each other. This means that when the battery voltage is slightly low or the motors draw a large amount of current, the Beaglebone shuts down.

The Beaglebone Black GPIO pins are extremely sensitive to electrical variation. If the PWM pins are misaligned on the PCB, the Beaglebone Black circuits are damaged. Four of the Beaglebone Blacks have been burnt out due to this design issue. The system is not designed wrong, but the PCB design should be modified to provide protection for the Beaglebone. The modification would limit the current flow of the PWM pins.

The Pixy camera is effective at detecting colored objects. Computer vision is very bad at determining color when ambient lighting changes. Due to this, the license plate detection scheme used by the Pixy is unreliable outdoors. This is not a flaw in the Pixy camera — a different vehicle detection algorithm should be used.

The vehicle platform was developed as a testbed for evaluating the security of autonomous ground vehicles. The platform fulfills the requirement to provide a testing method.

CHAPTER 5

Attacker-Induced Traffic Flow Instability in a Stream of Automated Vehicles

5.1 Introduction

The two major benefits that follow a transition to an Automated Highway System are human safety and efficient use of the current infrastructure. The American highway system will be expensive to expand with the growing population, and vehicle automation is one potential solution to the overcrowding of roads. Automated vehicles can reliably travel at high speeds with small distances between vehicles, which provides increased throughput on the highways.

An important aspect of determining the reliability of an AHS is finding system vulnerabilities. Dunn [33] proposed a method to incite string instability in an automated highway system. String stability is a measure proposed by Yanakiev [11] that determines if coupled vehicles will decrease position error. If a system of vehicles is string stable, the position error will be attenuated, which in turn decreases velocity oscillation. Yanakiev shows that string stability is required for a system of automated vehicles to function without a drastic reduction in throughput.

The attack proposed by Dunn was to inject marginally stable control gains into 8% of the vehicles (passive attackers), followed by a brief system disturbance. The disturbance caused the passive attackers to begin oscillating. Normal vehicles attenuated the oscillations, but attacker density caused the oscillations to be amplified before they could be damped out completely. Figure 5.1 shows the amplitude reduction of an oscillation with the vehicles in nominal operation and the attack vehicles using stable control gains, and Figure 5.2 shows the effects of an attack with passive attackers using unstable control gains.

Dunn showed that the attack is effective against five control algorithms used in vehicle platooning or automated highway systems.

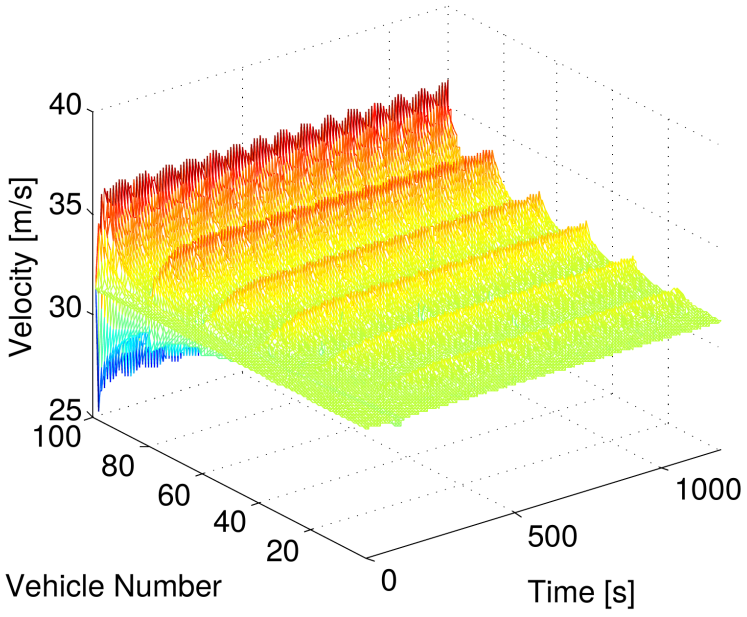

Fig. 5.1: The results of the Dunn attack with passive attackers with stable gains. Note the peaks in the velocity at the location of the passive attacker. The nominal vehicles dampen the oscillatory response of the system.

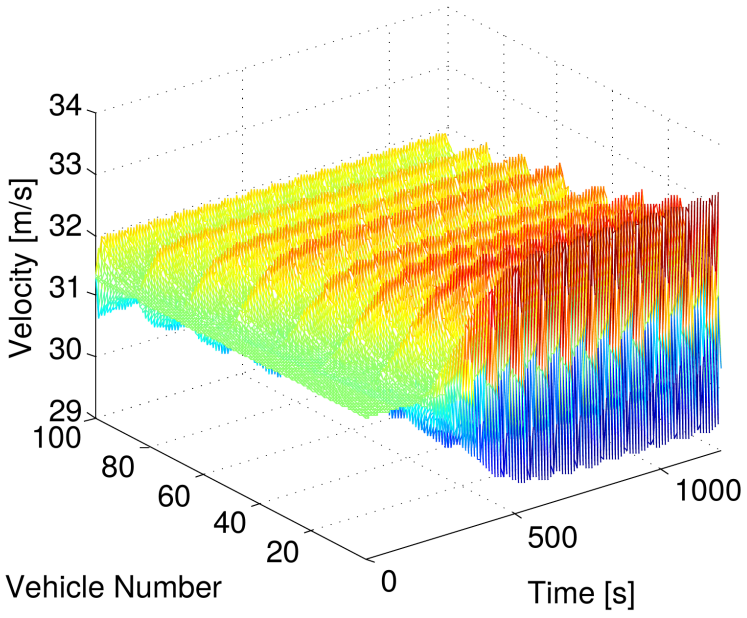

Fig. 5.2: The results of the Dunn attack with passive attackers utilizing unstable control gains. The velocity oscillation was amplified through the system.

The remainder of the chapter is divided into two sections: an analysis of a potential method to mitigate the effects of the attack, and a description of the experimental validation of the attack. The experimental vehicle platform presented in Section 4 was developed to test the efficacy of attacks such as Dunn's.

5.2 System Stabilization in the Presence of an Attack

An interesting aspect of the attack is that it is ineffective against human drivers [33], because humans tend to reduce velocity if the preceding vehicle behaves strangely. This finding was based on utilizing the intelligent driver model (IDM) as a model of human driving behavior. The IDM is an empirically validated car-following model that captures the features of traffic flow in freeway and urban environments, particularly for congested systems [70].

The attack presented by Dunn was ineffective against human drivers. This led to the idea that one method to combat the instability caused by the attack was to introduce human drivers (IDM) into an automated system as a stabilizing force. The drawback to this method is that the IDM decreases highway throughput. This postulate was simulated across a range of attack, IDM, and standard automated vehicles.

The attack was simulated against a 100-vehicle system utilizing the control algorithms from [33]. A collision avoidance system was implemented in order to more accurately model vehicle behavior — vehicle manufacturers are likely to employ an avoidance system that rapidly decelerates if a collision is imminent. The results of the attack against Control Algorithm 1 are shown in Figure 5.2. The attack resulted in a traffic jam where each vehicle rapidly accelerated to the maximum velocity and braked to a halt.

5.3 Experimental Validation of Attack

To perform experimental validation we developed a testing platform consisting of 1/10 scale autonomous vehicles, as described in Chapter 4. The objective of this platform is to provide a means of testing theoretical vehicle automation on a physical system. The vehicles were differential steer robots driven by DC electric motors and capable of top velocities of

$10\,\mathrm{m\,s^{-1}}$ (22 mph). Quadrature encoders were used to determine vehicle velocity. Lidar range finders were used to measure relative distance between vehicles. The quadrature encoder and velocity data were numerically differentiated using the method of [71]. This platform provides a relatively low cost method of testing control schemes compared to full scale vehicle implementation allowing a variety of control methods to be easily verified. The vehicles are guided along a 400m track using a tensioned cable (Figure 5.5).

The control algorithms were implemented discretely on a micro-controller aboard each vehicle. To mask non-linearities a split-level control architecture was employed: algorithms 1 to 5 provided high-level control input (e.g. target acceleration), while a low-level controller (adapted from [6,32]) was utilized to convert this control input to a motor voltage and ensure that the vehicle maintained the commanded input. Position and velocity data were collected from each vehicle at 4 Hz. As our vehicles lack accurate accelerometers, the numerical integration required for algorithms 4 and 5 introduced instability, so only algorithms 1–3 were used for experimentation.

5.3.1 String Stability Verification

The 9 vehicle platoon was tested to determine how accurately the vehicles realized Control Algorithms 1–3. The vehicles are numbered 9 to 1, with 9 being the lead vehicle. The test trajectory was the lead vehicle (9) accelerated from $0\,\mathrm{m\,s^{-1}}$ to $2.4\,\mathrm{m\,s^{-1}}$. This step acceleration provided sufficient frequency information to verify the string stability of the string of vehicles.

Gains were selected as recommended by the authors of the respective papers. The selected stable gains are shown in Table 5.1. The controllers referenced in Table 5.1 correspond the the algorithms stated in Table I. The vehicle string was tested as outlined, and the oscillations induced by the step input were dampened out upstream. The velocities of each vehicle are shown in Figure 5.3. Variations in vehicle velocity were due to sensor and environmental noise.

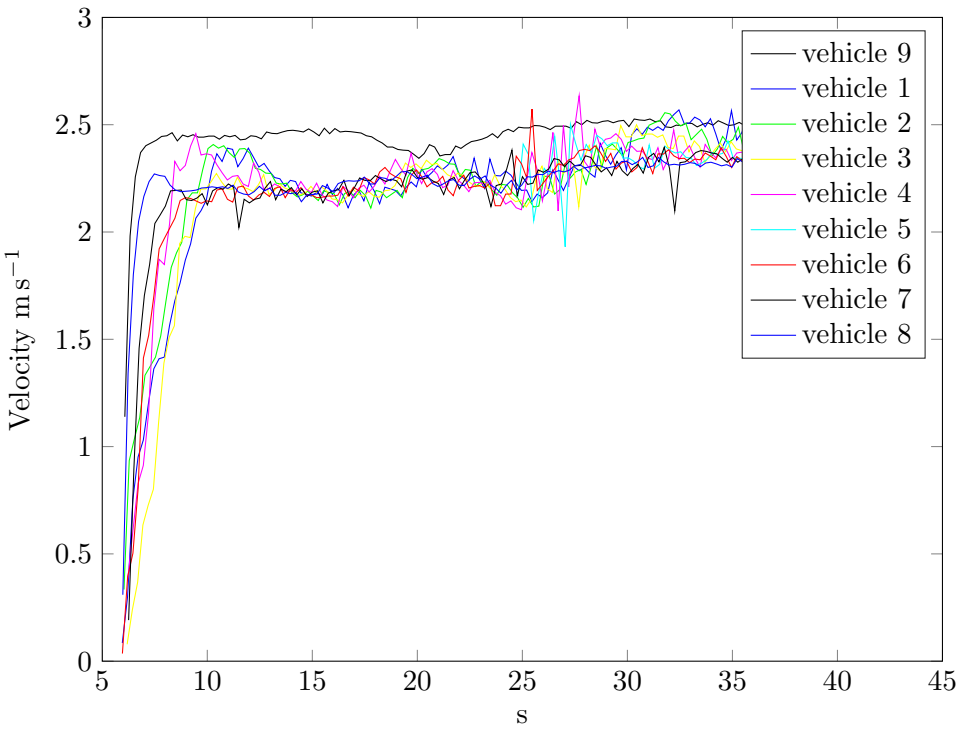

Fig. 5.3: Stable platoon operation of control algorithm 1.

5.3.2 Destabilization Attack Verification

The test trajectory to demonstrate the string destabilization attack was the lead vehicle (9) accelerated from $0\,\mathrm{m\,s^{-1}}$ to $2.4\,\mathrm{m\,s^{-1}}$. One follower (8) was placed between the leader and attacker. The attacker (7) utilized the string unstable gains from Control Algorithm 1. These string unstable gains were computed using the method described in [33], $k_d \leq -k_p h$, resulting in the gains from Table 5.1. Subsequent followers maintained stable gains. The victim vehicles utilized string stable gains.

The result of the attack is shown in Figure 5.6. The passive attacker amplified the oscillations from the step input. The value in this attack is the passive attacker induced oscillations in other vehicles without compromising its own safety. Each control algorithm is susceptible to the attack. Further testing was performed by following the prior test trajectory, modified by inserting a second passive attacker as the rear vehicle (1). Due to the high density of attackers, the system became string unstable, causing collisions. The testing was sufficient to demonstrate the destabilization of a string stable system by

maliciously selecting controller gains in a percentage.

To perform experimental validation we developed a testing platform consisting of 1/10 scale autonomous vehicles. The objective of this platform is to provide a means of testing theoretical vehicle automation on a physical system. The vehicles were differential steer robots driven by DC electric motors and capable of top velocities of $10\,\mathrm{m\,s^{-1}}$ (22 mph). Quadrature encoders were used to determine vehicle velocity. Lidar range finders were used to measure relative distance between vehicles. The quadrature encoder and velocity data were numerically differentiated using the method of [71]. This platform provides a relatively low cost method of testing control schemes compared to full scale vehicle implementation allowing a variety of control methods to be easily verified. A further description of the platform is described in Section 4. As these tests were exclusively concerned with longitudinal control, the vehicles were guided along a 400m track using a tensioned cable (Figure 5.5).

This testing was performed before the system in Section 4 was fully developed. The control algorithms were implemented discretely on a micro-controller aboard each vehicle. To mask non-linearities a split-level control architecture was employed: algorithms 1 to 5 provided high-level control input (e.g. target acceleration), while a low-level controller (adapted from [6,32]) was utilized to convert this control input to a motor voltage and ensure that the vehicle maintained the commanded input. Position and velocity data were collected from each vehicle at 4 Hz. As our vehicles lack accurate accelerometers, the numerical integration required for algorithms 4 and 5 introduced instability, so only algorithms 1–3 were experimented with.

5.3.3 Attack Measurement

One important aspect of measuring efficacy of the stabilizing efforts was to determine what constituted a successful attack, and finding a metric to quantitatively measure the impact of the attack.

An effective attack increases the velocity oscillation amplitude through the system. The standard deviation of the velocity of all vehicles was increased substantially. In terms of throughput, the greatest benefit from the transition to automated vehicles is the increased

Table 5.1: Gains used in experimental validation for Control Algorithms 1–3.

Control Algorithm	k_p	k_d	h	k_h	v_d	L
1	1	2	0.5	-	-	0.5
2	1	1	0.55	0.1	2	0.5
3	1	1	1	1.75	2	1
Atk (1)	3	-1	0.5	-	-	0.5

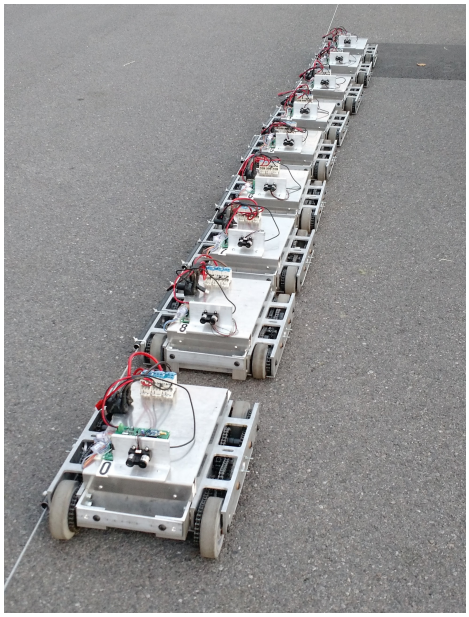

Fig. 5.4: The platoon in strictly longitudinal operation.

Fig. 5.5: The testing platform of 1/10 scale vehicles. Experimental setup of platoon when operating in strictly longitudinal operation. The cable was utilized to guide the vehicles before lateral guidance was functional.

highway throughput. The attack is measured with a comparison of the system velocity standard deviation and the current highway throughput. With human drivers, a highway with vehicles traveling at $33\,\mathrm{m\,s^{-1}}$ ($120\,\mathrm{km\,h^{-1}}$) can safely fit 17–28 vehicles/km.

5.4 Simulation Results

The percentage of attackers and IDM vehicles was varied from 5% to 50% in 5% increments. A 100-run Monte Carlo simulation was performed, varying the control values of attack and standard algorithms. The velocity standard deviation and highway utilization were evaluated for each simulation. The mean of the 100 simulation values is shown in Figures 5.9 and 5.10.

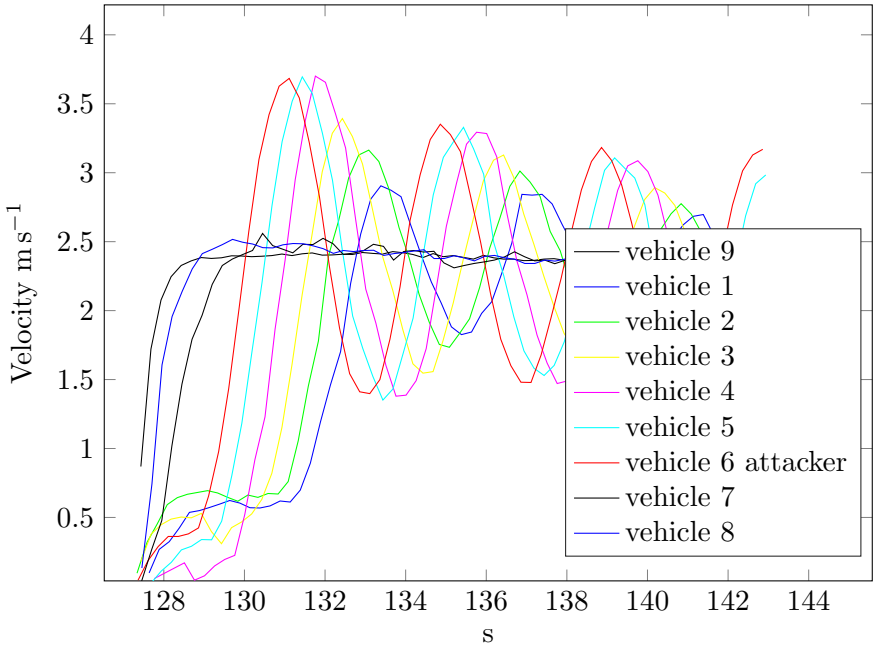

Fig. 5.6: A physical system of 9 vehicles using Control Algorithm 1 with active attacker at location 6.

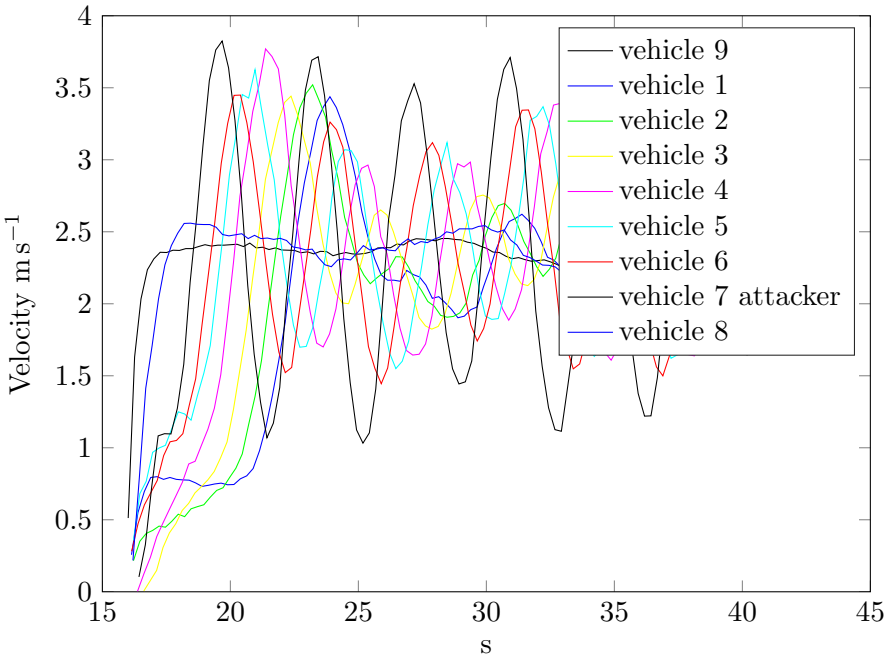

Fig. 5.7: A physical system of 9 vehicles using Control Algorithm 2 with active attacker at location 7.

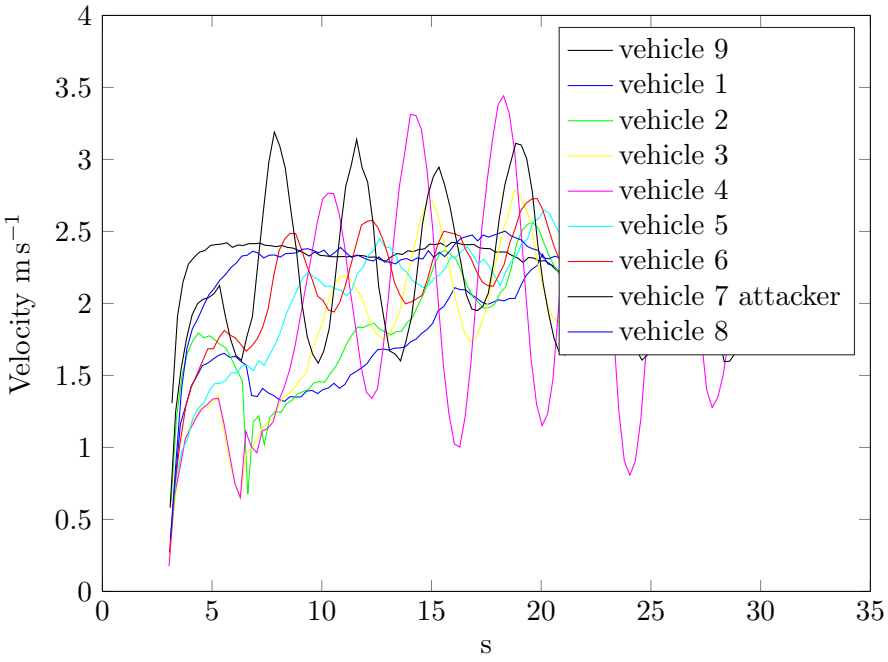

Fig. 5.8: A physical system of 9 vehicles using Control Algorithm 3 with active attacker at location 7.

This protection method was tested in a 100 vehicle platoon with 8% attacker and 8% human driver density simulated for 20 minutes. As can be seen from Figures 5.9 and 5.10, the system is marginally stabilized by an increase of IDM vehicles, at a cost of highway utilization. The efficacy of the IDM stabilization is marginal due to the throughput reduction.

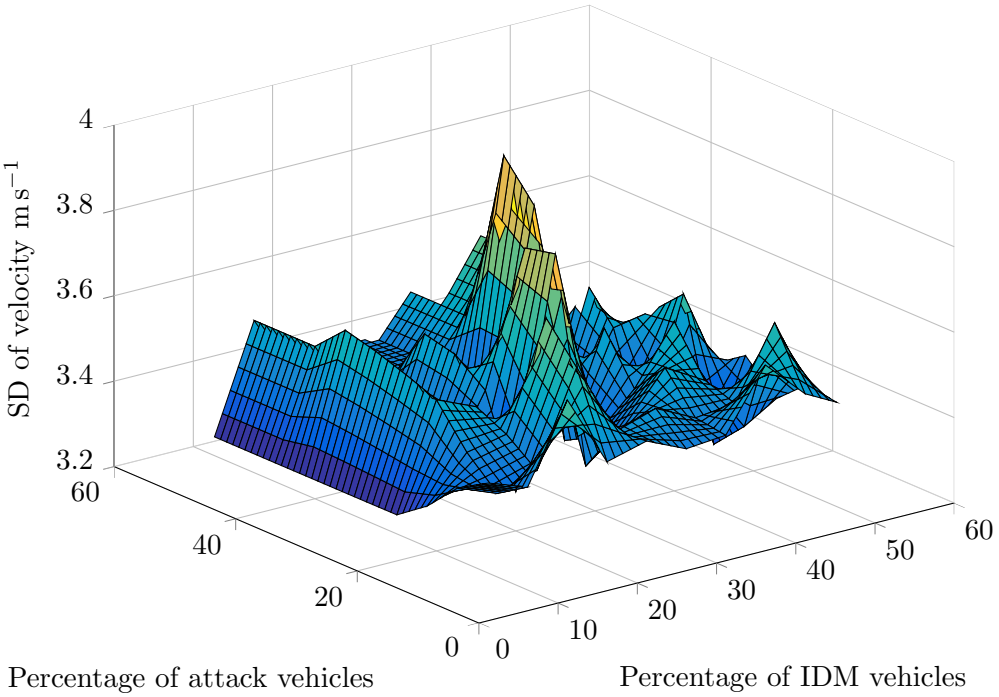

Fig. 5.9: System velocity standard deviation a function of the percentage of IDM and Attack Vehicles.

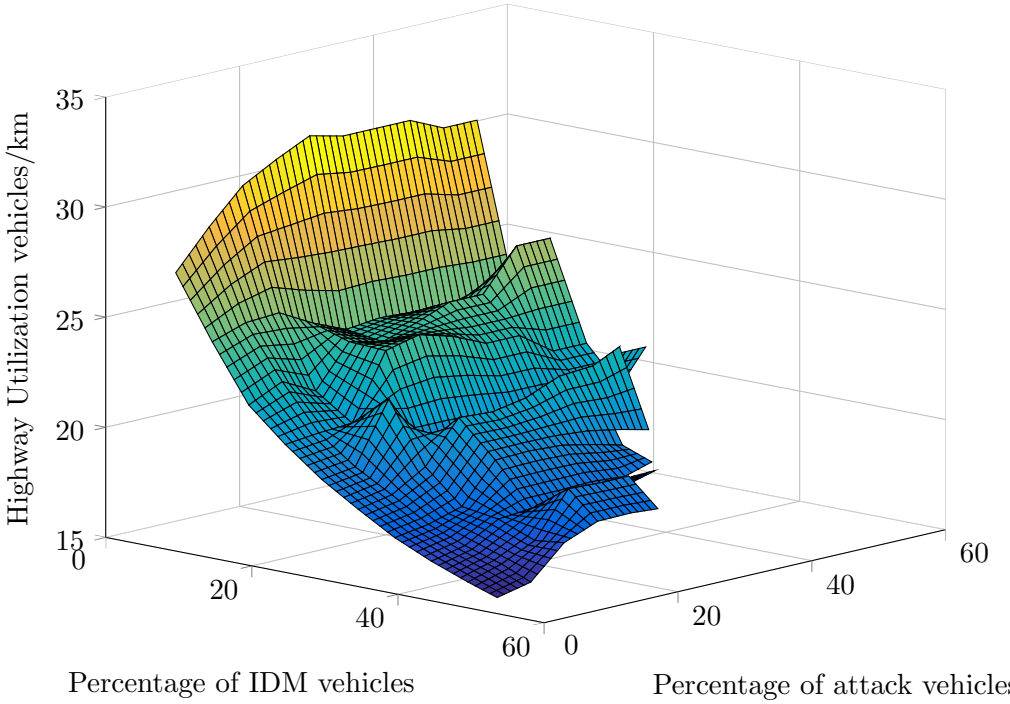

Fig. 5.10: Highway utilization as a function of the percentage of IDM and Attack Vehicles.

CHAPTER 6

Experimental Validation of Vehicle System

6.1 Introduction

The experimental validation of the vehicle platform is the capstone of this work. The controls work, platform design and implementation, and state estimation all must function reliably for the vehicle platform to drive around the test track.

Simulations were performed prior to physical implementation of the control algorithms used. An accurate simulation can mitigate the chance of damage to the test platform. The controller implementation was separated into three stages: low level characterization, high level simulation, and high level implementation.

6.2 Low Level Controller Characterization

The lateral and longitudinal controllers selected for use were presented in Chapter 2. The high level controller outputs needed to be aligned before the low level controller could be designed and calibrated. The longitudinal controller commanded a desired acceleration, while the lateral controller utilized desired velocity as a state. Alignment occurred by integrating the desired acceleration in the high level controller at a rate of 10 Hz.

The purpose of the low level controller is to achieve velocity and steering commands from the high level controller. The steering controller determined a desired velocity for each motor based on vehicle geometry. The motor controller gave a PWM value between -1 and 1 to the motor controller board. This PWM value corresponds to -36 V and 36 V, respectively.

6.2.1 Motor PID Controller

Motor control was performed using a discretized PI (proportional integral) controller [72].

The velocity error e, or difference between desired velocity v_d and current velocity v is the parameter to be minimized, as seen in Equation 6.1. In a PI controller, the integral term removes steady state error, while the proportional term responds more quickly to input. The proportional and integral gains, k_p and k_i are selected to reduce rise time and settling time. The control input is calculated from the old control input $u[k-1]$, proportional error, and integral error which is calculated using a numerical integration method, as seen in Equation 6.2.

$$e[k] = v_d[k] - v[k] \tag{6.1}$$

$$u[k] = u[k-1] + k_p\left(e[k] - e[k-1]\right) + \frac{k_i}{T}\left(e[k] - 2e[k-1] + e[k-2]\right) \tag{6.2}$$

Team members tuned the PI gains via simulation, using motor parameters that were obtained via characterization of the motors on board. Once the desired response was obtained, the controller was tested on the experimental platform. Further tuning was performed, resulting in the gains from Table 6.1.

6.2.2 Low Level Steering Controller

The vehicle model used was for a differential drive vehicle with one wheel on each side, seen in Equation 6.3. The experimental platform has two wheels on each side, and the drag of the extra wheel causes the model to work imperfectly.

$$\begin{aligned} v_L &= v_d\left(1 + \kappa\frac{L}{2}\right) \\ v_R &= v_d\left(1 - \kappa\frac{L}{2}\right) \end{aligned} \tag{6.3}$$

The turn accuracy was tested on the test track, and we found that the drag caused by the extra wheel could be accounted for by modeling the vehicle as having a wider wheelbase

Table 6.1: The gains for the PI motor controller which achieved the fastest settling time with minimal overshoot.

Gain parameter	Value
k_p	0.5
k_i	0.3

L. The vehicle was sent the command to drive in circles of 1 m, 2 m, and 3 m, with the wheelbase coefficient adjusted to achieve the desired radius. The resulting vehicle width was increased by a factor of 2, as shown in Equation 6.4.

$$\begin{aligned} v_L &= v_d\,(1 + \kappa L) \\ v_R &= v_d\,(1 - \kappa L) \end{aligned} \tag{6.4}$$

6.3 Simulation of Driving on Test Track

The simulation used to determine which controller to use neglected the motor dynamics of a differential drive vehicle. The simulation limited the maximum acceleration of the vehicle, but it did not incorporate limitations on the turning capabilities of the car. A simulator was developed to test the vehicle dynamics while following GPS waypoints, as described in Section 4.2.2. The simulator can be found in Appendix C.

Motor parameters were selected from a previously characterized vehicle. The low level controller utilized the gains in Table 6.1. The high level longitudinal gains utilized are shown in Table 6.2. The lead vehicle used a fixed velocity controller; the desired velocity was sent directly to the low level controller.

6.3.1 Simulation of a Single Vehicle

The observed vehicle position was meant to replicate the GPS signal received by the lead vehicle. The VN-200 position resolution was not given, but our tests revealed that it is ± 1 m. Gaussian noise with zero mean and standard deviation of 1 was added to the sensor measurement.

Table 6.2: The gains for the high level longitudinal controller.

Parameter	Value
k_p	0.75
k_d	1
time headway h	1 s
nominal distance d_0	1 m

Path error was the parameter to minimize. Figure 6.1 shows the path error of one sample run. The lateral stability of the vehicle is contingent on a combination of velocity and lookahead distance. The vehicle is expected to travel at speeds up to $10\,\mathrm{m\,s^{-1}}$, with nominal speeds between $1\,\mathrm{m\,s^{-1}}$ and $5\,\mathrm{m\,s^{-1}}$. In simulation, the vehicle velocity was incremented from $1\,\mathrm{m\,s^{-1}}$ to $5\,\mathrm{m\,s^{-1}}$ with a step size of $0.5\,\mathrm{m\,s^{-1}}$. The lookahead distance was varied between 0 m and 7 m.

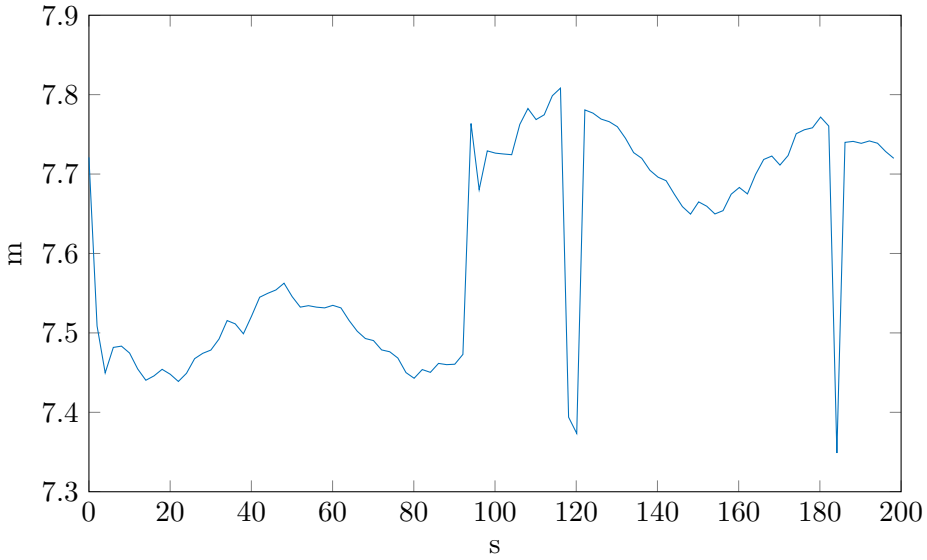

Fig. 6.1: Path error for the single vehicle following a path of GPS points when using a desired velocity of $3\,\mathrm{m\,s^{-1}}$ and lookahead distance of 19 for the leader, with the follower using high level gains in Table 6.2

When the vehicle followed at too close a distance, lateral control went unstable, causing unbounded oscillations. A long lookahead distance reduced the vehicle's ability to react to the curves of the test track. For following speeds of less than $5\,\mathrm{m\,s^{-1}}$, a lookahead distance between 5.5 m and 10 m prevented the vehicle from oscillating too wildly, with the best results occurring at 7.2 m lookahead distance for $3\,\mathrm{m\,s^{-1}}$.

6.3.2 Simulation of a 2-Vehicle Platoon

The purpose of the experiment is to determine the ideal controller gains for vehicle following, so instead of simulating license plate detection, the following vehicle directly detects the position of the preceding vehicle. Despite this, in order for the license plate

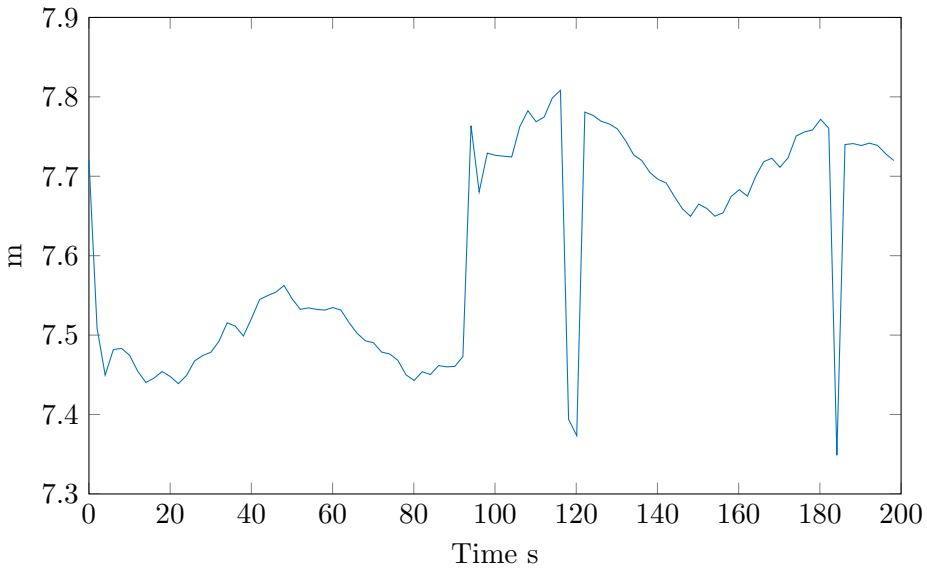

Fig. 6.2: Longitudinal distance to the target for the vehicle following a GPS path.

detection method to work with the current hardware, the following vehicle must maintain a distance of 0.5 m to 3 m. The single vehicle following showed that a following distance smaller than 5 m could cause lateral instability.

The operation of the vehicle at 3 m following distance caused the vehicle to oscillate opposite the lead vehicle. These oscillations could cause the camera to lose track of the preceding vehicle. Further simulations should be performed to determine a method to reduce the following distance.

6.4 Implementation

6.4.1 Validation of a Single Vehicle

The implementation of a single vehicle following GPS coordinates was simple once the WiFi connection was reliable and the simulated gains were utilized. One consideration with using the VN-200 GPS module for guidance was that upon startup, the GPS magnetometer maintains a large error, up to 90°. The VN-200 has a calibration method built into the system, so the error was reduced by driving the vehicle at $2\,\mathrm{m\,s^{-1}}$ for roughly 50 m, followed by sitting stationary until the magnetometer reading matched an external compass.

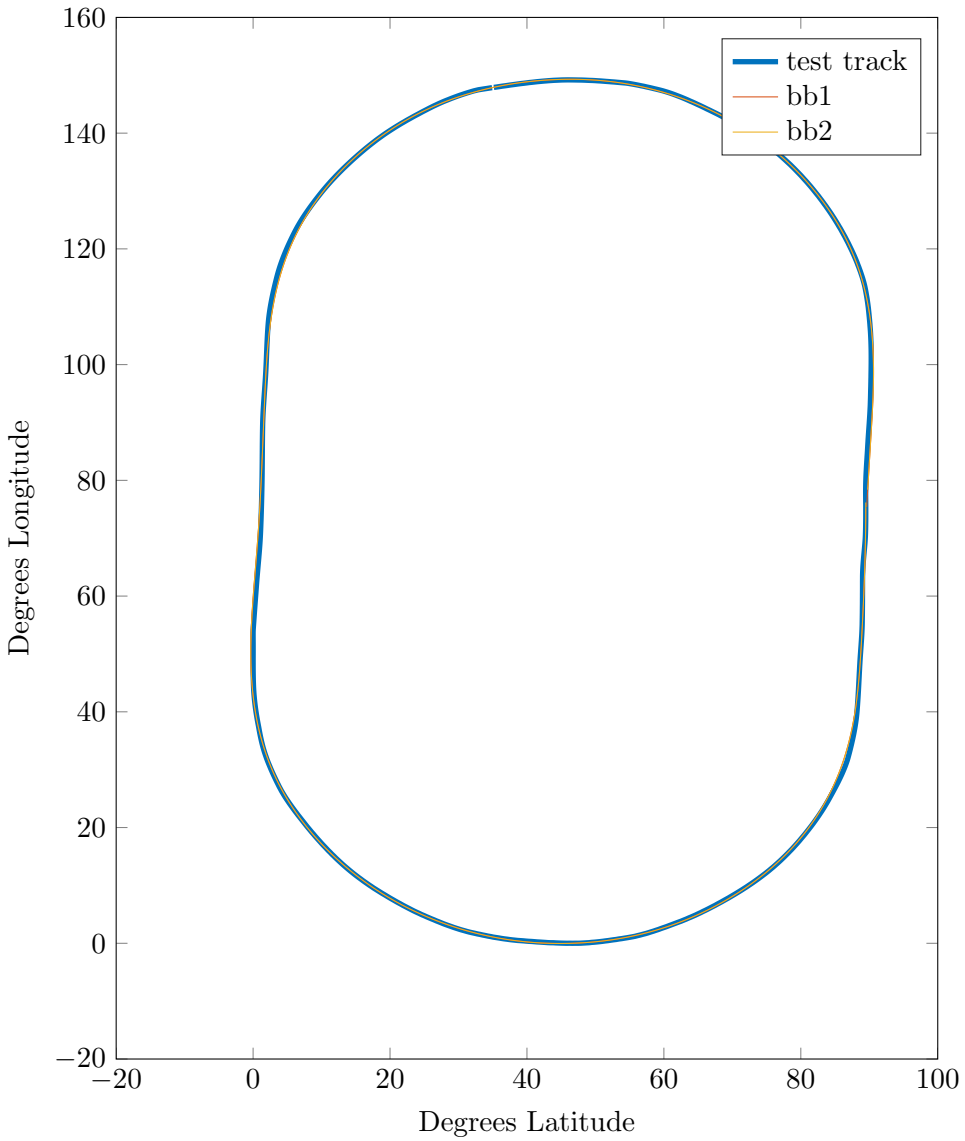

Fig. 6.3: Full path of 2 vehicles on the test track. It is clear that the vehicles are following the path. Path error and the longitudinal distance to the target are shown in Figures 6.4 and 6.5.

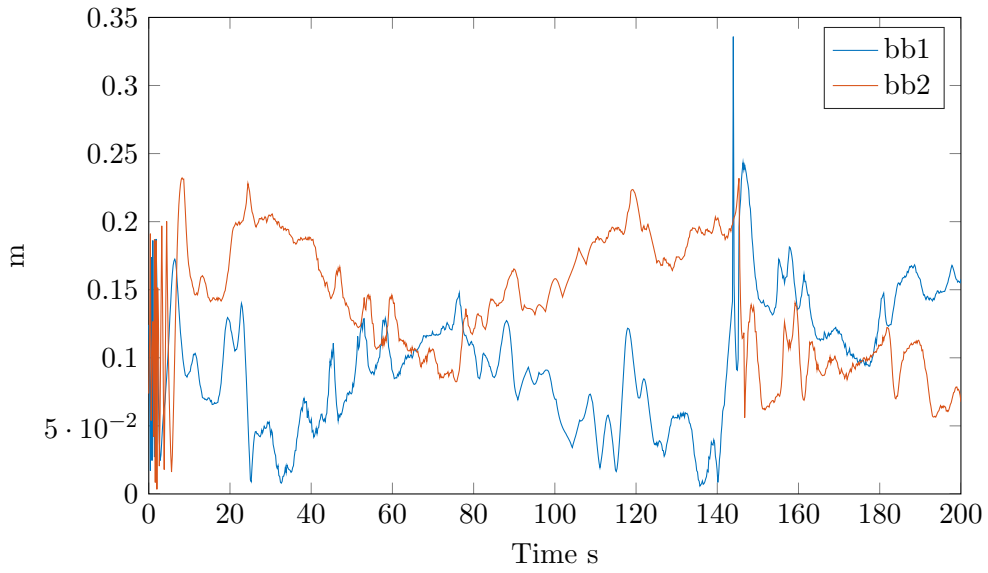

Fig. 6.4: Path error for each vehicle in the following system when using a desired velocity of $2\,\mathrm{m\,s^{-1}}$ and lookahead distance of 25 for the leader, with the follower using high level gains in Table 6.2.

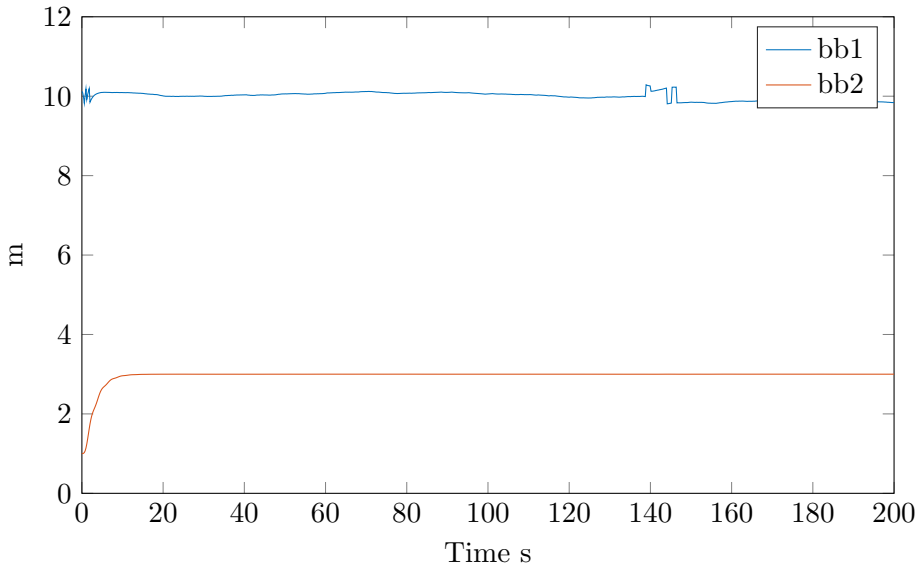

Fig. 6.5: Longitudinal distance to the target for the two vehicles. bb1 is following GPS coordinates while bb2 is following bb1.

The experimental validation followed the results found in simulation. Increasing vehicle velocity decreased lateral stability. The vehicle could reliably travel around the test track at speeds up to $3\,\mathrm{m\,s^{-1}}$.

Due to the GPS accuracy, the path on the road could not be guaranteed. On portions of the test track, the center of the road is uneven. The bumps in the road are large enough to disrupt vehicle operation for the standard vehicles. The enhanced vehicle with the larger wheel size can handle these bumps without disruption of any consequence.

6.4.2 Validation of a 2-Vehicle Platoon

The validation of the 2-vehicle platoon was more difficult than working with the individual vehicle. The simulations showed that the following vehicle would likely oscillate as it saw errors in following. In simulation, increasing the inter-vehicle distance dampened the oscillations, but the limited range of the camera required the inter-vehicle distance to be shorter than desired for stability. In addition, the camera's limited field of view required that the magnitude of the oscillations to be small.

As tests were being run, it became apparent that the hue-based license plate detection algorithm could be distracted by objects of a similar color. When the camera was trained to detect a red license plate, the algorithm had a false positive on a red tree, and when it was trained to detect blue, the algorithm detected the sky. A team member developed a Kalman filter to reduce the number of false positives generated by the detection algorithm.

The vehicle following algorithm was tested and found that if the initial conditions were correct, the vehicles could successfully operate as a platoon for half of the test track. If the vehicles didn't start directly in line with the path as they converged to the desired speed, the following vehicle would diverge from the path.

Team members implemented the vehicle following methods and obtained oscillation-free behavior at $4\,\mathrm{m\,s^{-1}}$. The solution was to add a damping term into the pure pursuit algorithm.

6.5 Discussion

The control algorithms selected from Chapter 2 were simulated and implemented on the vehicle experimental platform. Low level controllers which control steering and individual motor velocity were simulated to determine a range of reasonable gains. The gains were then tested physically and tuned to better match the vehicle system. The lateral and longitudinal controllers were simulated using GPS data and the update model to replicate the physical data gathering methods. Lookahead distances for various velocities were determined to be stable.

One vehicle was run on the test track to determine the accuracy of the selected gains. The gains from simulation allowed the vehicle to reliably track the path between $1\,\mathrm{m\,s^{-1}}$ and $3\,\mathrm{m\,s^{-1}}$. A second vehicle was added to the system, using the computer vision-based angle and distance estimators from Chapter 3 to determine the preceding vehicle position. The following vehicle accurately followed its predecessor under specific conditions, and it could not follow the predecessor fully around the track. Since then, team members have gotten the control system to work by applying a dampening term into the pure pursuit algorithm.

There is recommended area for future work in order to make the full platoon operate as a whole. Due to the limited accuracy of GPS, a physical marker system should be utilized. Potential options for the marker system are a painted line or a magnetic strip. This would allow all 10 vehicles to drive on the track without risk of hitting the uneven portions of the road. The lead vehicle or all 10 vehicles would then utilize a lanekeeping lateral control method to ensure path convergence.

CHAPTER 7

Conclusion

The purpose of this work was to determine effective methods for operating a set of automated ground vehicles in an adversarial environment. A platform was developed that allows for testing control algorithms in an extended attack.

Analysis of an extended attack required the vehicles to follow each other as part of a platoon. Lateral control algorithms were analyzed, and the algorithm with the minimum path error was selected. An adaptive backstepping lateral-longitudinal algorithm was presented as a platooning control law that estimates motor parameters.

Vehicles utilize various sensing methods to determine lateral and longitudinal control commands. A method to sense the inter-vehicle distance using a monocular camera was presented. This sensing method would be a mitigation technique if other sensors failed or were attacked. The sensing method is effective at short distances; longer distances would be attainable if using a camera with greater optical zoom or higher resolution.

An attack on a platoon of both automated vehicles and human-driven vehicles was simulated. The presence of human drivers stabilized the velocity of the platoon, but the humans also reduced the highway throughput.

The development of the automated vehicle platoon provided a vehicle platform and software package that enables future research on the reliability of automated vehicles.

The next path of work that I would approach after this is analyzing methods to determine the heading of the preceding vehicle. If the current heading of the target vehicle is known, the planning algorithm can more accurately follow the target trajectory.

The heading of the preceding vehicle can be difficult to detect. One method is to utilize the expected shape of a vehicle when using LIDAR input data. Another method would be to detect the distance to the rear lights on the preceding vehicle.

When a license plate is viewed straight on, the plate will appear to be rectangular. Due to graphical perspective, a rotated license plate appears as a trapezoid. This can be used to estimate the heading angle of the license plate. Future work will be to determine the details of this estimation method and the reliability of the approach.

In a vehicle following system, the trajectory accuracy can be improved by determining the heading of the preceding vehicle with respect to the current vehicle frame. Consider a vehicle detection system that can report the distance and line-of-sight (LOS) angle to a vehicle. The LOS angle and distance can be numerically differentiated to provide sufficient information to estimate the heading of the preceding vehicle.

This would be a significant accomplishment because it would allow for capturing data that can be difficult to obtain with a typical webcam. This method could also be used when determining the state of the preceding vehicle using LIDAR modules or other sensors. If the heading angle of the preceding vehicle is obtained by other means, a sensor fusion method can be utilized to increase state accuracy.

Some limitations of the work are the relatively short range of the monocular camera distance estimator (0.5–3 m) and the vehicle controller described in Chapter 2. The vehicle controller relied extensively on the skidding provided by the differential braking of the system, which would be hard on the vehicle.

The purpose of this work was achieved by exploring the design space of creating a reliable self-driving vehicle from a security perspective.

REFERENCES

[1] C. Vallance, "Car hack uses digital-radio broadcasts to seize control," *BBC, July*, 2015.

[2] D. Schrank, B. Eisele, and T. Lomax, "2014 urban mobility report," 2015.

[3] National Highway Traffic Safety Administration, "Traffic safety facts 2013," U.S. Department of Transportation, Tech. Rep., 2014.

[4] J. Miller, "Autonomous guidance and control of a roving robot." in *IJCAI*. Citeseer, 1977, pp. 759–760.

[5] G. H. Walker, N. A. Stanton, and M. S. Young, "Where is computing driving cars?" *International Journal of Human-Computer Interaction*, vol. 13, no. 2, pp. 203–229, 2001.

[6] D. Swaroop and J. Hedrick, "String stability of interconnected systems," *Automatic Control, IEEE Transactions on*, vol. 41, no. 3, pp. 349–357, 1996.

[7] J. Carbaugh, D. N. Godbole, and R. Sengupta, "Safety and capacity analysis of automated and manual highway systems," *Transportation Research Part C: Emerging Technologies*, vol. 6, no. 1, pp. 69–99, 1998.

[8] W. Yan, "A two-year survey on security challenges in automotive threat landscape," in *2015 International Conference on Connected Vehicles and Expo (ICCVE)*. IEEE, 2015, pp. 185–189.

[9] S. Dadras, R. M. Gerdes, and R. Sharma, "Vehicular platooning in an adversarial environment," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*. ACM, 2015, pp. 167–178.

[10] I. T. Erekson, "Modified trajectory shaping guidance for autonomous path following control of platooning ground vehicles," Master's thesis, Utah State University, 2016.

[11] D. Yanakiev and I. Kanellakopoulos, "A simplified framework for string stability analysis in ahs," in *Proceedings of the 13th IFAC World Congress*. Citeseer, 1996, pp. 177–182.

[12] R. Rajamani and C. Zhu, "Semi-autonomous adaptive cruise control systems," *Vehicular Technology, IEEE Transactions on*, vol. 51, no. 5, pp. 1186–1192, 2002.

[13] C. Chien and P. Ioannou, "Automatic vehicle-following," in *American Control Conference, 1992*. IEEE, 1992, pp. 1748–1752.

[14] J. Eyre, D. Yanakiev, and I. Kanellakopoulos, "String stability properties of ahs longitudinal vehicle controllers," UCLA Electrical Engineering, Tech. Rep., 1997.

[15] ——, "A simplified framework for string stability analysis of automated vehicles," *Vehicle System Dynamics*, vol. 30, no. 5, pp. 375–405, 1998.

[16] E. J. Rossetter and J. C. Gerdes, "A study of lateral vehicle control under a virtual-force framework," in *Proceedings of the International Symposium on Advanced Vehicle Control (AVEC)*. Citeseer, 2002, pp. 9–13.

[17] W. F. Powers and P. R. Nicastri, "Automotive vehicle control challenges in the 21st century," *Control engineering practice*, vol. 8, no. 6, pp. 605–618, 2000.

[18] S.-Y. Yi and K.-T. Chong, "Impedance control for a vehicle platoon system," *Mechatronics*, vol. 15, no. 5, pp. 627–638, 2005.

[19] P. Petrov, "Nonlinear adaptive control of a two-vehicle convoy," *Open Cybernetics & Systemics Journal*, vol. 3, pp. 70–78, 2009.

[20] C.-Y. Chan and H.-S. Tan, "Evaluation of magnetic markers as a position reference system for ground vehicle guidance and control," *California Partners for Advanced Transit and Highways (PATH)*, 2003.

[21] Q.-T. Luong, J. Weber, D. Koller, and J. Malik, "An integrated stereo-based approach to automatic vehicle guidance," in *Computer Vision, 1995. Proceedings., Fifth International Conference on*. IEEE, 1995, pp. 52–57.

[22] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, gps-denied environments," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 2643–2648.

[23] T. Bell, "Automatic tractor guidance using carrier-phase differential gps," *Computers and electronics in agriculture*, vol. 25, no. 1, pp. 53–66, 2000.

[24] J. Che, H. Yang, and B. Gao, "Study on evaluation of photoelectric jamming effectiveness on ranging lidar," in *Selected Proceedings of the Photoelectronic Technology Committee Conferences held June-July 2015*. International Society for Optics and Photonics, 2015, pp. 97 951V–97 951V.

[25] Q. Zhao and J. Jiang, "Reliable state feedback control system design against actuator failures," *Automatica*, vol. 34, no. 10, pp. 1267–1272, 1998.

[26] W. E. Halal, M. D. Kull, and A. Leffmann, "Emerging technologies: What's ahead for 2001-2030," *The Futurist*, vol. 31, no. 6, p. 20, 1997.

[27] R. A. El-laithy, J. Huang, and M. Yeh, "Study on the use of microsoft kinect for robotics applications," in *Position Location and Navigation Symposium (PLANS), 2012 IEEE/ION*. IEEE, 2012, pp. 1280–1288.

[28] K. Hausman, J. Müller, A. Hariharan, N. Ayanian, and G. S. Sukhatme, "Cooperative control for target tracking with onboard sensing," in *Experimental Robotics*. Springer, 2016, pp. 879–892.

[29] C. D. Crane Iii, D. G. Armstrong Ii, R. Touchton, T. Galluzzo, S. Solanki, J. Lee, D. Kent, M. Ahmed, R. Montane, S. Ridgeway *et al.*, "Team cimars navigator: An unmanned ground vehicle for the 2005 darpa grand challenge," in *The 2005 DARPA Grand Challenge*. Springer, 2007, pp. 311–347.

[30] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[31] R. Behringer, W. Travis, R. Daily, D. Bevly, W. Kubinger, W. Herzner, and V. Fehlberg, "Rascal-an autonomous ground vehicle for desert driving in the darpa grand challenge 2005," in *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.* IEEE, 2005, pp. 644–649.

[32] R. Rajamani, *Vehicle dynamics and control*. Springer, 2011.

[33] D. D. Dunn, "Attacker-induced traffic flow instability in a stream of automated vehicles," Master's thesis, Utah State University, 2015.

[34] G. P. Stein, O. Mano, and A. Shashua, "Vision-based acc with a single camera: bounds on range and range rate accuracy," in *Intelligent vehicles symposium, 2003. Proceedings. IEEE*. IEEE, 2003, pp. 120–125.

[35] C. Featherstone and M. Lowson, "Viability and benefits of platooning in automated transport systems," http://www.cybercars.org/docs/CTF%20Lowson%20Report.pdf, 2004, [Technical report; online; accessed 04-June-2014].

[36] Y. Seto, "Vehicular traveling control apparatus and method," Nov. 30 2010, uS Patent 7,844,384.

[37] D. Swaroop, "String stability of interconnected systems: An application to platooning in automated highway systems," *California Partners for Advanced Transit and Highways (PATH)*, 1997.

[38] S. Haj-Assaad. (2014) Top 10 affordable cars with adaptive cruise control. [Online]. Available: http://www.autoguide.com/auto-news/2014/09/top-10-affordable-cars-adaptive-cruise-control.html

[39] M. Y. Abualhoul, M. Marouf, O. Shagdar, and F. Nashashibi, "Platooning control using visible light communications: A feasibility study," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 1535–1540.

[40] T. Fujioka and M. Omae, "Vehicle following control in lateral direction for platooning," *Vehicle System Dynamics*, vol. 29, no. S1, pp. 422–437, 1998.

[41] T. C. Ng, J. I. Guzman, and M. D. Adams, "Autonomous vehicle-following systems: A virtual trailer link model," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 3057–3062.

[42] J. B. Hoagg and D. S. Bernstein, "Nonminimum-phase zeros-much to do about nothing-classical control-revisited part ii," *IEEE control systems*, vol. 27, no. 3, pp. 45–57, 2007.

[43] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," DTIC Document, Tech. Rep., 1992.

[44] J. Lowy and T. Krisher. (2016, July) Tesla driver's death using car's 'autopilot' probed by NHTSA. [Online]. Available: http://hosted.ap.org/dynamic/stories/U/US_SELF_DRIVING_CAR_DEATH

[45] N. Shchetko, "Laser eyes pose price hurdle for driverless cars," *The Wall Street Journal*, vol. 21, 2014. [Online]. Available: http://www.wsj.com/articles/laser-eyes-pose-price-hurdle-for-driverless-cars-1405969441

[46] X.-j. Cheng, K.-j. Cao, J.-n. Xu, and B. Li, "Analysis on forgery patterns for gps civil spoofing signals," in *Computer Sciences and Convergence Information Technology, 2009. ICCIT'09. Fourth International Conference on*. IEEE, 2009, pp. 353–356.

[47] A. Dimmeler, B. Eberle, J. C. van den Heuvel, A. L. Mieremet, H. Bekman, B. Mellier *et al.*, "Laser dazzling of focal plane array cameras," in *Optics/Photonics in Security and Defence*. International Society for Optics and Photonics, 2007, pp. 67 380O–67 380O.

[48] R. Chauhan, "A platform for false data injection in frequency modulated continuous wave radar," Ph.D. dissertation, UTAH STATE UNIVERSITY, 2014.

[49] (2016, July) Eyesight driver assist technology. Subaru of America. [Online]. Available: http://www.subaru.com/engineering/eyesight.html

[50] Y.-M. Chan, S.-S. Huang, L.-C. Fu, P.-Y. Hsiao, and M.-F. Lo, "Vehicle detection and tracking under various lighting conditions using a particle filter," *Intelligent Transport Systems, IET*, vol. 6, no. 1, pp. 1–8, 2012.

[51] C.-C. R. Wang and J.-J. J. Lien, "Automatic vehicle detection using local featuresa statistical approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 83–96, 2008.

[52] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (alpr): A state-of-the-art review," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, no. 2, pp. 311–325, 2013.

[53] H. Bai, J. Zhu, and C. Liu, "A fast license plate extraction method on complex background," in *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, vol. 2. IEEE, 2003, pp. 985–987.

[54] B.-Y. Feng, M. Ren, X.-Y. Zhang, and C.-L. Liu, "Effective license plate detection using fast candidate region selection and covariance feature based filtering," in *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*. IEEE, 2014, pp. 1–60.

[55] R. C. Luo and C.-C. Chang, "Multisensor fusion and integration: A review on approaches and its applications in mechatronics," *Industrial Informatics, IEEE Transactions on*, vol. 8, no. 1, pp. 49–60, 2012.

[56] J. P. Gray and V. V. Vantsevich, "Multi-vehicle convoy mobility in severe terrain conditions: Factor impact analysis, estimation and control strategy," *Journal of Terramechanics*, vol. 61, pp. 43–61, 2015.

[57] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 3, pp. 583–596, 2015.

[58] C.-N. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos, and E. Kayafas, "License plate recognition from still images and video sequences: A survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 3, pp. 377–391, 2008.

[59] C. Anagnostopoulos, I. Psoroulas, V. Loumos, E. Kayafas, and C. Anagnostopoulos, "Medialab lpr database," *Multimedia Technology Laboratory, National Technical University of Athens. Available: http://www. medialab. ntua. gr/research/LPRdatabase. html*.

[60] V. Tung. (2014, April) An updated guide to get hardware-accelerated opencv on beaglebone black. [Online]. Available: http://vuanhtung.blogspot.com/2014/04/and-updated-guide-to-get-hardware.html

[61] (2014) Cross compilation for arm based linux systems. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/introduction/crosscompilation/arm_crosscompile_with_cmake.html

[62] M. Darling. (2013, September) How to achieve 30 fps with beaglebone black, opencv, and logitech c920 webcam. [Online]. Available: http://blog.lemoneerlabs.com/3rdParty/Darling_BBB_30fps_DRAFT.html

[63] A. M. Muad, A. Hussain, S. A. Samad, M. M. Mustaffa, and B. Y. Majlis, "Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system," in *TENCON 2004. 2004 IEEE Region 10 Conference*. IEEE, 2004, pp. 207–210.

[64] A. D. Costanza, M. S. Breault, N. A. Wood, M. J. Passineau, R. J. Moraca, and C. N. Riviere, "Parallel force/position control of an epicardial parallel wire robot," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1186–1191, 2016.

[65] Battlekits - combat robot kits. [Online]. Available: http://battlekits.com/

[66] C. W. De Silva, *Mechatronics: an integrated approach*. CRC press, 2004.

[67] (2016, Feb) Ros.org — powering the world's robots. Open Source Robotics Foundation. [Online]. Available: http://www.ros.org/

[68] J. Kridner. (2016, Jun) Beaglebone black. [Online]. Available: https://beagleboard.org/black

[69] D. Thomas, D. Scholz, and A. Blasdel. (2016, Feb) rqt package summary. Open Source Robotics Foundation. [Online]. Available: http://wiki.ros.org/rqt

[70] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.

[71] S. Väliviita and O. Vainio, "Delayless differentiation algorithm and its efficient implementation for motion control applications," in *Instrumentation and Measurement Technology Conference, 1998. IMTC/98. Conference Proceedings. IEEE*, vol. 2. IEEE, 1998, pp. 881–886.

[72] M. Tham, "Discretised pid controllers," *Chemical Engineering and Advanced Materials, University of Newcastle Upon Tyne*, vol. 1998, 1996.

[73] J. Eskesen, "pixy_ros," https://github.com/jeskesen/pixy_ros, 2014.

APPENDICES

Appendix A

SATS Test Vehicle Hardware

The focus of the SATS Group is determining the security of autonomous vehicles, thus the experimental platform must be analogous to full scale highway vehicles. In addition to this requirement, the platform must be able to withstand collisions with other cars at high relative velocity. Due to the second requirement, full-scale vehicles could not be used, so vehicle kits from Battlekits were selected.

Two styles of vehicles were purchased, standard and enhanced. The standard configuration corresponds to a nominal highway vehicle. It is assumed that an attack vehicle would have greater velocity and acceleration capabilities than a normal highway vehicle, hence the enhanced vehicle configuration. The rest of this section will describe the standard vehicle configuration, with deviations for enhanced vehicles placed in parenthesis. For example, the standard vehicle has wheels with 4 in (5 in) diameter.

A.1 Electronics Platform

Daniel Dunn designed the electronics platform. It provides an enclosure that restricts the batteries from being ejected from the vehicle, as well as an area provided to mount the electronics system.

The platform was modified by team members to accomodate mounting a second lidar module for bidirectional platooning. The mechanical electronic system is now entirely housed under the electronics platform, with the power distribution block mounted on the underside of the electronics platform.

A.2 Vehicle Shell

The base robot with the electronics platform is a short box. The Lidar module has a very narrow beam, which makes it so a larger target would improve the reliability of the

Lidar sensor. In addition to this, the electronics platform was not covered. Semiconductors can behave erratically when directly exposed to sunlight, so a cover for the electronics would benefit the system.

Jackson Reid was recruited to design and construct a shell for the vehicles. He designed an aerodynamic vehicle shell to protect from the elements and a bumper system to protect the shell when the vehicles collide. The designed system is shown in Figure A.1.

Potential options for the material for the vehicle shroud were injected plastic, carbon fiber, Kevlar, fiberglass, and a carbon fiber / Kevlar weave. The cost of these methods was quoted out by a few companies, and the best prices are shown in Table A.1. The majority of the cost was from constructing the mold and labor. Due to this, we selected carbon fiber to construct the vehicles.

The shrouds were attached to the vehicles with hinges on one side, with side-release buckles on the other side. This secured the shroud sufficiently well while allowing for fast access to the electronics under the platform.

A.3 Quadrature Encoders

Two quadrature encoders are used to determine vehicle velocity, one encoder for each motor. The encoders measure motor rotation, sending a pulse at 2000 pulses per revolution. The encoder is interfaced with the quadrature encoder input module of the Tiva C microcontroller. The Tiva C calculates the velocity of each wheel using Equation A.1. This method is quite accurate at determining system velocity, but the system position is less reliable when the wheels slip. As the vehicle relies on skidding to steer, we do not use accumulated encoder distance to determine system position over time.

$$v = QEICOUNT\left(\frac{samples}{\text{sec}}\right)\left(\frac{rev}{encoder\ PPR}\right)\left(\frac{1}{gear\ ratio}\right)\pi(wheel\ diameter) \quad \text{(A.1)}$$

A.4 Lidar Module

The Lidar-LITE range sensing module is being used to measure distance to the target. One potential alternative was ultrasonic range sensors, but inexpensive ultrasonic range

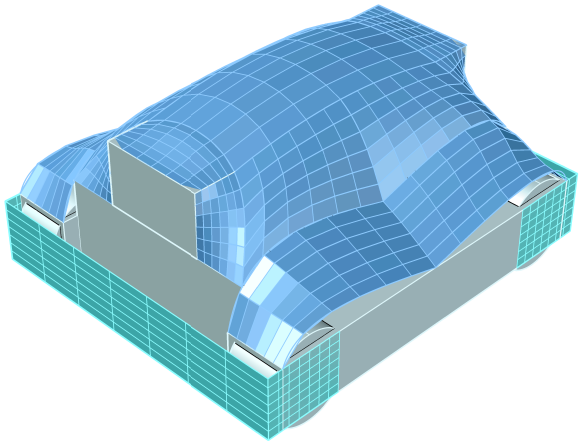

Fig. A.1: The vehicle shell, designed by Jackson Reid.

Material	Cost per unit
Plug and Mold	$500
Fiberglass	$150
Kevlar	$200
Carbon Fiber	$225
Carbon / Kevlar Weave	$275

Table A.1: Prices of materials to construct the vehicle shroud. The final plug was constructed using carbon fiber.

sensors had a maximum range of 2 m, which is a nominal following distance.

The Lidar-LITE module is not a true LIDAR module due to its mode of operation. Lidar-LITE sends a pulse of light using an LED instead of a laser, and detects the return pulse in terms of pixel shift. Due to this, the module was in the range of $100 instead of greater than $1000 per unit. With the inexpensive nature of the module, the Lidar-LITE module is not as well developed as other products.

Bidirectional platooning requires the use of two Lidar-LITE modules. This presented some problems in terms of communication with the sensor hub. The Lidar-LITE module transmits distance information via the I2C communication protocol. I2C is designed to allow multiple devices on one transmission bus. In nominal system operation, the "master" sends a device address onto the I2C bus, the "slave" device with the corresponding address responds, and the two communicate. One important aspect of I2C is that the device address should be reprogrammable, but the Lidar-LITE module does not have the ability to maintain a new device address after reboot. Due to these restrictions, two I2C busses must be active in order to read data from two Lidar-Lite modules.

The distance data from the Lidar-LITE module is numerically differentiated using the method from [71] to provide accurate velocity information.

A.5 Pixy Camera

The Pixy camera is described in Section 3.4.3. It utilizes an on-board processor to detect color signatures at high rates. The camera must be trained every time it encounters a new lighting situation, so we trained the camera on the same license plate in various lighting conditions. This allowed the camera to detect the proper region in multiple lighting situations.

A.6 USB Wireless Adapter

The USB wireless adapter is utilized to maintain connection with the command station. An 802-11n adapter was selected due to its capability to utilize the 5 GHz band. Further description of the reasoning for this is described in Section 4.3.2.

A.7 Tiva C Microcontroller

The sensor hub for the vehicles is required to interface with the sensors (Lidar-LITE module, GPS, quadrature encoders) then send the information to the Beaglebone Black via UART communication. The micro-controller used to accomplish this task is the Texas Instruments Tiva C EK-TM4C123GXL ARM Cortex-M4. The peripheral driver library is utilized to configure the peripheral devices for ease of use.

A.8 Beaglebone Black

The Beaglebone Black was selected to be the brains of the computer due to its extensive number of peripherals and association with Texas Instruments. The Beaglebone Black utilizes a 1 GHz ARM Cortex-A8 processor with NEON hardware acceleration. In addition to these features, it has two Programmable Real-time Units (PRUs).

The original design was supposed to replace the Tiva C with one of these on-board PRUs. After purchasing the Beaglebone Black modules, it was discovered that the PRUs do not have access to the required peripherals, and there is little documentation on how to interface with the PRUs.

The Beaglebone Black is capable of executing the control requirements as long as image processing is performed externally to the board. In the next iteration of the project, the main computing device should be something with multiple cores and hardware acceleration for image processing, such as the ODROID-XU4, Intel Galileo, or Raspberry Pi 3.

A.9 GPS

The VectorNav VN-200 Rugged GPS / INS is used by the lead vehicle to determine its position and orientation relative to the test track. The system transmits the GPS coordinates and orientation information to the Beaglebone Black via the sensor hub.

Testing and configuring this device proved to be problematic. When the unit is traveling at a velocity below $5\,\mathrm{m\,s^{-1}}$, the VN-200 relies on magnetometer measurements to determine its orientation. The lab where we were testing the system had a very strong magnetic field,

so when we moved the machine from one side of the room to the other, the observed north changed more than $\pi/2$ rad. This error was detected using a magnetic compass.

A.10 Battery System

The Battlekits manufacturers recommended utilizing the cars with three UB1250 12 V batteries connected in series, providing a 36 V supply to the motor controller. The batteries should be charged between 0.5 A and 1 A. The chargers the lab currently owns charges batteries at a maximum current of 2 A. One potential solution to this problem is to charge two vehicles in parallel. Until then, the batteries are charged at a higher current than is recommended.

Appendix B

ROS Package Operation

The *sats_car_ros* package utilizes two scripts, one for the controller and another for the sensor hub. The code for these scripts can be found at `github.com/smitchell7/sats_car_ros`. The system also relies on the pixy_ros package [73].

B.1 Messages

The nodes communicate using fixed message types. The message types are called in the preamble of the python file. The message types were developed to aid in inter-task communication as well as debugging purposes. The message structure is included in this document to clarify how data is passed.

Table B.1: CarCommand message. This message is used to configure each vehicle.

```
1  # this defines the structure of CarCommand. Do not use this directly
2  #
3  std_msgs/Header header
4  bool HL_active
5  bool HL_atk_mode
6
7  bool LL_active
8
9  # Vel mode. Set a velocity with an amplitude and period.
10 # Vel mode is only active if HL and LL are active
11 bool VEL_mode
12
13 float32 command_velocity
14 float32 d_amplitude
15 float32 d_period # 0 means velocity is constant
16
```

```
17 # Curvature automatic
18 bool curv_auto
19 float32 curv
20
21
22 # PWM mode is used for characterization. Use velocity mode for normal
       operation.
23 # PWM mode is given priority over the HL and LL
24 bool PWM_mode
25 # pwm values are used only if pwm mode is set
26 float32 pwmL
27 float32 pwmR
```

Table B.2: ControllerCommand message. The controller command is used to pass commands from the high level to the low level controller.

```
1  # differential drive steering commands.
2
3  std_msgs/Header header
4
5  #accel command
6  float32 accel
7  float32 vel
8
9  # set curvature OR angular_velocity
10 float32 curvature
11 #float32 angular_velocity
12
13 float32 velL
14 float32 velR
15
16 float32 pwmL
17 float32 pwmR
```

Table B.3: GpsData message. The GPS information is passed from the sensor hub to the controller if GPS mode is activated.

```
1  std_msgs/Header header
2
3  # yaw pitch roll
4  geometry_msgs/Point32 ypr
5
6  # angular rate xyz
7  geometry_msgs/Point32 angular
8
9  # lat long altitude 64 bit
10 geometry_msgs/Vector3 gps
11
12 # velocity ned
13 geometry_msgs/Point32 ned
14
15 # accel xyz
16 geometry_msgs/Point32 accel
```

Table B.4: KalmanData message. The Kalman filter reports its data to the user via this message.

```
1 #
2
3 float32 r
4 float32 rdot
5 float32 theta
6 float32 thetadot
7 float32[] covariance
```

Table B.5: MasterCommand message. The base station sends this one message to activate all of the vehicles simultaneously.

```
1 # Message that transports user commands.
2
3 # the header contains basic timestamp info
4 std_msgs/Header header
5 # manual activates / deactivates the high_level controller
6 bool master_on
7 # update parameters on all vehicles
8 bool update_param
```

Table B.6: SensorData message. The sensor hub sends all of the sensor data to the controller via this message.

```
1 # Message containing the sensor information of the sats car
2 std_msgs/Header header
3
4 VehicleState front
5 VehicleState rear
6 VehicleState left
7 VehicleState right
8
9 #GpsData vec
```

B.2 Sensor Hub

The sensor hub node retrieves control inputs from the Tiva C controller in a predefined format, described in the code. The sensor hub is run with a high nice level in order to allow the controller to execute at the control requirements.

B.3 Controller

The controller node operates by executing the low level controller every 50 Hz, with the high level controller activating for 1 in 5 iterations. The controllers were originally separate, but the message passing between nodes caused excessive delay in the system.

Appendix C

Vehicle Dynamics Simulator

C.1 High Level Controller

Table C.1: The high level controller function. The controller accepts the current velocity, desired velocity, and position, returning the desired velocity for each motor. The high level controller behaves exactly the same as the high level controller in the *sats_car_ros* package.

```
function [vL,vR,curv,dist,theta] = hl_ctrl(lat,long,heading,v_d,lad,kk,
    v,dt)
  lat = lat(:); long=long(:); heading = heading(:); v = v(:);
  pg = find_goal([lat(1),long(1),heading(1)],lad);

  gLat = [pg(1);lat(1:end-1)];
  gLong = [pg(2);long(1:end-1)];
  [dist,bearing] = haversine(gLat,gLong,lat,long);

  kp=0.75;kd=1;h=1; nom_dist = 1;
  accel = [dist(2:end)-nom_dist,-v(2:end),v(1:end-1)-v(2:end)]*[kp;kp*h
      ;kd];
  persistent v_des
  if isempty(v_des)
      v_des = [v_d;0*v(2:end)];
  end
  v_des = [v_d;v_des(2:end)+accel*dt];

  theta = bearing - heading;
  curv = pure_pursuit(dist,theta);
  [vL,vR] = steering(v_des,curv,kk);
  %v = [vL;vR];
```

```
21 end
22
23 function curv = pure_pursuit(dist, theta)
24   curv = 2.*sin(theta)./dist;
25 end
26 function [vL,vR] = steering(vel,curv,kk)
27   L = 0.319 .* kk;
28   vL = vel .* (1 + curv .* (L ./ 2));
29   vR = vel .* (1 - curv .* (L ./ 2));
30 end
31 function pg = find_goal(pr,l)
32   persistent pn
33   if isempty(pn)
34 %     file = load('../set_goal/pn3.mat','pn'); %coord-by-obs (2-by-n)
35 %     pn = file.pn;
36     pn = csvread('../path_norm/path.csv');
37     clear file;
38   end
39
40   % find nearest point on path
41   d = abs(pr(1)-pn(1,:)) + abs(pr(2)-pn(2,:));
42 %   d = sqrt((pr(1)-pn(1,:)).^2 + (pr(2)-pn(2,:)).^2);
43   [~,Ipc] = min(d);
44
45 %   pc = pn(:,Ipc); %closest point
46
47   % get goal point
48   Ipg = mod(Ipc + l - 1, size(pn,2)) + 1;
49   pg = pn(:,Ipg);
50 end
51 function [distance,bearing] = haversine(glat,glong,rlat,rlong)
52   R = 6372800; % radius of earth in meters
53   lat1 = degtorad(rlat); lat2=degtorad(glat);
54   dLat = lat2-lat1;
55   dLon = degtorad(glong-rlong);
```

```
56
57    a = sin(dLat./2).^2 + cos(lat1).*cos(lat2).*sin(dLon./2).^2;
58    c = 2.*asin(sqrt(a));
59    distance = R .* c;
60    y = sin(dLon) .* cos(lat2);
61    x = cos(lat1) .* sin(lat2) - sin(lat1) .* cos(lat2) .* cos(dLon);
62    bearing = atan2(y,x);
63    %[distance/1000, bearing * 180/pi]
64  end
```

C.2 Vehicle Dynamics Simulator

Table C.2: The high level integrated track function. The function utilizes vehicle dynamics to update the path of a vehicle on the test track. The simulator relies on the high level controller function to calculate the desired trajectory.

```
1
2  %% functionname: function description
3  function pe=hl_integrated_track(v_des,lad,kk,plot_on,plot_mov)
4
5  switch nargin
6      case 0
7          v_des = 2;
8          lad = 41;
9          kk = [1;1];
10         plot_on = 1;
11         plot_mov = 0;
12     case 1
13         lad = 19;
14         kk = 1;
15         plot_on = 1;
16         plot_mov = 0;
17     case 2
18         kk = 1;
```

```
19          plot_on = 1;
20          plot_mov = 0;
21      case 3
22          plot_on = 1;
23          plot_mov = 0;
24      case 4
25          plot_mov = 0;
26  end
27
28  deglat = 111069; % meters
29  deglong = 83162;
30  N=2;
31  HP = struct(...
32      'lookahead',lad...
33      ,'kk',kk...
34      ,'v_des',v_des...
35      ,'L',0.319...
36      ,'k',[0.5;0.3]...
37      ,'N',N ...
38      );
39
40  c1 = [41.75939;... %lat
41   -111.8166653;... % long
42   1.*pi;... % heading
43   0;0; ... %error integral
44   0;0 ... %velocity
45   ];
46  c2 = c1; c2(1)=c2(1)+1/deglat;
47  dyn = reshape([c1,c2]',[],1);
48  sim_time = 200; dt = 0.2;
49
50  t=linspace(0,sim_time,sim_time/dt).';
51  % v_des = 5; %lad = 30;
52  f_w = @(t,Z) f(t,Z,HP);
53  y = ode4(f_w,t,dyn);
```

```
54
55 lat =           y(:,1:N);
56 long=           y(:,N+1:2*N);
57 heading =       y(:,2*N+1:3*N);
58 error_int_L =   y(:,3*N+1:4*N);
59 error_int_R =   y(:,4*N+1:5*N);
60 vL =            y(:,5*N+1:6*N);
61 vR =            y(:,6*N+1:7*N);
62
63 pn = csvread('../path_norm/path.csv');
64 if plot_on
65     pnm = bsxfun(@times,bsxfun(@minus,pn.',min(pn.')),[deglat,deglong])
          ;
66     ym = bsxfun(@times,bsxfun(@minus,y(:,1:2),min(pn.')),[deglat,
          deglong]);
67     latm = (lat-min(pn(1,:))) * deglat;
68     lonm = (long-min(pn(2,:))) * deglong;
69
70     figure(1)
71     plot(pnm(:,2),pnm(:,1),'LineWidth',2);hold on
72     plot(lonm,latm); hold off
73     title('Vehicle following gps coordinates');
74     legend('test track','bb1','bb2')
75
76     figure(2)
77     hold off
78     vid = VideoWriter('on_test_track');
79     vid.FrameRate = round(1/dt);
80     open(vid);
81     %
82     plot(pn(2,:),pn(1,:))
83     axis([min(long(:)),max(long(:)),min(lat(:)),max(lat(:))])
84     title('Vehicle following gps coordinates')
85     xlabel('degrees longitude')
86     ylabel('degrees latitude')
```

```
87      hold on
88      handle = plot(long(1,:),lat(1,:),'o');
89  end
90  path_error = nan(length(y),N);
91  dist = path_error;
92  curv = path_error;
93  theta= path_error;
94      for i = 1:length(y)
95          [path_error(i,:),~] = min(...
96              sqrt(...
97              (bsxfun(@minus,kron(pn(1,:),ones(N,1))',lat(i,:)).*111069)
                    .^2+...
98              (bsxfun(@minus,kron(pn(2,:),ones(N,1))',long(i,:)).*83162)
                    .^2));
99  %          [~,curv(i)]= hl_ctrl(y(i,1:3),v_des,lad,kk);
100         [~,~,curv(i,:),dist(i,:),theta(i,:)] = hl_ctrl(lat(i,:),long(i
                ,:),heading(i,:),v_des,lad,kk,(vL(i,:)+vR(i,:))/2,dt);
101         if plot_mov && ~mod(i,1)
102             set(handle,'XData',long(i,:),'YData',lat(i,:));
103             title(['Vehicle following gps coordinates. t=',num2str(t(i)
                    )])
104
105             drawnow
106             writeVideo(vid,getframe);
107         end
108     end
109 if plot_on
110     hold off
111     figure(3)
112     subplot(3,1,1)
113     plot(t,path_error);title('path error')
114     subplot(3,1,2)
115     plot(t,curv); title('Commanded curvature');
116     subplot(3,1,3)
117     plot(t,dist); title('Distance to target');
```

```
118     figure(4)
119     subplot(3,1,1)
120
121     plot(t,(vL+vR)/2)%,t,v_des+v_des./2.*sin(0.1.*t))
122     title('velocity')
123     subplot(3,1,2)
124     plot(t,heading)
125     title('heading')
126     subplot(3,1,3)
127     plot(t,wrapTo180(theta*180/pi))
128     title('angle to target');legend('bb1','bb2')
129 %     subplot(3,1,1)
130
131     close(vid)
132 end
133 pe = max(path_error);
134 end
135 %% f:
136 function [Z_dot] = f(t,Z,HP)
137 persistent counter
138 if isempty(counter)
139     counter = 0;
140 end
141
142 lookahead = HP.lookahead;
143 kk = HP.kk;
144 v_des = HP.v_des;
145 % v_des = v_des+v_des * sin(0.1*t)/2;
146
147 L = HP.L;
148 k = HP.k(:);
149 N = HP.N;
150 db = [0.13;0.14]*36;
151 a = [-2.7,0;0,-2.6];
152 b = [0.8,0;0, 0.77];
```

```
153
154
155 %L = 0.319;
156 %k = [0.5;0.3];
157 deglat = 1/111069; % meters
158 deglong = 1/83162;
159
160 % fill the position and velocity vectors.
161 noise = kron([0.5*deglat;0.5*deglong;10*pi/180;0;0;0.05;0.05],ones(N,1)
        ) .* randn(length(Z),1);
162 noise = 0;
163
164 Z = Z+noise;
165
166 lat = Z(1:N);
167 long= Z(N+1:2*N);
168 heading = Z(2*N+1:3*N);
169
170 error_int_L = Z(3*N+1:4*N);
171 error_int_R = Z(4*N+1:5*N);
172
173 vL = Z(5*N+1:6*N);
174 vR = Z(6*N+1:7*N);
175
176 vel = (vL+vR)./2;
177
178 vN = vel .* cos(heading);
179 vE = vel .* sin(heading);
180 dt = 0.2;
181 [vL_des,vR_des] = hl_ctrl(lat,long,heading,v_des,lookahead,kk,vel,dt);
182
183 v_err = [vL_des,vR_des] - [vL,vR];
184 vL_err = vL_des - vL;
185 vR_err = vR_des - vR;
186
```

```
187 uL = [vL_err,error_int_L] * k;
188 uR = [vR_err,error_int_R] * k;
189 % u = v_err' * k;
190 %u = [uL,uR]';'
191 %error = vd_vec-v;
192 %u = [error,error_int] * k;
193
194 % v = [vL,vR]';
195 % dv = a*v + b*(u+db);
196 dvL = a(1,1)*vL + b(1,1)*(uL+db(1));
197 dvR = a(2,2)*vL + b(2,2)*(uR+db(2));
198
199
200 Z_dot = [...
201     vN*deglat;... % lat
202     vE*deglong;... % long
203     (vL-vR)./L;... % heading
204     dvL;dvR; ... % motor accel
205     vL_err; ... % motor error
206     vR_err ... % motor error
207     ];
208
209 end
```