DEVELOPMENT AND IMPLEMENTATION OF AN ADAPTIVE CRUISE
CONTROL FEATURE IN A CONSUMER VEHICLE

A Thesis
Presented to
The Academic Faculty

By

Nicholas Hummel

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

May 2022

© Nicholas Hummel 2022
ACKNOWLEDGMENTS

First I would like to thank the members of the Georgia Tech EcoCAR Mobility Challenge team for their tireless effort and passion for this project. Without them none of this would have been possible. The impact of numerous late nights and early mornings shows starkly in the final product of the team heading into the final competition of this four year project.

I would also like to thank my advisor, Dr. David Taylor. His undergraduate class was my first step down a journey that proved impactful beyond anything I could have imagined, and his support throughout this project has facilitated the most dramatic growth in my engineering career so far.

Finally, I would like to thank our other faculty advisors, Dr. Leamy and Dr. Fuller, and the sponsors and organizers of the EcoCAR Mobility Challenge for their support of the EcoCAR competition at Georgia Tech. I have seen this project help dozens of young engineers improve their skills over the past two years. It is truly unique in the collegiate landscape.
# TABLE OF CONTENTS

Acknowledgments ................................................................. iii

List of Tables ................................................................. vi

List of Figures ................................................................. vii

Summary .......................................................... ix

Chapter 1: Introduction and System Architecture ....................... 1

1.1 Competition Overview ............................................... 1

1.2 CAVs System Architecture .......................................... 2

Chapter 2: Perception System Design ..................................... 4

2.1 Sensor Fusion .......................................................... 4

2.2 Hardware Considerations ............................................ 5

2.3 Algorithm Overview .................................................. 5

2.4 Analysis of Different IMM Architectures ......................... 10

Chapter 3: Perception System Implementation .......................... 14

3.1 Parsing CAN Data ....................................................... 14

3.2 Standardizing Messages ............................................. 15

3.3 ROS Communication ................................................ 16
A table of contents is shown:

<table>
<thead>
<tr>
<th>Table Number</th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Performance Errors for Mismatched Parameters</td>
<td>28</td>
</tr>
<tr>
<td>5.1</td>
<td>Parameter Identification of Chevy Blazer</td>
<td>43</td>
</tr>
</tbody>
</table>
### LIST OF FIGURES

1.1 EcoCAR CAV System Architecture ........................................... 1

2.1 IMM Filter Algorithm .............................................................. 8

2.2 GOSPA Metric for CV-CA-CT Architecture ................................. 11

2.3 IMM Probabilities for CV-CA-CT Architecture ............................. 12

2.4 GOSPA Metric for CV-CA Architecture ...................................... 12

2.5 IMM Probabilities for CV-CA Architecture .................................. 13

3.1 Perception Test with Stationary Lead Vehicle ............................... 17

3.2 Perception Test with Dynamic Lead Vehicle ................................. 18

3.3 Perception Test with Cut-In and Cut-Out .................................... 19

3.4 Perception Validation Drive on I-85 ........................................... 20

4.1 Distance Controller Setup ......................................................... 23

4.2 ACC Mode Switching Behavior .................................................. 24

4.3 ACC Vehicle Controller Architecture ......................................... 25

4.4 Feedback Controller Simulation with 10deg Decline ....................... 27

4.5 ACC Controller Simulation Velocity Mode ................................... 29

4.6 ACC Controller Simulation Distance Mode .................................. 30

4.7 ACC Controller Simulation Velocity to Distance Gradual Switch ........ 31
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.8</td>
<td>ACC Controller Simulation Velocity to Distance Cut-In Switch</td>
<td>32</td>
</tr>
<tr>
<td>4.9</td>
<td>ACC Controller Simulation Distance to Velocity Gradual Switch</td>
<td>33</td>
</tr>
<tr>
<td>4.10</td>
<td>ACC Controller Simulation Distance to Velocity Cut-Out Switch</td>
<td>34</td>
</tr>
<tr>
<td>5.1</td>
<td>Modified ACC Mode Switching Behavior</td>
<td>36</td>
</tr>
<tr>
<td>5.2</td>
<td>ACC 0 to 40mph Test with No Lead Vehicle</td>
<td>38</td>
</tr>
<tr>
<td>5.3</td>
<td>ACC Stop and Go Test</td>
<td>40</td>
</tr>
<tr>
<td>5.4</td>
<td>ACC Highway Drive Cycle Simulation</td>
<td>41</td>
</tr>
<tr>
<td>5.5</td>
<td>ACC Parameter Identification</td>
<td>42</td>
</tr>
<tr>
<td>5.6</td>
<td>ACC Controller Gain Search for Highway Drive Cycle</td>
<td>44</td>
</tr>
<tr>
<td>5.7</td>
<td>ACC Endurance Run</td>
<td>45</td>
</tr>
</tbody>
</table>
This thesis covers work to develop and implement an Adaptive Cruise Control (ACC) feature in a consumer vehicle. More specifically, the development of a perception system and vehicle controller enabling automated control over a vehicle’s longitudinal motion will be discussed. The ACC feature is able to follow other vehicles while maintaining a safe following distance or to maintain a user-set velocity in the absence of a lead vehicle. This research is driven by a larger competition, the EcoCAR Mobility Challenge, sponsored by the U.S. Department of Energy, General Motors (GM), and Mathworks. The competition tasks teams with integrating a hybrid drivetrain and developing Connected and Autonomous Vehicle (CAVs) technology to improve energy efficiency, rider comfort, and safety in a 2019 Chevy Blazer. This research leveraged off-the-shelf industry grade tools, sensors, and actuators including MATLAB’s Automated Driving and Sensor Fusion and Tracking Toolboxes, the Robot Operating System (ROS), a Mobileye camera unit, a Bosch radar sensor, and electric machines from Denso and Magna. The problems were modeled in a software environment and then deployed in the road vehicle to evaluate and tune performance. These features will be evaluated by the competition in a final event in May 2022 at a GM proving grounds facility.
1.1 Competition Overview

The EcoCAR Mobility Challenge is a four year competition tasking 11 collegiate teams across North America to develop hybrid and automation features for consumer vehicles. The main focus of the vehicle automation component of this competition is development of an adaptive cruise control feature, which controls the distance of a vehicle to the next vehicle in its lane or maintains a user-set speed in the absence of a lead vehicle. This feature will be developed from an algorithmic standpoint, then discussion will follow with the integration of the feature in a 2019 Chevrolet Blazer.

Figure 1.1: EcoCAR CAV System Architecture
1.2 CAVs System Architecture

In order to achieve the goals outlined by the competition, the team augmented the stock vehicle with a suite of sensors, computing devices, and communication modules shown in Figure 1.1. Chief among these components is the CAVs main controller, the Intel TANK-870-Q170. The Tank is responsible for receiving information from sensors, computing the distance to the nearest lead vehicle, computing a control action, and relaying said action to the vehicle’s supervisory controller, a dSPACE MicroAutoBox (MABx), which interfaces with the vehicle’s power train. The sensor suite selected for the forward facing perception system is comprised of an Intel Mobileye 6 camera unit, and a Bosch MRR4 Front Radar. In addition to the main controller and sensors, the team has integrated a Cohda Mk5 Dedicated Short-Range Communications (DSRC) radio for development of V2X features including connected corridor traversal. Finally, the team added an Nvidia Jetson Xavier AGX controller for control of features specific to the Human Machine Interface (HMI).

The components shown in blue in Figure 1.1 are components that the team wrote software for. The sensors shown in yellow are unmodified from the manufacturer. The amount of modification varied for each component. For example, the CAVs controller exclusively ran team-written programs, but the V2X radio had only minor modifications made to the stock firmware.

The CAVs controller communicates with the sensors of the forward perception suite and the vehicle’s hybrid supervisory controller via Controller Area Network (CAN) protocol. A diagnostics port was also added to satisfy the requirements of the competition. This port communicates information about the objects detected by the CAVs system, the torque request to the vehicle, and signals to communicate the current operating mode of the system. The CAVs controller did not come with native CAN communication support, so the team added hardware from Kvaser to relay CAN information to the controller. The V2X radio
and the HMI controller are connected to the CAVs controller via an Ethernet interface. These connections were individually analyzed and tested to ensure that each had suitable bandwidth for the information the team needed to convey.
 CHAPTER 2
PERCEPTION SYSTEM DESIGN

The ACC problem has been decomposed into two primary components: a perception system and a vehicle controller. The purpose of the perception system is to reliably inform the vehicle controller about the next closest vehicle in front of and in the same lane as the ego vehicle. In this project, the vehicle’s sensors provide 3D positional and velocity data about surrounding vehicles and obstacles. It is the task of the designer to track these detections over time and fuse the output of different sensors. Previous work in this area proposes the Kalman filter as the optimal estimator given knowledge of an object’s dynamics and Gaussian sensor noise [1]. Surrounding vehicles’ motion cannot be known a priori, since their motion is controlled by human agents. The Interacting Multiple Model (IMM) filter was proposed as a more accurate estimator in this case [2]. The IMM filter runs multiple Kalman filters with different internal dynamics in parallel and treats an object as being represented by a linear combination of these Kalman filter’s estimates. It associates a running probability with each filter representing an object at a given time step. In order to measure the effectiveness of the perception system, commonly used metrics had to be understood. The OSPA metric was proposed as such a metric specifically in the automotive space [3]. Additionally, the similar Generalized Optimal SubPattern Assignment (GOSPA) metric is provided off-the-shelf in MATLAB. It combines positional accuracy along with false detections and track association into a scalar value.

2.1 Sensor Fusion

The primary objective of the team’s sensor fusion algorithm is to combine the objects detected by the Bosch front radar and the Mobileye camera unit into a more accurate repre-
sentation of the world around the vehicle and select the lead vehicle if there is one present. While the Bosch radar and the Mobileye camera have been tested and validated in industry and are therefore highly reliable, they both have their own strengths and weaknesses. The radar sensor detects the position of objects much more accurately, while the camera is more effective at telling the difference between vehicles and other objects. Thus, the team was motivated to create an algorithm to combine the strengths of each sensor.

In designing the fusion system, the team outlined certain minimum performance metrics the algorithm must achieve. First, it must be capable of detecting objects in the vehicle’s path at a sufficiently far distance, 120m, so that the car can be brought to a stop with the maximum deceleration rate allowed by the competition, -0.5g, at the maximum anticipated speed for ACC to be active, 60mph. Second, the system must be able to track objects effectively even if the detections from the sensors are intermittent or contain noise. The team utilizes multiple proven filtering methods to predict object position for future time steps and update accordingly. Finally, the system must avoid generating false tracks. Here also the team utilizes filters commonly found in industry to build a robust perception system.

## 2.2 Hardware Considerations

The team outlined certain criteria that must be satisfied for the sensor fusion system to function properly and safely. These included the following: no fault messages from any sensors, consistent reception of messages from all sensors, and validation of a rolling counter with the vehicle’s supervisory controller. If any of these conditions were not satisfied, the CAVs controller would trigger the CAVs system fault flag in the CAN messages sent to the supervisory controller, disable the ACC feature, and alert the driver.

## 2.3 Algorithm Overview

The inputs to the sensor fusion algorithm are object detections from the radar and vision sensors at the current time step and the output is the longitudinal displacement of the lead
vehicle, which is fed to the feedback controller. If no lead vehicle is detected, the perception system sets the output to Inf. In addition to the input and output signals, the algorithm contains state variables to include information from prior detections. Since the motion of other vehicles on public roadways is highly variable, the team needed to employ a method of tracking other vehicles which would be computationally efficient and robust. To achieve this, the team implemented an IMM filter with three distinct models to track objects across multiple time steps and a Global Nearest Neighbors (GNN) filter to match detections of the same vehicle from the two sensors. The three motion models the team explored for use inside the IMM are the Constant Velocity (CV) model, the Constant Acceleration (CA) model, and the Constant Turn Rate (CT) model. The architecture of the algorithm was developed to balance the computational complexity with the efficacy of the estimation process.

**Tracks vs Object Detections:** Throughout this discussion the term ‘track’ will refer to an object that has been detected with high certainty over multiple time steps. The team has further made a distinction between two subclasses of tracks: confirmed and tentative. Confirmed tracks have a lower uncertainty. The team uses a tunable parameter for setting this threshold. Tentative tracks are tracks that have a higher uncertainty and thus aren’t used to select the lead vehicle, but these tracks are still used in future updates, since they may become confirmed tracks with new sensor data. Object ‘detections’ are measurements from the radar and vision sensors at the current time step and includes information such as object position, velocity, and acceleration. The sensors in the team’s perception system are industry grade with their own internal estimation algorithms and transmit a probability of existence for each object. The present version of the team’s algorithm treats any object with a reported probability of existence greater than 70% as valid, which was cited as a reasonable threshold by the manufacturers. The problem for the sensor fusion algorithm then becomes matching detections over multiple time steps and combining objects seen by both sensors. To do this the team implements an IMM filter for matching detections to
tracks and a GNN filter for combining radar and vision tracks.

**The IMM Filter:** The team conducted a literature review of estimation algorithms to better understand the subject, where it found the Interacting Multiple Model filter. The IMM filter has been shown to be effective in tracking objects executing complex maneuvers [4]. The work by A. Genovese at Johns Hopkins has confirmed this for the system of aircraft in 3D space [2]. The team sought to apply this work to the 2D system of vehicles on a road. The necessity for a more complex filtering method than a traditional linear Kalman filter is dictated by the complexity of the system the team is trying to estimate. For the unpredictable movement of other vehicles, the team uses a combination of multiple simple filters to predict and calculate the probability of each model representing an object at the current time step. Specifically the algorithm estimates the object as a linear combination of an object moving with constant velocity, an object moving at constant acceleration, and an object moving at a constant turn rate. In order to store the information about the past, the filter must generate a likelihood of any given model representing a detection based on a single update. These predictions are then used in the next time step to calculate new probabilities for each model. If the probability matrix for every time sample had to be stored and compared with new sensor measurements, one can quickly see how the required amount of memory and number of calculations would rapidly approach infinity. To avoid this infinite computational load the team implemented the IMM filter, a tool developed in the late 20th century, which only relies on information from the last time step, the current time step, and a pre-defined matrix which models the likelihood for an object to transition between models [5].
Figure 2.1 shows an illustration of how the IMM filter is organized. The state interactions block combines the previous state estimates from each filter along with the model probability. The state estimate for the overall filter is denoted as $\hat{X}$, the state estimate for the $i$th model is $\hat{X}^i$, the update for the $i$th model is $Z^i$, the mixed state estimate for model $i$ is $\hat{X}^{0i}$, the likelihood for each filter model is $\Lambda^i$, the updated transition probability matrix is $\mu$, the conditional model probabilities are $\tilde{\mu}$, and $1/z$ is the unit time delay.

$$\hat{X} = \text{state estimate for the overall filter}$$
$$\hat{X}^i = \text{state estimate for the } i\text{th model}$$
$$Z^i = \text{update for the } i\text{th model}$$
$$\hat{X}^{0i} = \text{mixed state estimate for model } i$$
$$\Lambda^i = \text{likelihood for each filter model}$$
$$\mu = \text{updated transition probability matrix}$$
$$\tilde{\mu} = \text{conditional model probabilities}$$
$$1/z = \text{unit time delay}$$
transition probability matrix $\tilde{\mu}$ to create the combined previous time step predictions for each model. Each model then filters new sensor measurements in an update step, which are passed back to the model probability update block. This block calculates the probability of models representing the actual object using the previous model transition probability matrix, a user-defined baseline transition matrix, and the outputs of the three models. Finally, the state estimation block calculates the overall output of the filter $\hat{X}$ using the model probability update matrix and the new state predictions from the three models.

The Linear Kalman Filter: The team uses a linear Kalman filter to predict the updates for the CV and CA models. The Kalman filter works in two discrete steps: prediction and update. The prediction step uses a pre-defined model for the dynamics of the system it is meant to estimate, to predict the state of the system at the next time step along with an uncertainty metric for each state. The update step takes the predicted state, uncertainty measures, and new sensor information and updates the current state and uncertainties. The Kalman filter has been shown to be the optimal estimation algorithm when the performance metric is the mean squared estimation error for a linear system with known dynamics and Gaussian sensor noise [6]. To implement the Kalman filter for the CV and CA models the team utilizes the TrackingKF class from the MATLAB Sensor Fusion and Tracking Toolbox.

The Extended Kalman Filter: In the case of estimating the states for the CT model the team encountered a problem with applying the linear Kalman filter. Namely, the dynamics of the CT model were nonlinear and the first order linearization only held for a small neighborhood. To account for this the team uses an Extended Kalman Filter (EKF) to predict the updates for the CT model. The EKF functions almost identically to the linear Kalman filter, except it linearizes the dynamics of the model about the current state for each time step. While the EKF is not an optimal estimator due to the loss of higher order terms, it has still been shown to be effective in estimating systems that are locally linearizable [1]. In its Model in the Loop (MIL) testing the team found that this filter was effective in
modeling the constant turn model. To implement the Extended Kalman Filter for the CT model the team utilizes the TrackingEKF class from the MATLAB Sensor Fusion and Tracking Toolbox.

The GNN Filter: Once tracks have been updated for the current time step with new sensor information, the team utilizes a GNN filter to group tracks from the two sensors together. For example, if there are multiple tracks in closer proximity than the size of a car, the team can assume these tracks are identifying the same object. This is necessary so that the algorithm never has to decide between two tracks representing the same object when selecting the lead vehicle.

2.4 Analysis of Different IMM Architectures

After a literature review of the IMM filter and development of a test-bed in MATLAB, the team was interested in investigating the architecture of the IMM filter and the trade-off between complexity and performance. The team reviewed the work by A. Genovese at John’s Hopkins APL in developing a radar tracking systems for aircraft [2]. In this research, an IMM filter is implemented with different combinations of CV, CA, and CT models. Interestingly the team found that the CV-CA architecture had very similar performance to the CV-CA-CT architecture despite having no nonlinear components. The team was interested in investigating if this simpler model, shown to be almost equally effective in 3D space, would hold for a planar system such as the lead vehicle identification problem.

The team tested multiple scenarios in a MATLAB simulation environment with access to a host of ground truth data to evaluate the relative change in performance of the estimation algorithm. The team evaluated the number of false tracks and the GOSPA metric, which penalizes localization errors for properly detected targets, missed targets and false targets, for multiple scenarios ranging from a vehicle traveling in a straight line to more complex scenarios with turns in the road with multiple vehicles present. After MIL testing the team concluded that the two architectures, CV-CA and CV-CA-CT, were very compa-
rable, confirming that the findings of Genovese applied to the planar projection of the lead vehicle selection problem. Figure 2.2 and Figure 2.4 show the response of the GOSPA metrics and the components that comprise it for a three model and two model IMM filter. Figure 2.3 and Figure 2.5 show the internal probability measures for each model in the three model and two model IMM filters. In this scene one vehicle moves in and out of the ego lane where the road is straight for the first 80 seconds and has a constant curvature for the remainder of the scenario.

As illustrated in Figure 2.3 the IMM filter models the lead vehicle as a constant velocity object while both vehicles are traveling at the same velocity on a straight road. Then once the lead vehicle enters the turn around the 80 second mark, the IMM filter quickly switches to high confidence in the constant acceleration model. Finally, once both cars are in the turn around the 100 second mark, the filter switches back to high probability of a constant velocity model. Since the turn radius was so wide, 200m, the constant turn model had almost 0 probability of representing the object. As expected the constant turn model was more suited for a turning target and a stationary sensor, as was the case in [2], where an aircraft traveled in 3D space and the sensors were fixed on the ground. For the entirety of the test the tracking performance metric, which penalizes localization errors for properly
Figure 2.3: IMM Probabilities for CV-CA-CT Architecture

detected targets, missed targets, and false targets remains low.

Figure 2.4: GOSPA Metric for CV-CA Architecture

It was confirmed in this test scenario that the CV-CA architecture could perform approximately equivalently to the CV-CA-CT architecture. As is show in the GOSPA plots, the maximum value localization errors for both architectures is the same.
In trying to understand why the CV-CA architecture performed as well as the more complex CV-CA-CT model, the team developed a model for describing the first order approximation of the system. The team found that for small time steps and large curves, a vehicle entering a turn could be approximated by a constant acceleration model and a vehicle in a turn with the ego vehicle turning at the same rate could be approximated by a constant velocity model. This gave a plausible explanation for the simulation data the team had seen. To confirm this the team evaluated the IMM model probabilities for a scenario where the ego car follows the lead car on a straight, into a turn, and then continues on the turn. As expected the IMM filter had high certainty in the CV model for the straight, high confidence in the CA model when entering the turn, and then high confidence in the CV model again once both vehicles were in the turn.
CHAPTER 3
PERCEPTION SYSTEM IMPLEMENTATION

After development of the initial sensor fusion algorithm, the team needed to implement the perception system in the vehicle. To do this communication needed to be established between the sensing components and the CAVs controller, the proprietary structure of the messages from different manufacturers had to be standardized, and the system needed to be validated in a closed course environment.

3.1 Parsing CAN Data

In order to make the data received by the CAVs controller from the sensors usable by the sensor fusion code, the team needed to write a program to convert the CAN messages into the object detection structure expected as an input to the sensor fusion program. The team was given decoding information from the manufacturers of both the radar and camera sensors for this purpose. A message could be decoded from 8 bytes of unsigned data to multiple scaled and shifted signals based on the decoding operations outlined in this supplied information. The team’s first attempt at accomplishing this task utilized the can-tools Python package. The program could parse a CAN message by its ID and call the cantools API to return a string with the decoded signal names and values. Then the string would be parsed and individual signals could be outputted. With this structure, the average time of each decode callback was 2ms with a standard deviation of 0.65ms. Since the CAVs controller could receive more than 1000 messages on one CAN channel in a single second, this program was not acceptable for real-time use.

To address this, the team utilized a feature in the cantools package to auto-generate C source code that performs all bit-wise operations necessary to decode the incoming CAN
data. This required the team to utilize pre-compiled C++ code instead of runtime interpreter
dependent Python code. After this conversion the average callback time for the decoding
function was $27\mu s$ with a standard deviation of $9\mu s$. This resulting decrease of almost 100x
in time required to parse a single CAN message led to the feature being implementable
real-time in the vehicle.

### 3.2 Standardizing Messages

Due to confidentiality agreements, the exact structure of object definitions from the manu-
ufacturers of the team’s utilized sensors will not be revealed. However, it is important
to note that there may be differences across various sensors (radar, camera, etc.), across
different manufacturers, and even across different generations of the same product. Stan-
dardizing this data into a more general format usable by multiple programs is therefore
a desirable feature. To accomplish this for the perception system, the team standardized
on the MATLAB defined `ObjectDetection` class. The documentation for this class is
publicly available on MathWorks’ website. This class contains various options for storing
data. One example is the use of Cartesian or spherical coordinates. In this implementation,
the team chose to use SI units and Cartesian coordinates with the coordinate system’s ref-
erence frame being the ego vehicle.

To increase the generality of the team’s sensor fusion code, separate functions were cre-
ated to convert radar and camera data to the MATLAB `ObjectDetection` class. This
allowed for each function to be changed independently of the sensor fusion program, lead-
ing to more flexibility and modularity during development. This also allowed for each
sensor to be easily tested independently with the sensor fusion algorithm.
3.3 ROS Communication

A final component to note before discussion of in-vehicle results of the perception system is the tool that allows the various aforementioned software programs running on the CAVs controller to communicate: ROS. ROS is a middle-ware tool that facilitates the transmission of data between programs agnostic of what language they were written in. This leads to faster development, since team members can stick with the tools they’re most familiar with. Additionally, ROS adds many debugging and visualization tools. These include the logging of all signals that are sent or received by every program in the ROS network and live plotting of signals.

3.4 In-Vehicle Testing

To validate the perception system’s functionality before public road testing, the team went to a closed course and tested the vehicle in multiple scenarios representative of real world driving. The first scenario was approaching a stopped vehicle at the maximum speed the perception system was anticipated to be used at, 60mph. For this set of tests, the vehicle was on a runway where no lane detections were available, so the lane is assumed to be straight and of constant width. The results of the first test, approaching a stationary lead vehicle can be seen in Figure 3.1. The perception system initially picks up the lead vehicle at 127m, satisfying the requirements of the system even at the maximum velocity.
To test behavior more indicative of highway driving a test was conducted where both the ego and lead vehicles would travel at higher speed and stay within the range of the perception system. The purpose of this test was to validate the consistency of detection of the lead vehicle. The results of this test can be seen in Figure 3.2. The vehicle maintains constant detection of the lead vehicle even during large acceleration and deceleration events.
The team was also interested in testing the ability to detect a vehicle in its lane. To accomplish this, the lead vehicle and the ego vehicle were accelerated to approximately 50mph with the ego vehicle following a straight path parallel to the lead vehicle but laterally offset by 10m. Then the lead vehicle would be maneuvered into the ego vehicle’s lane. Then the lead vehicle would move back out of the lane. The test would be successful if the lead vehicle is consistently tracked while in the same lane as the ego vehicle. The results of this test are shown in Figure 3.3. The lead vehicle is successfully picked up just after the 100s mark and correctly identified as out of the lane just after the 130s mark.
Finally, to confirm the implementation of the perception system in the vehicle was acceptable for use with the ACC feature, the team tested the perception system on the highway with a human driver in control. The result of this test is shown in Figure 3.4. The lead vehicle is consistently tracked when its change in velocity is relatively slow. However, there were many instances where the lead vehicle was not detected. These occurred around tighter turns where the lane detection output of the camera was not accurate enough. To fix this, the team uses the driver’s steering angle to extrapolate out a future path of the ego vehicle and use this path to define the lane boundaries. Additionally, the radar picked up overpasses or overhead road signs as a vehicle in three instances during this test. This issue is still under investigation by the team, but has been given lower priority, since the final event of the EcoCAR Mobility Challenge will not have any overpasses or overhead signs.
Figure 3.4: Perception Validation Drive on I-85
CHAPTER 4

VEHICLE CONTROLLER DESIGN

The vehicle controller can also be decomposed into two parts: a low level controller and a high level controller. The low level controller is responsible for regulating the torque produced by individual components including the internal combustion engine, electric machines, and friction brakes. This work will focus on the high level controller, which is responsible for regulating the longitudinal motion of the vehicle. From the perspective of the high level controller, the system can be described as a mass with drag forces and a commandable positive or negative torque input. Unlike the low level controller, the high level vehicle controller relies upon the output of the perception system to function. The ACC vehicle controller operates in two modes: velocity control, when a lead vehicle is not present, and distance control, when a lead vehicle is present. The switching between these two modes has been previously addressed by other researchers in multiple ways: the concept of a virtual lead vehicle and a discrete two-mode system with pre-defined switching conditions [7]. The virtual lead vehicle would eliminate discrete mode switching, but would add internal dynamics for matching the virtual vehicle to the real vehicle [8]. On the other hand, the two-mode system could lead to rapid switching behavior when the system is on a boundary between modes. To address this, a hysteresis control is implemented, where the modes will only switch when the desired following distance or speed limit are violated by some preset deadband. The optimality of this problem for fuel economy has been explored [9] as well as optimizing rider comfort [8]. Safety is also a critical element of any consumer vehicle, and the goals of safety and comfort tend to contradict each other. The former may tend to utilize more aggressive control actions where the latter would favor lower acceleration magnitude. These two concepts have been addressed by [10], which proposes a new reference command policy in the distance control mode to ensure crash
avoidance while maximizing rider comfort. This thesis will explore the formulation and implementation of the simplest of these schemes: the two mode controller.

4.1 System Requirements

As outlined by the competition, the team needed to develop a controller capable of staying within a pre-specified distance to a lead vehicle but avoid collisions at all costs. Additionally, the ACC feature would need to maintain a constant speed set by the end user in the absence of a lead vehicle, while still maintaining safe operation. In order to ensure the safe operation of the ACC feature, the team developed a set of requirements for the vehicle controller system consisting of the following: the sensor fusion system must have no active faults from sensors, the CAV controller cannot miss more than one out of 1000 CAN messages from the vehicle’s supervisory controller, when checked with a rolling counter, and the vehicle speed must be under 70mph. If any of these criteria fail, the ACC feature is automatically disabled and the CAVs controller sends a fault signal in a CAN message to the vehicle’s supervisory controller. In addition to the software safety considerations, the role of the driver must be considered. To this end, the ACC feature is automatically aborted if the driver depresses the brake pedal more than 5%. The maximum deceleration rate allowed for the controller was derived from the competition rules: -0.5g. The positive acceleration limit was decided to be 0.25g, which is the standard for public transportation in many countries [11] [12]. The limit on deceleration was motivated more by safety in contrast to the positive acceleration’s limit inspired by passenger comfort.

4.2 Vehicle Controller Overview

The requirements of the controller lend themselves naturally to two modes of operation: regulating velocity when a lead vehicle is either not present or far enough away that the concern for a possible collision is low and regulating distance when a lead vehicle is close enough. Figure 4.1 shows the setup of the distance control problem, the more complicated
of the two modes. The velocity control problem follows the same direction convention, but is only a first order system. In distance control mode, the lead vehicle velocity is treated as a disturbance to the system, since the knowledge of the lead vehicle’s behavior cannot be known by the ego vehicle beforehand. Additionally, the desired distance, $d^*$, is calculated at each time step to create a constant time gap to the lead vehicle plus a small offset to leave a gap between vehicles when at a stop. For example, when the ego vehicle is traveling at 20m/s it is desired to follow twice as far behind the lead vehicle as when traveling at 10m/s, neglecting the small offset.

Figure 4.1: Distance Controller Setup

$$v = \text{ego vehicle velocity}$$

$$d = \text{distance from front of ego vehicle to rear of lead vehicle}$$

$$d^* = \text{desired following distance}$$

$$e = \text{following distance error}$$

$$u = \text{ego vehicle torque request}$$

$$w_u = \text{disturbance forces on ego vehicle}$$

$$w_{\nu} = \text{lead vehicle velocity}$$

The velocity and distance modes can be implemented with linear feedback regulators, where the velocity regulator is designed to maintain the user-set speed and the distance
regulator is designed to maintain a constant time gap to a lead vehicle. To make these regulators function together, the logic for switching between modes based on the state of the system needed to be explored. Figure 4.2 illustrates this behavior.

Figure 4.2: ACC Mode Switching Behavior

\[ T_d = d < d^* - \delta_d \text{ and } (v <= v^* + \delta_v \text{ or } \dot{u}_d < u_v) \]

\[ T_v = (v > v^* + \delta_v \text{ and } (d >= d^* - \delta_d \text{ or } \dot{u}_v < u_v)) \text{ or } d > d^* + \delta_d \]

In the current implementation, the controller switches from the velocity to the distance regulator if the lead vehicle distance is less than the desired distance, \(d^*\). In contrast, the controller switches from the distance to the velocity regulator if the vehicle’s speed exceeds the user-set limit, \(v^*\), or if the distance to the lead vehicle becomes larger than the desired distance. To avoid rapid switching between modes, hysteresis variables, \(\delta_v\) and \(\delta_d\), are introduced.

One special case must also be considered: when the system conditions satisfy both switching criteria. More precisely, this occurs when the ACC feature is enabled while the vehicle’s speed is greater than the user-set speed and the vehicle is closer than the desired following distance to the lead vehicle. This can lead to rapid switching between modes. This behavior has been observed in previous testing and can lead to constant re-initialization of the regulators, which can cause the vehicle to accelerate to speeds much
larger than the user-set speed. The team’s current solution to this issue is to allow the controller to switch modes only if the above switching criteria is satisfied. If the conditional \( d < d^* - \delta_d \) and \( v > v^* + \delta_v \) is evaluated as true at any time step, then the controller re-initializes the regulator not currently in use one time step behind the current time, evaluates its tentative torque request at the current time step, \( \hat{u}_v \) or \( \hat{u}_d \), and only switches modes if that tentative request is less than the torque request of the current mode.

The intuition behind utilizing the lesser of the two control actions is to get the system out of this edge case as quickly as possible. When the system is in a state where the lead vehicle is closer than desired and the ego vehicle’s velocity is greater than the user-set speed, it is always necessary to slow down, since the only way to widen the distance to the lead vehicle is to decelerate, and the only way to approach the user-set speed when the vehicle’s current speed is larger is to decelerate. This does not guarantee no mode switches, but by taking the lowest control action at each time step, the system continuously approaches a region where this edge case will no longer apply. More elegant handling of this scenario will be the subject of future work for the team.

The overall architecture of the controller takes the form shown in Figure 4.3. Both regulators run in parallel and only one control action is fed to the actual vehicle.

![Figure 4.3: ACC Vehicle Controller Architecture](image)

Additionally, each regulator is augmented with an integrator state, denoted as \( \sigma_v \) and \( \sigma_d \) for the velocity and distance regulators respectively, to eliminate steady-state error. When
utilizing either the velocity or distance regulator, the integrator state variable in the other regulator may experience undesirable behavior, since that regulator’s output request is not being honored. This can lead to undesirable transients when switching between controllers. However, the integrator state variable is just a software signal and does not correspond to a physical state of the system like the vehicle’s velocity. Therefore, the designer can arbitrarily change the integrator at any time step. This degree of freedom can be used to make the torque request to the vehicle continuous during switching events.

4.3 Design and Simulation Models

The design of the linear regulators relies on a linearization of the governing dynamics of the system. In reality, the system is subject to nonlinear forces. To account for this, the team uses the linearized dynamics in calculations such as pole placement, but always uses the true dynamics in simulation and validation.

To demonstrate the validity of the team’s modeling simplifications (linearized dynamics, zero road pitch angle, constant mass), the team conducted some simulation experiments demonstrating the robustness of the controller. The pitch angle of the road and instantaneous mass of the vehicle are not directly observable by the sensors already integrated in the vehicle. The team developed a simulator for the real-world nonlinear dynamics of the system, which accounts for aerodynamic drag, rolling resistance, and the potential energy gained or lost when traversing hills. The team tested the controller for constant road pitch angles between -10deg and 10deg. The team also tested vehicle masses $\pm 25\%$ of the originally measured value. Figure 4.4 is the plot of one scenario with a 10deg decline and no change in the assumed vehicle mass. For the controller mode, 1 corresponds to velocity mode and 2 corresponds to distance mode.
The response for the model utilizing the true nonlinear dynamics can be seen to under-shoot the model tested with linearized dynamics and the reference command. However, the maximum error at any given time remains relatively low. This can be seen in the third subplot of Figure 4.4, where the RMS velocity error between the responses utilizing the linearized and true dynamics never exceeds 3m/s and in steady state always settles to a small value, under 0.2m/s. It can be seen that the largest errors occur at transient events,
such as a sudden change in the rate of the lead vehicle. More samples of tests varying the angle of the road and the mass of the vehicle are included below along with peak RMS error in velocity between the two models to illustrate the efficacy of the linear feedback controller utilized in the two-mode system.

As expected, the incline angle had a greater effect on the error of the linearization, since the force due to gravity as a function of pitch is very nonlinear for larger angles. However, the maximum error seen is still under 3 m/s for the worst scenario in Table 4.1. The team will need to validate these scenarios with hardware and the vehicle itself in the loop in order to judge the real-world generalization capabilities of the controller.

<table>
<thead>
<tr>
<th>Incline (deg)</th>
<th>Mass (%)</th>
<th>Avg RMS Vel Error (m/s)</th>
<th>Max RMS Vel Error (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10(^\circ)</td>
<td>100%</td>
<td>0.0673</td>
<td>0.6748</td>
</tr>
<tr>
<td>-10(^\circ)</td>
<td>100%</td>
<td>0.4851</td>
<td>2.0871</td>
</tr>
<tr>
<td>0(^\circ)</td>
<td>75%</td>
<td>0.1974</td>
<td>1.8399</td>
</tr>
<tr>
<td>0(^\circ)</td>
<td>125%</td>
<td>0.1766</td>
<td>0.9433</td>
</tr>
<tr>
<td>0(^\circ)</td>
<td>100%</td>
<td>0.1714</td>
<td>1.4296</td>
</tr>
</tbody>
</table>

Table 4.1: Performance Errors for Mismatched Parameters

4.4 Simulation Validation of Controller

Finally, to validate the functionality of the vehicle controller in each mode and when switching between modes, a simulation test-bed was developed in MATLAB. Base cases were defined where the lead vehicle behavior was chosen to lock the controller into only one mode. For all test cases shown in this section, the user set speed is 25m/s, the time gap setting is 2s, and the torque limits for the vehicle are -4000Nm and 2000Nm.

Figure 4.5 shows the simplest test case where there is no lead vehicle present. The controller is always in velocity mode and has a smooth response, accelerating the vehicle to the user set speed.
Figure 4.5: ACC Controller Simulation Velocity Mode

Figure 4.6 shows a test case where a lead vehicle accelerates, maintains speed, then decelerates. The ego vehicle is initialized in such a way that the controller begins in distance control mode. The controller stays in distance mode for the remainder of the test and follows the lead vehicle’s velocity trajectory with a small delay.
Figure 4.7 shows a test case where a lead vehicle is traveling slower than the user-set speed but starts far away from the ego vehicle. This case is designed to analyze switching behavior of the ACC feature when changing modes of operation from velocity control to distance control. The controller’s response is to accelerate to try and achieve the user-set speed in velocity mode. Then when the lead vehicle is close enough, the controller correctly switches the mode to distance control and brakes to establish a constant time gap.
In this case, the benefits of integrator reset are apparent, since the acceleration of the vehicle exhibits a continuous response.

Figure 4.7: ACC Controller Simulation Velocity to Distance Gradual Switch

Figure 4.8 shows another test case where the controller is expected to change modes of operation from velocity control to distance control. However in this case, a lead vehicle cuts into the lane of the ego vehicle, resulting in a step change in the following distance state. At the time of the cut-in event the controller correctly switches modes to distance.
control and begins to apply a braking force until the desired time gap to the lead vehicle has been established.

Figure 4.8: ACC Controller Simulation Velocity to Distance Cut-In Switch

Figure 4.9 shows a test case where a lead vehicle starts closer to the ego vehicle than the desired distance, which results in the controller initializing in distance mode. However, the lead vehicle is traveling at a speed higher than the user-set speed for the duration of the test, which quickly leads to the following distance growing. This causes the ego vehicle to
accelerate until it violates the user-set speed plus a hysteresis tolerance. Then the controller correctly switches the mode to velocity control and regulates to the user-set speed for the remainder of the test.

Figure 4.9: ACC Controller Simulation Distance to Velocity Gradual Switch

Figure 4.10 shows another test case where the controller is expected to change modes of operation from distance control to velocity control. However, in this case, similar to the lead vehicle cut-in test, a lead vehicle cuts out of the lane of the ego vehicle, resulting in a
step change in the following distance state. At the time of the cut-out event the controller correctly switches modes to velocity control and regulates the vehicle velocity to the user-set speed.

Figure 4.10: ACC Controller Simulation Distance to Velocity Cut-Out Switch
CHAPTER 5
CONTROLLER IMPLEMENTATION

With a functional perception system and an understanding of the structure of the controller, the team could move to final integration of the ACC feature in the vehicle. However, before all systems could be tested together, the controller needed to be validated in isolation. The same procedure for communicating between hardware components using a CAN network and communicating between software programs utilizing a ROS interface discussed in Chapter 3 applies to the controller integration. Separate from the communication interfaces, implementing the two-mode controller in the vehicle required some modifications.

5.1 Hardware Considerations

When considering the target hardware to run the vehicle controller, actuator limits and communication interface delays needed to be considered. From the competition, the maximum deceleration of the vehicle was limited to -0.5g, and the team decided on a lower positive acceleration limit of 0.25g for rider comfort. Empirically, the team validated that the vehicle is capable of generating acceleration in excess of these bounds. Therefore traction force limits were not an implementation concern for the team. However, an element that mandated a change in the controller structure was the nonlinear response of the vehicle’s engine control unit at low speeds. In consumer vehicles with automatic transmissions, when the driver removes their feet from both the brake and accelerator pedals, essentially requesting zero torque, the vehicle creeps forward. The linear two-mode controller developed in Chapter 4 would rely on the vehicle not accelerating at low speed when attempting to come to a stop. The integral action of the team’s controller would eventually compensate for this creeping behavior, but it was observed in a closed course environment to take too long. Rather than cranking up the gains, the team opted to create a new mode “Low Speed
Mode” which utilizes the same velocity regulator as the existing velocity control mode, but overrides the reference command to 0m/s. Additionally, the team created a “Stop and Hold” mode which essentially holds the brakes while the vehicle is stopped. These two additional modes are only used when the vehicle attempts to come to a stop. Figure 5.1 illustrates the switching behavior of the modified controller.

![Figure 5.1: Modified ACC Mode Switching Behavior](image)

\[
T_{v,d} = d < d^* - \delta_d \text{ and } (v \leq v^* + \delta_v \text{ or } \dot{u}_v < u_v)
\]

\[
T_{d,v} = (v > v^* + \delta_v \text{ and } (d \geq d^* - \delta_d \text{ or } \dot{u}_v < u_d)) \text{ or } d > d^* + \delta_d
\]

\[
T_{d,ls} = v < 2 \text{ and } d < d^* + \delta_d
\]

\[
T_{ls,v} = d > d^* + \delta_d
\]

\[
T_{ls,sh} = v < 0.5 \text{ and } d < d^* + \delta_d
\]

\[
T_{sh,v} = d > d^* + \delta_d
\]
For the purposes of visualization, the various controller modes are encoded as an enum with the following values:

\[
\begin{align*}
0 &= \text{Manual Control - ACC disabled} \\
1 &= \text{Velocity Control} \\
2 &= \text{Distance Control} \\
3 &= \text{Low Speed Mode} \\
4 &= \text{Stop and Hold Mode}
\end{align*}
\]

### 5.2 Closed Course Validation

To validate this controller in isolation, the team completed a series of arbitrary acceleration and braking tests. These included accelerating from a stop and gradually ramping up the top speed until the full range of the feature was covered. Figure 5.2 shows the response of the vehicle when the user-set speed is 40mph and there is no lead vehicle present. The vehicle can be seen to accelerate to the desired velocity from a stop and stays exclusively in velocity control (mode 1) after the ACC feature is enabled.
This same procedure was followed for testing bringing the vehicle to a stop and for following a lead vehicle. Figure 5.3 shows the response of the controller during a stop and go test, where the feature is initialized with both vehicles close together and the lead vehicle stationary. Then the lead vehicle accelerates away, while the ego vehicle autonomously follows. Finally the lead vehicle comes to a stop and the ego vehicle autonomously comes to a stop behind it. When the feature is initialized around the 66s mark, the controller correctly switches to low speed mode (mode 3) then stop and hold mode (mode 4), since
the lead vehicle is too close. The reason for not going straight to stop and hold mode is that the feature was initialized while the vehicle had a nonzero velocity, creeping forward. Once the lead vehicle accelerates away, the controller switches to velocity control (mode 1), since the lead vehicle is accelerating away faster than the ACC feature is capable of and the following distance quickly grows greater than the desired distance. When in velocity control, the controller regulates the vehicle’s velocity to the user-set speed, which was 60mph (26.82m/s) in this test. The lead vehicle accelerates to a velocity lower than the user-set speed, which leads to the ego vehicle catching up and entering distance control (mode 2) when the distance to the lead vehicle is less than the desired distance. Finally, the lead vehicle comes to a stop and the ego vehicle slows down behind it, eventually entering low speed mode (mode 3), before the feature is disabled by the user at 129s. The four gaps in the lead vehicle detection were due to the lack of lane detections in the closed course testing environment. Since no lane markings were physically present, the driver’s steering wheel angle was used to define a window for the lane. When the driver made small adjustments, this led to the lead vehicle not being selected as the output of the perception system, despite an object track existing for it. This issue was resolved in on road testing by using the camera to detect lanes and select a lead vehicle more accurately from all detections.
5.3 Controller Simulation for Validation

In the event where a change to the controller is considered for deployment in the vehicle, the team must validate that its performance meets the design requirements. To check this, the team increased the functionality of the ACC simulator code to include base cases for operating in each of velocity and distance mode, switching between each mode, using competition supplied lead vehicle behavior, and even randomly generating lead vehicle
behavior to stress a proposed controller. Figure 5.4 shows the output of this simulator for the test case of the competition supplied highway drive cycle. This test case stresses the controller with abrupt starts and stops and covers the full speed range of the feature. Here the full set of mode switching can be observed. The controller is shown to successfully avoid collisions and keep up with the lead vehicle. The user-set speed limit is set to the max value of the lead vehicle’s speed.

Figure 5.4: ACC Highway Drive Cycle Simulation
5.4 Parameter Identification

After the base functionality of the controller was established, the team sought to further improve the accuracy of the simulation model it was using to judge the suitability of each controller. To do this the team performed parameter identification experiments to verify the plant model used in simulation. Tests were conducted where the vehicle was asked to follow arbitrary velocity profiles and parameters were found via least squares minimization in post. Figure 5.5 shows one of these tests, where the vehicle was asked to accelerate to 60mph then brake to a stop. After 11 such tests, the average of the identified parameters was taken and used in the simulation test-bed. The results of the parameter identification is shown in Table 5.1. A significant factor to note, which led to experimental inaccuracies, was the incline of the road used to gather this data.

Figure 5.5: ACC Parameter Identification
### Table 5.1: Parameter Identification of Chevy Blazer

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Analytical</th>
<th>Empirical</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>1.55e-2</td>
<td>1.28e-2</td>
</tr>
<tr>
<td>$\beta$</td>
<td>4.49e-4</td>
<td>8.42e-4</td>
</tr>
</tbody>
</table>

5.5 **Gain Selection**

Once the controller had reached a baseline level of functionality, the team turned its attention to improving performance. The primary goal of the final competition event for the ACC feature is minimizing energy consumption, and a secondary goal was to stay within a velocity dependent distance envelope. With these requirements in mind, the team evaluated the performance of the ACC feature in simulation for many different closed loop pole locations and time gap settings, with the goal of finding a set of pole locations and a time gap setting to improve the vehicle’s fuel economy and adherence to the competition distance envelope. Figure 5.6 shows the result for testing 1000 combinations of time gap setting, $\tau$, and closed loop pole location for all poles in the distance regulators, $\lambda_d$. The best time gap and pole location were found as $2.50s$ and $s = -0.55$ respectively.
Finally, the team needed to test the ACC feature on public roads to validate its performance before the final competition in May 2022. Although some edge cases are still handled undesirably such as the vehicle incorrectly identifying an overpass as a lead vehicle or the controller slamming on the brakes when another vehicle cuts into its lane, the team has been able to refine the feature to the point where it is able to complete all requirements in the scope of the competition. As a demonstration of the high level of functionality achieved, Figure 5.7 shows a 60mi continuous run of the ACC feature on I-85 south of Atlanta. The vehicle was successfully able to follow other vehicles during regular highway driving and kept a continuous track of the lead vehicle. It additionally handled two cut-in events.
Figure 5.7: ACC Endurance Run
6.1 Conclusion

In summary, the Georgia Tech team was able to develop a perception system to fuse sensor data into a more accurate representation of the world, create a controller to regulate distance to another vehicle or velocity when no lead vehicle is present, and implement these systems in a consumer vehicle. At this point, the ACC feature has been tested for over 580mi on public roads and many more in closed course environments. The team has started exploring other automated features as well and has taken steps to ensure the knowledge gained in the development of this feature will not be lost with team member turnover.

6.2 Future Work

Possible future extensions of this work include refinement of the perception system to be more robust to obstacles found on public roads, including overpasses and drivers that cut into the ego vehicle’s lane aggressively. Additionally feed forward disturbance rejection could be added to improve the controller’s performance when the lead vehicle exhibits large acceleration or deceleration. The current implementation of the perception system contains an estimate of the lead vehicle’s velocity as part of its state measurement, so the necessary signals are already present in the CAVs controller. Additionally, novel control schemes for implementing the ACC feature have been proposed, including the utilization of barrier functions for safety applications [13] [14]. This could be an area of future research for the team, as safety is always given high priority in automotive applications. A final feature that this ACC feature could be used for is integration with connected mobility features. These include Connected Adaptive Cruise Control (CACC), where multiple vehicles travel in a
platoon and knowledge of each vehicle’s acceleration can be communicated via V2X radios [15] [16] [17], and Connected Corridor Traversal, where a vehicle can know information about traffic signals future states ahead of time [18] [19].
REFERENCES


