5-13-2022

Automotive sensor fusion systems for traffic aware adaptive cruise control

Jonah T. Gandy
jonahgandy@gmail.com

Automotive sensor fusion systems for traffic aware adaptive cruise control

By

Jonah T. Gandy

Approved by:

John E. Ball (Major Professor)
Randolph F. Follett
Chaomin Luo
Qian (Jenny) Du (Graduate Coordinator)
Jason M. Keith (Dean, Bagley College of Engineering)

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical and Computer Engineering
in the Department of Physics and Astronomy

Mississippi State, Mississippi

May 2022

Name: Jonah T. Gandy

Date of Degree: May 13, 2022

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: John E. Ball

Title of Study: Automotive sensor fusion systems for traffic aware adaptive cruise control

Pages of Study: 86

Candidate for Degree of Master of Science

The autonomous driving (AD) industry is advancing at a rapid pace. New sensing technology for tracking vehicles, controlling vehicle behavior, and communicating with infrastructure are being added to commercial vehicles. These new automotive technologies reduce on road fatalities, improve ride quality, and improve vehicle fuel economy. This research explores two types of automotive sensor fusion systems: a novel radar/camera sensor fusion system using a long short-term memory (LSTM) neural network (NN) to perform data fusion improving tracking capabilities in a simulated environment and a traditional radar/camera sensor fusion system that is deployed in Mississippi State's entry in the EcoCAR Mobility Challenge (2019 Chevrolet Blazer) for an adaptive cruise control system (ACC) which functions in on-road applications. Along with vehicles, pedestrians, and cyclists, the sensor fusion system deployed in the 2019 Chevrolet Blazer uses vehicle-to-everything (V2X) communication to communicate with infrastructure such as traffic lights to optimize and autonomously control vehicle acceleration through a connected corridor.

Key words: thesis, Sensor Fusion, Automotive, Adaptive Cruise Control, Vehicle-to-Everything

DEDICATION

I dedicate this thesis to my Mom and Dad, Christie and Thomas Gandy.

ACKNOWLEDGEMENTS

I would first like to acknowledge my major advisor Dr. John Ball. Dr. Ball has been a fantastic advisor and teacher during my graduate studies at Mississippi State. This research was inspired through his Sensor Processing for Autonomous Vehicles class, and I am very thankful to have him as a major advisor.

I would also like to acknowledge the Mississippi State EcoCAR team. The team members have all provided the necessary means for the this research be developed and tested. I would also like to acknowledge to the team for the structure, advice, and encouragement I received during the development of this sensor fusion system. I have gained very valuable skills by being a part of this team, and I am very grateful.

I would like to thank all the entities who helped make the EcoCAR Mobility Challenge a reality. A special thanks to the US Department of Energy, Argonne National Lab, General Motors, MathWorks, NXP, and all the competition and team sponsors for providing the software, hardware, and support necessary for this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 Motivation

1.1.1 Advanced Driver Assistance Systems

Advanced driver assistance systems (ADAS) have become prominent in today's automotive industry. Original equipment manufacturer's (OEMs) are integrating ADAS technologies into vehicles to improve driver safety, comfort, and fuel economy. Traffic related accidents have high monetary and human costs. In 2018 over 38,000 fatalities occurred due to automotive crashes in the U.S. alone [9]. To reduce this number a large effort is taking place to include ADAS technologies such as emergency braking, lane keep assist (LKA), vehicle-to-everything (V2X) communication, and adaptive cruise control (ACC) in today's automobiles. ADAS not only helps reduce on road accidents, it also improves the driver's ride quality and driving experience.

1.1.2 SAE Levels of Driving Automation

The Society of Automotive Engineers (SAE) has defined six levels of automation. The SAE J3016 standard defines each level of driving automation by defining both the human's responsibility when using the autonomous driving features and the vehicle's autonomous driving capability [30]. SAE driving autonomy level zero defines a vehicle with no active autonomous driving systems. SAE level one defines a vehicle with one active ADAS system, such as LKA or ACC. For a vehicle to be classified as SAE level two, it must have two active ADAS systems active, such as LKA and

ACC. It is important to note that for autonomy levels one and two, the driver must be engaged and actively driving at all times and is responsible for the cars action's. SAE level three defines a vehicle that can operate autonomously in limited conditions, such as on-ramp to off-ramp highway driving, or perform autonomous driving in certain geographical locations. In a level three autonomous vehicle, the driver does not have to be actively driving, but needs to be on alert for when the vehicle hands control back to the driver. SAE level four does not require any driver intervention and is able to operate fully autonomously, but is limited to certain locations or operating conditions. Finally, SAE level five defines a vehicle that can operate in all conditions, drive in all on-road locations, and operate without any driver intervention.

1.1.3 Challenges in Autonomous Driving

There are many challenges in autonomous driving for the vehicle to be in a safe, comfortable, and efficient state. An Autonomous driving system can be broken down into four subsystems: sensing, perception and tracking, path planning and control, and acting. The sensing subsystem manages all of the sensors on the vehicle and performs light operations on sensor data so that it can be used by the perception and tracking subsystem. The perception and tracking subsystem's goal is to use the sensor data to recreate the environment around the vehicle as accurately as possible. This gives the vehicle the information it needs to make a decision on what areas are drivable and what areas are not drivable. The path planning and control subsystem defines the path on which the vehicle must travel. The path is then converted into a set of steering and acceleration commands to drive the vehicle. Finally, the acting subsystem is the hardware actuators on the vehicle that physically control the vehicle such as the friction brakes or motor.

All of the subsystems must work in harmony with one another, or issues quickly begin to quickly occur. For instance, if the sensing subsystem does not detect an in-path obstacle, the vehicle will never know it is there. This could cause a vehicle crash. Another example could be the path planning subsystem failing to create an optimal path, which could result in inefficient motor behavior or uncomfortable ride quality. On top of working together, there must be a minimal latency between subsystems. If the delay is too great, the vehicle might not make a maneuver quickly enough to avoid an accident.

1.2 Contributions

This thesis contributes to the advancement of autonomous driving technology in three categories: i) sensor fusion, ii) V2X application, and iii) procedures to develop and test autonomous driving systems.

In the first portion of this thesis, a novel sensor fusion system based on the LSTM neural network is developed and tested in a simulated environment. A long short-term memory (LSTM) network has been proven to yield comparable results to state of the art state estimation filters such as the extended Kalman filter (EKF) and interacting multiple model (IMM) filter.

In the second portion of this thesis, a systematic approach for testing and developing an autonomous driving perception system is discussed. A detailed discussion using model in the loop (MIL), component in the loop (CIL), software in the loop (SIL), hardware in the loop (HIL), and vehicle in the loop (VIL) testing environments is discussed and applied to a radar-camera sensor fusion system. This process describes how to navigate from the early design phase of building system requirements to successfully validating the system on road ACC system.

In the final portion of this thesis, V2X radios are explored in a vehicle-to-intersection aware ACC system. By using CIL testing methodologies, a V2I system is integrated into a radar-camera sensor fusion system. This system is transitioned to on road testing, where the system safely and successfully navigates a single V2I intersection.

1.3 Publications

This thesis contains content from the following Publications:

1. Jonah Gandy,Dr. John E. Ball. (2022). Long Short-Term Memory Networks for Automotive Sensor Fusion. SPIE Defense and Commercial Sensing, April 2022
2. Amine Taoudi, Jonah Gandy, Vance Hudson, Chaomin Luo, Randolph F. Follett. (2022). Model Based Design and Verification of Automated Driving Features Using XIL Simulation Platforms. SAE

CHAPTER II

BACKGROUND

2.1 Automotive Sensors for Perception

Automotive sensors come in a variety of flavors to address many of the challenges in autonomous vehicle perception. In robotic literature, sensors are broken down into two types: proprioceptive and exteroceptive[11]. A proprioceptive sensor monitors the internal state of the vehicle or robot. Exteroceptive sensors gather measurements of a vehicle's surrounding environment, such as other vehicles, lane information, and obstacles. This section will cover three proprioceptive sensors used in modern automotive autonomy: camera, radar, and LiDAR. Table 2.1 summarizes each sensor's strength's and weaknesses.

2.1.1 Camera Sensor

A camera sensor is typically passive sensor that receives photons and creates a digital image which is used for target classification and range measurement. Cameras are a common sensor on autonomous vehicles due to low cost and the sensor's ability to represent color and textures in a digital format. Camera systems can be as basic as a backup camera to complex stereo cameras used for accurately measuring objects in 3D space. Monocular cameras are a single-camera system that typically excel in target classification, lane line tracking, and lateral measurements. A weakness of a monocular camera system is depth perception, where the camera must use a technique known

Table 2.1

Radar and Camera Sensor Weaknesses and Strengths

Performance Trait	**Camera**	**Radar**	**LiDAR**
Longitudinal Measurement	Weak	Strong	Strong
Lateral Measurement	Strong	Average	Strong
Velocity Measurement	Very Weak	Strong	Strong
Object Classification	Very Strong	Weak	Average
Cost	Strong	Neutral	Very Weak
Weather Performance	Weak	Strong	Average

as structure from motion [37] to translate obstacles from a 2D space into 3D space. Two main disadvantages of a camera is its poor performance in low light environments and degradation in rainy and foggy environments. The camera sensor also requires a great deal of processing power to train neural networks for obstacle classification as well as to run the networks in real-time in the vehicle. A common practice is to fuse a camera sensor with a range detection sensor such as a radar or LiDAR sensor. This practice creates a "complementary" pair which is useful for accurately perceiving the environment around the vehicle [5].

2.1.2 Radar Sensor

The radio detection and ranging (radar) sensor is a mature technology that transmits and receives radio frequencies (rf) to measure a target's range and velocity. Common frequency bands for automotive radar include 24 Ghz, 77 Ghz, and 79 Ghz. Today, most automotive radar sensors operate in the 77 Ghz and 79 Ghz frequencies. Automotive radar sensors can also be broken down into three range classifications: short-range radar (SRR) with a detection range of 0 - 50 m, medium-range radar (MRR) with a detection range of 50 - 100 m , and long-range radar (LRR)

with a detection range of 100 - 200 m [38]. Relative to other automotive sensors, radar sensors excel at range and velocity measurement, maintain consistent performance in fog and rain, and are relatively cheap compared to most LiDAR sensors.

2.1.3 LiDAR Sensor

The light detection and ranging (LiDAR) sensor is a relatively novel technology in the automotive space. In 2004, the winning vehicle in the DARPA Grand Challenge used a LiDAR sensor [34]. LiDAR uses pulses of light to calculate distances derived from the pulse's time of flight (ToF). In the automotive space, there are three types of LiDAR: rotary-based mechanical LiDAR, scanning solid-state LiDAR, and full solid-state LiDAR [32]. A rotary mechanical LiDAR typically provides a 360 degree 3D point cloud. A solid-state LiDAR uses Microelectromechanical (MEMS) systems rather than a rotary mechanism. Because a solid-state LiDAR is not spinning, solid-state LiDARs do not provide a 36 degree field of view. Finally, the full solid-state LiDAR do not use any mechanical systems. A full solid-state LiDAR uses a photodetector (similar to a camera sensor) to capture light pulses. Due to the current state of photodetector technology, full solid-state LiDARs are limited to ranges of 50 - 100 m [32]. In the current state of technology, LiDAR sensors are expensive compared to radar and camera sensors, limiting their use in commercial vehicles.

2.2 Sensor Fusion

There are many ways to approach sensor fusion, but a traditional approach involves multiple homogeneous and heterogeneous sources measuring a system's surroundings. A sensor fusion system that everyone understands is the human body. Human's have six senses, all contributing to an individual's ability to perceive and interpret the environment. The same analogy can be drawn

for an autonomous vehicle, where different sensors measure the surrounding environment to create an accurate picture of the environment. In the automotive case, there are three levels to define the types of sensor fusion [12].

1. Low-level fusion: Raw data such as signals or pixels are used during the sensor fusion process. This is also known as data fusion.
2. Medium-level: Data includes features such as pattern, classification, position, shape, or lane information. This is also known as feature fusion.
3. High-level: This level is known as decision-level fusion, where the sensor fusion performs probabilistic approaches such as Bayesian methods to fuse data together.

2.2.1 Data Association

Data association is the process of associating sensor detections with the same object over time. This is a very important component in a sensor fusion system and depending on sensor performance, data association algorithms must be chosen carefully and tuned towards available sensors. In this paper, two assignment algorithms to perform association were explored: Munkres Assignment and Suboptimal Nearest Neighbor (SNN) [27] [28] which both aim to solve the data association problem.

The SNN algorithm uses a gating procedure to throw away out of bounds detections. Gating is the process of defining a boundary around an object and throwing away any detections that appear outside of an object's radius. A boundary can either be defined as a circle or a rectangle. A simple gating equation seen in eq. (2.1) was used to change the gating boundary based on longitudinal distance. This equation assumes that as objects move further away in the longitudinal direction, sensor measurements will be less accurate. The constants in eq. (2.1) were discovered empirically through simulation.

$$gating\ radius = \begin{cases} object_{long.\ distance} < 50 & , object_{long.\ distance} * 0.1 \\ object_{long.\ distance} \geq 50, 4 \end{cases} \tag{2.1}$$

Eq. (2.1) ensures that as objects move closer, the gate becomes smaller and accurate measurements are required to move forward to detection assignment. When the longitudinal distance is greater than 50, the gating radius is set to four, as seen in eq. (2.1). In the case where detections are within the gating boundary, an association algorithm will assign the best detections to an object. SNN algorithm then assigns detections with the gating boundary to the track. It is important to note that detections closer to the vehicle are assumed to be more accurate (according to sensor model characteristics). A scaling factor of 0.1 was chosen to reduce the size of the gating boundary as objects get relatively closer to the vehicle. This scaling factor ensures that unsuitable detections do not get associated with existing tracks. At a longitudinal distance of 50 meters, the scaling factor is dropped. It is assumed that sensor detections at a longitudinal distance of 50 meters will have a greater measurement error. By dropping the scaling factor, the data association algorithm associates more detections to existing tracks.

The Munkres Assignment algorithm uses the Hungarian method to assign detections to tracks [27]. In place of a gating mechanism, a cost matrix is used. The cost matrix is created using the Euclidean distance of each detection to each target. The algorithm then uses the Hungarian method to assign the best detection to each track. It is important to note that the Munkres assignment algorithm solves for a global solution where the SNN algorithm solves for a local solution.

Global and local solutions become apparent when two vehicles are in close proximity. In the case where two vehicles are close together, a local solution could associate a non-optimal detection

with a track, where a global solution will associate optimal detections tracks. A common trade-off between algorithms that solve for local and global solutions is computation speed. An algorithm that solves for a global solution often requires more computational power to solve for the global solution.

2.2.2 State Estimation Filters

State estimation filters provide estimations of where an object is by using a motion model and removing measurement noise. State estimation filters are multiple types of state estimation filters, but one of the most popular filters is the Kalman Filter, originally proposed by Rudolph Kalman [23]. The Kalman Filter blends the measurement uncertainty and the state transition model together. If correctly setup and tuned, this removes noise and uncertainty from a sensor(s) measurement. While the Kalman Filter is ideal for linear system state estimation, a well-known weakness is its application to non-linear systems. A second variant of the Kalman Filter was proposed in 1997, known as the Extended Kalman Filter (EKF), that addressed non-linear systems [1]. A third vairent, known as the interacting multiple model (IMM) filter uses multiple Kalman Filters with different motion models. This allows for the IMM filter to "blend" multiple different types of state estimation motion models together. This section will provide background on three filters: the original Kalman Filter, the EKF, and the IMM filter.

2.2.2.1 Kalman Filtering

The original Kalman Filter is composed with eq. (2.2) and is designed for linear applications with gaussian noise distribution [3]. The goal of the Kalman Filter is to take a system model and uncertain measurement matrix, y , as an input and output the ideal system state. The Kalman Filter

system design equations are seen in eqs. (2.2) and (2.3). Two vectors x_k and y_k represent the state vector and output vector respectively. The notation below uses k as the current discrete time step, $k-1$ as the previous discrete time step, and $(k|k-1)$ read as "k *given* $k-1$", that is, data at time step k, given $k-1$. In eq. (2.3), an observation matrix H can also be seen.

$$x_k = F_{k-1}x_{k-1}G_{k-1}u_{k-1} + w_{k-1} \tag{2.2}$$

$$y_k = H_k x_k + v_k \tag{2.3}$$

To implement the Kalman Filtering algorithm, a series of equations are used to provide prediction and update functionality. The first step in the Kalman Filtering algorithm is the state estimation calculation, $\hat{x}$, and the error covariance matrix, P_0, calculation. Once the filter is initialized, it makes a state update using the discrete equation in (2.4). F is the state transition matrix, which in the sensor fusion case, is a kinematic motion equation for nearly constant velocity (NCV) or nearly constant acceleration (NCA). Both motion models are seen in (2.5) and (2.6), where equation (2.5) is NCV and equation (2.6) is NCA. A "nearly" constant motion model is assumed because capturing a perfect constant motion model would be very computationally expensive.

$$\hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1} + G_{k-1}u_{k-1} \tag{2.4}$$

$$F_{cv} = \begin{bmatrix} x_{k+1} \\ vx_{k+1} \\ y_{k+1} \\ vy_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ vx_k \\ y_k \\ vy_k \end{bmatrix} \tag{2.5}$$

$$F_{ca} = \begin{bmatrix} x_{k+1} \\ vx_{k+1} \\ ax_{k+1} \\ y_{k+1} \\ vy_{k+1} \\ ay_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & T & \frac{1}{2}T^2 & 0 & 0 & 0 \\ 0 & 1 & T & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & T & \frac{1}{2}T^2 \\ 0 & 0 & 0 & 0 & 1 & T \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ vx_k \\ ax_k \\ y_k \\ vy_k \\ ay_k \end{bmatrix} \tag{2.6}$$

The formulation of the error covariance matrix, P, is shown in (2.7). The process noise matrix, Q, is modelled to represent any noise outside of the system. In the autonomous vehicle case, outside noise could be gusts of wind, road conditions, or a change in road grade.

$$P_{k|k-1} = F_{k-1} P_{k-1} F_{k-1}^T + Q_{k-1} \tag{2.7}$$

One of the most important variables in the Kalman Filter equations is the Kalman gain, K. The Kalman gain controls how much the prediction equation 'blends' the sensor measurements and the system model. In eq. (2.8), the Kalman gain is calculated using the measurement noise covariance matrix, R, and the measurement observation matrix H. The measurement noise covariance matrix models the noise characteristics from the sensor.

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1} \tag{2.8}$$

Finally, the Kalman Filter algorithm outputs a state estimate, $\hat{x}_k$, in (2.9). The equation output is controlled by the Kalman gain and outputs an optimal state estimation for the current time step. Along with the state estimation, an update state error covariance matrix, P_k, is calculated in (2.10).

$$\hat{x}_k = \hat{x}_{k|k-1} + K_k(z_k - H_k\hat{x}_{k|k-1}) \tag{2.9}$$

$$P_k = (I - K_k H_k)P_{k|k-1} \tag{2.10}$$

2.2.2.2 Extended Kalman Filtering

Sensor fusion algorithms typically need non-linear equations to properly perform vehicle tracking. An example of non-linear motion is a vehicle changing speeds in a constant velocity case. If the Kalman Filtering model is a constant velocity model and the tracked vehicle is changing velocities, the vanilla Kalman Filter would not be able to accurately perform state estimation for the tracked vehicle. A popular method to perform state estimation for the non-linear case is to use the EKF. The EKF solves non-linear systems using a set of Kalman Filtering equations modified with a Jacobian matrix.

To perform non-linear state estimation, the EKF linearizes measurements and state points at each discrete timestep. For each timestep, a Jacobian matrix is computed so that linear estimation can occur. The EKF system design equations in (2.11) and (2.12) are very similar to the Kalman Filter, the only difference is the utilization of the Jacobian matrix to help perform linearization.

$$x_{k|k-1} = f(\hat{x}_{k-1}, u_{k-1}) \tag{2.11}$$

$$y_k = h(x_k) + v_k \tag{2.12}$$

The Jacobian matrix is composed of the partial derivatives of the transition matrix, F, and the observation matrix, H as seen in Eqaution (2.13) and (2.14).

$$F = \frac{\partial f(x_k, u_k)}{\partial x}|_{x_k, u_k} \tag{2.13}$$

$$H = \frac{\partial f(\hat{x}_k)}{\partial \hat{x}}|_{\hat{x}_k} \tag{2.14}$$

2.2.2.3 Interacting Multiple Model Filtering

The IMM algorithm is a Bayesian filtering method for recursive state estimators, such as the Kalman Filter [10]. The IMM algorithm uses a Markovian chain algorithm to predict the probability of the current and future system model. By allowing individual models to interact and blend, an accurate state estimation for non-linear tracking scenarios can be achieved. An example of a set of motion models is seen in set m from eq. (2.15), where a NCV and NCA motion models are used to represent rudimentary movements of a tracked object. Each motion model in set m is known as a mode. Each motion model's probability is calculated using eq. (2.16) and stored in a vector named μ in eq. (2.17).

$$m = \{nearly\ constant\ velocity,\ nearly\ constant\ acceleration\} \tag{2.15}$$

$$\mu_i = P(m_i|Z) \tag{2.16}$$

$$\mu_i = \{\mu_1, \mu_2\} \tag{2.17}$$

The IMM algorithm transitions between modes by formulating a set of hypotheses generated from the Markov chain algorithm in matrix M, or the transition probability matrix. This matrix allows the IMM to compute mode probabilities and transition probabilities. To compute this, the total probability theorem in eq. (2.18) is used. In the context of the IMM algorithm, this can be represented with the summation as seen in eqs. (2.19) and (2.20).

$$P(A) = \sum P(A|B)P(B) \tag{2.18}$$

$$\begin{bmatrix} \mu_1 & \mu_2 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} = \begin{bmatrix} \mu_1 m_{11} + \mu_2 m_{21} & \mu_1 m_{12} + \mu_2 m_{22} \end{bmatrix} \tag{2.19}$$

$$\overline{c_j} = \sum_{i=1}^{N} \mu_i M_{ij} \tag{2.20}$$

An important part of the IMM algorithm is interaction between each filter. This interaction between filters "blends" the state estimations. It also gives each filter more information about what is happening, preventing filter divergence. In the context of the IMM, this is called mixing probabilities. To do this, the IMM computes a mixing weight, w_{ij}. Weights are formulated from the dot product of each filter's probability from u_i and M_{ij} shown in eq. (2.21). Weights adjust the state and covariance matrix of each filter according to filter likelihood. Eqs. (2.22) and (2.23) show the weighted covariance matrix, P_j^m and state, x_j^m.

$$w_{ij} = \|\mu_i * M_{ij}| \tag{2.21}$$

$$x_j^m = \sum_{i=1}^{N} w_{ij} x_i \tag{2.22}$$

$$P_j^m = \sum_{i=1}^{N} w_{ij} [(x^i - x_i^m)(x^i - x_i^m)^T + P_i] \tag{2.23}$$

Next, each filter within the IMM algorithm performs a prediction with adjusted states and covariances as shown in eqs. (2.24) and (2.25). In this example, the set of filters used in the IMM are Kalman Filters.

$$\overline{x}_j = F_j x_j^m \tag{2.24}$$

$$\overline{P_j} = F_j P_j^m F_j^T + Q_j \tag{2.25}$$

In final step of the IMM algorithm, the IMM makes a final state estimate, x, using the product of each probability and state estimate from each filter. The process equations are shown in eqs. (2.26) and (2.27).

$$x = \sum_{j=1}^{N} \mu_j \overline{x}_j \tag{2.26}$$

$$P_j = \sum_{j=1}^{N} \mu_j \left[(\overline{x}_j - \overline{x})(\overline{x}_j - \overline{x})^T + \overline{P}_j \right] \tag{2.27}$$

A full process flow of the IMM algorithm is seen in Figure 2.1, where a NCV and NCA Kalman Filter are used for the filters.

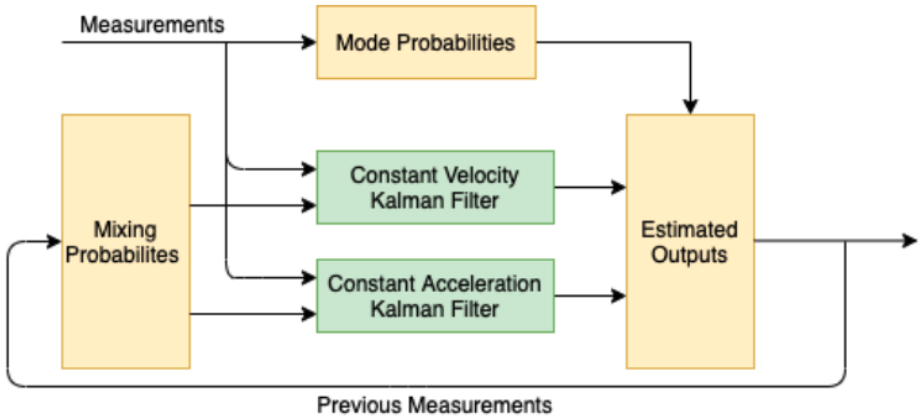


Figure 2.1

IMM Process Flow

2.2.3 Object Tracking Framework

An object tracking framework is a methodology to initialize, maintain, and delete tracks in a sensor fusion algorithm. Vehicle tracks in a perception system must be maintained overtime, where attributes for each track such as state, lane, classification, track ID, and track status are updated at the sensor fusion algorithm time. When a new obstacle is first detected by a sensor, it is initialized as a tentative track, where it must satisfy a confirmation threshold. The confirmation threshold defines a threshold of $X\ out\ of\ Y$ that must be met to confirm a track. When a track is confirmed, it can be assumed that the track represents a real obstacle and not a false positive [26].

Confirmed tracks are subject to a deletion threshold where a $X\ out\ of\ Y$ detection association threshold is applied. Sensor measurements are imperfect and asynchronous. In a tracking algorithm this must be accounted for using a method called coasting. Coasting is when a track is confirmed, but no detections in the current time step were associated with the track. The track is then coasting, meaning the state of tracking is predicted using a filter, such as the Kalman Filter. While coasting

is an effective mitigation strategy for dropped detections in a 10 Hz system, the more a track is coasted, the more inaccurate the state becomes. This is where a deletion threshold is used to remove the track from the track list if the track is not seen for $X \; out \; of \; Y$ detection frames.

2.3 Deep Learning in Autonomous Driving

Deep Learning (DL) has accelerated the growth of technology in the autonomous driving field. DL is used for object detection in computer vision (CV) tasks, data forecasting natural language processing, and path planning in robotics. A popular neural network in autonomous driving is the convolutional neural network (CNN). The CNN is used for computer vision tasks such as lane segmentation [40] and object classification [29]. The recurrent neural network (RNN) handles temporal data such as language processing and state estimation. Over time, a RNN builds weights in its layers, allowing it to "remember" and handle data over time [35]. A vanilla RNN is limited to short sequences of data due to the exploding gradient problem. Over time, a weight in a RNN can exponentially grow, where the weights in the network are so large that the data moving through the network is not affected by the weights [17].

2.4 Long Short-Term Memory Networks

Long short-term memory (LSTM) networks are a type of recurrent neural networks (RNN). RNNs are classified as a feedback network (as opposed to a feed forward network), and can handle time-sequences of data. RNNs use temporal data and have the ability to handle sequences of data and perform tasks such as speech recognition, language processing, and data forecasting. In problems with long sequences of data, RNNs are not well suited due to the exploding gradient

problem during network training. To mitigate this issue, LSTMs were introduced to handle long data sequences [18].

In literature, the LSTM has three gates: an input gate, an output gate, and a forget gate. This is known as the three-gate version of the LSTM. A vanilla LSTM only uses the input and output gate. The following discussion and experiments in this paper refer to the three-gate version of the LSTM [16], which introduced the third gate, known as the forget gate. The input gate, y^{in} activation functions to regulate the information flowing in and out of each LSTM cell. The forget gate, y^{out} , is used to reset cell information when it becomes outdated or stale. For each gate, the variable w is used to represent the weight. The sigmoid function for the input and output gates is seen in equations (2.28), (2.29) and (2.30). The data flowing through the cell is net_c.

$$y^{in}(t) = f_{in}\left(\sum_m w_{in} y^m(t-1)\right) \tag{2.28}$$

$$y^{out}(t) = f_{out}\left(\sum_m w_{in} y^m(t-1)\right) \tag{2.29}$$

$$y^{\varphi}(t) = f_{\varphi}\left(\sum_m w_{\varphi} y^m(t-1)\right) \tag{2.30}$$

A cell state, s_c from eq. (2.31), is used to learn and forget information as it passes through the cell.

$$s_c(t) = y^{\varphi}(t) s_c(t-1) + y^{in}(t) g(net_c(t)) \tag{2.31}$$

The LSTM is a modern approach to apply deep learning to temporal based tasks. The LSTM eliminates the vanishing gradient problem from vanilla RNNs to handle longer sequences. With this, the LSTM comes with several issues: it cannot handle sequences in excess of 10,000 and it is more computationally expensive by a factor of nine [16].

2.5 Model Predictive Control for ACC

The MPC has risen in popularity over the last decade due to an increase in computation speed and algorithm maturity. An MPC can optimize over multiple variables with a set of given constraints. Initially, the MPC was used to regulate multi-variable chemical processes, but has been recently adopted in the automotive space [19] due to the MPC's ability to optimize around a set of constraints in multivariable problems. Examples of automotive control applications that use the MPC include energy management strategies [31], transmission [7], steering [14], and ACC [2] [36].

At the core of the MPC, the output of an MPC is optimized around a cost function as seen in eq. (2.32) [15]. The cost function considers the vehicle dynamics and any constraints set on the MPC. In the MPC formulation, t describes the discrete time index. Eqs. (2.33) and (2.34) define the system model. The vectors x, y, and u define the system state, input, and output. The set of constraints on the output of each vector is seen in eqs. (2.35), (2.36), and (2.37). The variable N defines the control horizon and how many steps the MPC will take to optimize. The MPC's model is initialized using eq. (2.38), where in this case the initialization is zero. Finally, eq. (2.39) defines the number of prediction steps that occur before the terminal control law is applied where N is the prediction horizon iteration.

$$\min_{U(t)} F(x(N|t)) + \sum_{k=0}^{N=1} L(x(k|t), y(k|t), u(k|t)) \tag{2.32}$$

$$s.t.\ x(k+1|t) = f(x(k|t), u(k|t)) \tag{2.33}$$

$$y(k|t) = h(x(k|t), u(k|t)) \tag{2.34}$$

$$x_{min} \leq x(k|t) \leq x_{max}, k = 1, ..., N_c \tag{2.35}$$

$$y_{min} \leq y(k|t) \leq y_{max}, k = 0, ..., N_c \tag{2.36}$$

$$u_{min} \leq u(k|t) \leq u_{max}, k = 0, ..., N_{cu} \tag{2.37}$$

$$x(0|t) = x(t) \tag{2.38}$$

$$u(k|t) = k(x(k|t)), k = N_u, ..., N-1 \tag{2.39}$$

When in operation, the MPC begins its control with an initialization of the of the system state vector, x. By using a system model, the MPC makes a prediction into the future to a desired prediction horizon variable. The optimization function, equation (2.32), considers the system constraints and cost function when determining a control output. The control output is fed back to

the system model and to the plant [15]. The cycle is repeated for the next cycle using the feedback from the plant and the new input vector.

In the ACC system, a list of inputs, constraints and outputs must be defined for an MPC to operate correctly [36]. The MPC input vector, p_t, uses the in-path vehicle's relative longitudinal position and velocity as seen in eq. (2.40). In order to maintain a safe distance at varying velocities, the safe distance, d_s, changes with vehicle velocity as seen in eq. (2.41). Eq. (2.41), also known as the car following model, ensures that there is enough distance between the lead vehicle and the ego vehicle as the velocity increases. When the ego vehicle comes a stop, the minimum safe distance, d_0 is enforced. A time gap constant, T_h, is used to determine how large the safe distance will be.

$$x_t = \begin{bmatrix} p_{relative\ distance} \\ p_{relative\ velocity} \end{bmatrix} \tag{2.40}$$

$$d_s(t) = T_h v_e(t) + d_0 \tag{2.41}$$

Constraints for the MPC directly affect driver safety, fuel economy, and ride quality. The first constraint on the MPC is the safe distance constraint in equation eq. (2.42), enforcing that the MPC shall not violate the safe distance constraint, d_0. The second set of constraints affect ride quality and fuel economy by enforcing limitations on the minimum and maximum acceleration the MPC can request. The constraints on the acceleration are seen in eqs (2.43), (2.44), and (2.45). Finally a constraint on the change in acceleration (jerk) is set on the output of the MPC in eq. (2.46). Limiting the rate of change in acceleration will provide a smoother ride and increase fuel economy by allowing the vehicle powertrain to operate efficiently.

$$d_0 < d_r \tag{2.42}$$

$$a_{min} < a_e < a_{max} \tag{2.43}$$

$$a_{min} < 0 \tag{2.44}$$

$$a_{max} > 0 \tag{2.45}$$

$$j_{min} < j < j_{max} \tag{2.46}$$

2.6 Vehicle-to-Everything Communication

V2X communication brings a wide spectrum of applications to semi- and fully-autonomous vehicles. Traffic light assist [8] [22], a vehicle-to-intersection (V2I) technology, is an autonomous system that alerts the driver of an upcoming intersection and can autonomously slow the vehicle to stop for a red traffic light. Platooning [4] is another technology that utilizes vehicle-to-vehicle (V2V) messages to give string stability to a line of vehicles. With V2V messages, each vehicle receives a full and accurate dataset of all vehicles in the platoon.

V2X communication is typically broken down into two categories: dedicated short-range communicate (DSRC) and cellular V2X (C-V2X). DSRC uses the IEEE 802.11p wireless protocol for local transmission and reception of basic safety messages (BSMs), signal timing and phasing (STaP) messages, and map messages. In ideal environments, DSRC radios can communication

with vehicles in ranges of 3 - 5 km [20]. DSRC are used for low-latency applications such as inter-vehicle communication. In areas without 5G cellular coverage, vehicles with DSRC radios can create an ad-hoc network [6]. C-V2X leverages 5G network technologies and is standardized in 3rd Generation Partnership Project (3GPP) release 15. Green navigation [13] uses C-V2X to receive traffic information to optimize navigation throughout a city, improving fuel efficiency and avoiding highly congested areas. Optimal vehicle communication uses a hybrid V2X architecture. The hybrid V2X architecture uses both DSRC and C-V2X radios to achieve the benefits of both systems [6].

Fifteen DSRC messages are defined in SAE J2735 [21]. This message set includes messages defining V2v, V2I, and vehicle-to-pedestrian communication. In a V2V application, BSMs are transmitted and received by all vehicle that have a DSRC radio and are within working range of the radio. A BSM includes rich vehicle data such as vehicle dynamic data, blinker status, and GPS location. In a V2I application, Signal Phase and Timing (SPaT) and Map messages are transmitted in pairs to vehicles in range of the intersection. SPaT messages define the timing information about the traffic lights in the intersection. Map messages provide detailed information about geographical layout of the intersection. Map messages were designed to work with both BSMs and SPaT messages. BSMs include GPS information that can be plotted onto the map message, giving absolute location of each vehicle at an intersection. SPaT messages are associated with map messages, giving the vehicles on-board-unit (OBU) enough information to determine which intersections are relevant.

2.7 Mississippi State EcoCAR

The EcoCAR Mobility Challenge is part of the Advanced Vehicle Technology Competitions (AVTC) Series where eleven universities across the United States and Canadar are tasked with converting a stock 2019 Chevrolet Blazer into a SAE level two autonomous hybrid vehicle. The EcoCAR Mobility Challenge is headlined sponsered by General Motors, MathWorks, and the Department of Energy and managed by Argonne National Laboratory.

The EcoCAR Mobility Challenge is in the fourth and final year of the competition, where teams are focused on refining and finishing their SAE level two autonomous features and hybrid electric drive train. The research in this thesis was performed using the 2019 Chevrolet Blazer from the EcoCAR Mobility Challenge for the Mississippi State EcoCAR team. The team's forward perception sensors were donated by Bosch and Intel, both of whom are EcoCAR sponsors.

CHAPTER III

LONG SHORT-TERM MEMORY NETWORKS FOR AUTOMOTIVE STATE ESTIMATION

3.1 Introduction

Adaptive Cruise Control (ACC) is an important part of automotive autonomy. To have a robust Adaptive Cruise Control system, a well-defined sensor fusion system is required. This chapter focuses on a feature-level sensor fusion system where three fusion techniques are explored: independent Kalman filtering, interacting multiple model (IMM) filtering, and a novel hybrid technique using a combination of both an IMM Filter and deep neural network (DNN). Kalman Filtering is a popular approach for automotive sensor fusion, but heavily relies on the given kinematic models to make state estimations. This chapter explores the IMM filter to capture multi-model motion as well as a DNN approach to build a data-driven model of the system. The IMM filter is built with several Kalman Filters featuring different kinematic models. The proposed DNN used is a Long Short-Term Memory (LSTM) network trained on radar and camera data to forecast a vehicle's state. To overcome the weaknesses in both IMM filtering and LSTM networks, this chapter propose's a hybrid technique consisting of both the IMM filter and the LSTM network. The proposed hybrid system uses a single filter for each sensor. The filtered sensor data is synchronized and used as the input for a trained LSTM network. The LSTM network was trained on over 100 simulated highway driving scenarios that might occur in an ACC application. Simulations and code were developed using MATLAB and the Autonomous Driving Toolbox. Our hybrid

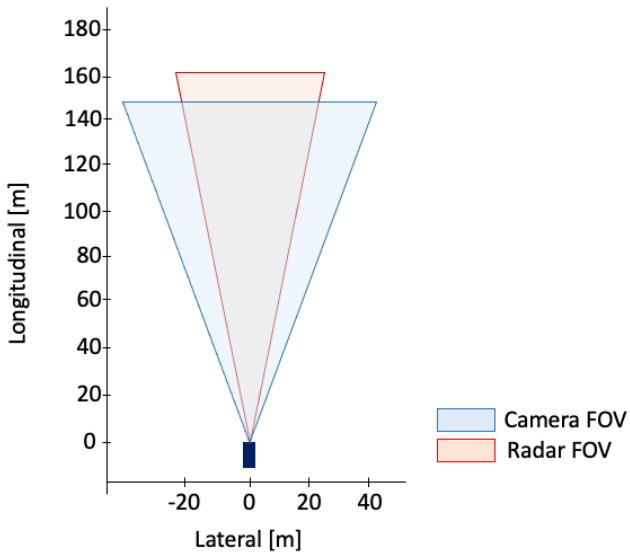


Figure 3.1

Simulated Sensor Field of View.

IMM-LSTM system outperformed the independent Kalman Filtering approach in longitudinal and lateral tracking accuracy (RMSE) by 23 percent and had promising results compared to the IMM filter where the tracking error was decreased by nearly 50 percent in certain driving scenarios.

3.2 Methods

3.2.1 Simulated Sensors

Data to train/test the LSTM network was generated using MathWork's Driving Scenario Designer [25]. To build a forward perception, camera and radar sensors were placed on a simulated vehicle. Sensor characteristics such as detection range, detection noise, and field of view were all tuned in the scenario designer tool. As seen in Figure 3.1, the camera and radar's field of view is plotted.

3.2.2 Sensor Fusion Using IMM Filter

The first approach proposed is an IMM state estimator configured with two EKFs: a NCV EKF and a NCA EKF. As seen in eqs. (2.5) and (2.6), the NCV and NCA motion models were chosen to represent the constant velocity maneuvers and constant acceleration maneuvers a tracked vehicle might make.

The IMM was initialized with a set of a filters, m in eq. (3.1), transition probabilities, mu, and state covariance matrix from the initialization measurement. The transition probabilities matrix limits how much a certain model can influence the system.

$$mu = [0.95, 0.95] \tag{3.1}$$

As seen in eq. (3.1), the transition probabilities are both 95 percent. In real-world scenarios, a specific model will never be 100 percent true. Therefore, one model will never represent the vehicle's state perfectly. The 5 percent margin allows for other models to influence the primary model.

Inputs to the IMM are timed aligned radar and camera detections. The detection alignment component produces a frame every 100 ms. To handle multiple tracks, a new IMM is created to handle state estimation for each track. As tracks are initialized and deleted, IMMs are created and deleted respectively.

3.2.3 Sensor Fusion Using LSTM Neural Network

In comparison the IMM system, a data-driven approach using an LSTM neural network was explored. The full LSTM-based sensor fusion architecture is seen in Figure 3.2. An LSTM was

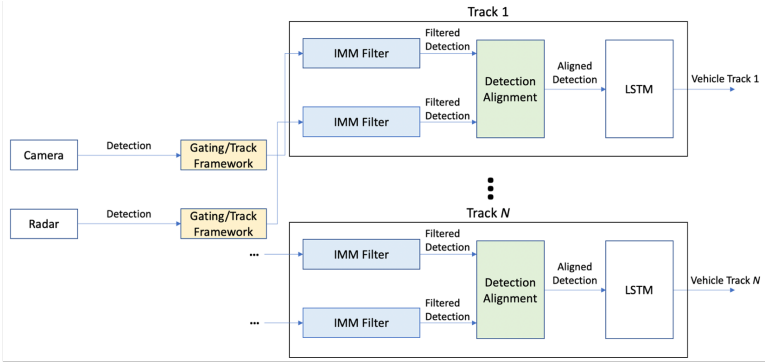


Figure 3.2

LSTM Sensor Fusion System

trained to perform state estimation using several input features from both the camera and radar sensors.

There are several advantages of using a DNN approach over a traditional model-based algorithm such as the IMM. DNNs have to ability to learn underlying patterns in data, where model-based algorithms are limited to the scope of their tuned model. In the first sensor fusion approach, the IMM has optimal performance in scenarios with nearly constant acceleration or nearly constant velocity. In scenarios where vehicles perform maneuvers outside of motion model scope, overall state estimation performance decreases. On the other hand, the LSTM network learns the motion models and underlying patterns of the input features. Furthermore, the LSTM network has a model agnostic capability to predict a wider variety of movements compared to the traditional IMM approach[33]. The immediate downside to the LSTM network is that the network is only as good as its training data. This means that if the training data does not include a wide variety of scenarios, the LSTM network will be limited with its capability to perform state estimation.

In the LSTM system architecture, the input features of the LSTM include filtered data from a camera and radar. The purpose of filtering preceding measurements is to remove measurement noise and consistently deliver input to the LSTM. Sensors are susceptible to dropped detections. To mitigate the issue of dropped detections, the preceding filters handle dropped detections by coasting state outputs. This ensures that the LSTM network is getting a measurement from both the camera and radar at each timestep.

Table 3.1

LSTM Input Features

Input Features	**Unit**
$Camera\ x_k$	m
$Camera\ vx_k$	m/s
$Camera\ y_k$	m
$Camera\ vy_k$	m
$Camera\ Variance\ x_k$	
$Camera\ Variance\ vx_k$	
$Camera\ Variance\ y_k$	
$Camera\ Variance\ vy_k$	
$Radar\ x_k$	m
$Radar\ vx_k$	m/s
$Radar\ y_k$	m
$Radar\ vy_k.$	m/s
$Radar\ Variance\ x_k$	
$Radar\ Variance\ vx_k$	
$Radar\ Variance\ y_k$	
$Radar\ Variance\ vy_k$	
$Ego\ v_k$	m/s
$Ego\ \psi_k$	degrees

Table 3.2

LSTM Output Features

Output Features	Unit
x_k	m
y_k	m

3.2.3.1 LSTM Network Training

The LSTM network was trained on 350 driving scenarios. The sequence lengths ranged from 36 – 412 detection sequences. The input and output features are seen in Table 3.1 and Table 3.2 respectively. The LSTM network regresses from radar and camera measurements to lateral (x) and longitudinal (y) in cartesian coordinates using sensor measurements, sensor variance vectors, and ego vehicle data. Most RNNs, including LSTMs, do operate using time as an input variable. It is important that all data in the sequences used to train an LSTM have the same discrete timestep. After an LSTM is trained using a specific timestep, running the network requires the data to enter the network at the specific timestap that the network was trained on. The trained LSTM in this architecture expects an input every 100 ms. In this chapter, the trained LSTM is limited to data frames at 100 ms intervals.

Input features to the LSTM network were normalized to [-1 1] to achieve optimal performance results. This method of normalization was chosen to handle negative sensor measurements. The normalization equation is shown in eq. (3.2). Output features were not normalized. The LSTM network regresses to the lateral and longitudinal coordinates.

$$X_{normalized} = \left(\left(\frac{x - x_{min}}{x_{max} - x_{min}}\right) * 2\right) - 1 \tag{3.2}$$

During the training cycle for the LSTM, one normalization tactic that was explored was to normalize both the input and output features. This method proved to train very well with validation data, but during the testing phase the network performed poorly. A second approach was found to normalize the input features and allow the network to regress to lateral and longitudinal coordinates.

3.2.3.2 LSTM Network Configuration

The LSTM network is composed of thirteen hidden layers as seen in Figure 3.3. Four fully connected layers were discovered to have optimal performance for the Euclidean coordinate regression. As discussed in [24], the input fully-connected layers allow the network to learn and handle the complex maneuvers of the tracked vehicle. The two LSTM layers address non-linearities and retain memory of the sequence data.

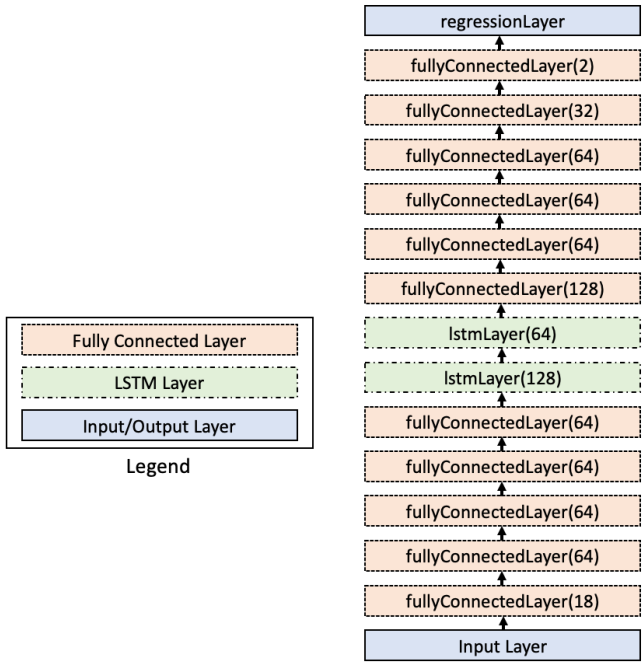


Figure 3.3

LSTM Network Layers

During training, the Adam Optimizer was chosen through a comparison process against other optimizers. Other optimizers such as Stochastic Gradient Descent with Momentum (SGDM) were explored but did not yield optimal results when compared to Adam. Table (3.3) shows the hyper parameters used to train the LSTM network in Figure (3.3).

3.2.4 Performance Metrics

Two metrics will be used to evaluate the performance of each state estimation approach. The mean absolute error (MAE) in eq. (3.3) is used to define the percentage error between the vehicle state estimate, y, and the ground truth, y' for the entire sequence, N. The root mean squared error

Table 3.3

LSTM Hyperparameters

Hyperparameter	Value
Learning Rate	0.0005
Epochs	4500
Mini-Batch Size	256

(RMSE) in (3.4) is used to observe the error in meters between the vehicle state estimate and the ground truth.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \mid y_i - y_i' \mid \tag{3.3}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - y_i')^2} \tag{3.4}$$

3.3 Results and Discussion

3.3.1 Simulated Sensor Performance

For the experiments in this chapter, a radar and camera sensor were modeled and simulated using MathWorks's Autonomous Driving Toolbox in MATLAB 2020b. Within the toolbox, sensors were tuned to represent real-world sensor measurement characteristics.

Individual sensor performance was first analyzed to observe the measurement performance in the longitudinal direction. Figures 3.4 and 3.5 show the sensor longitudinal measurement at ranges from 0 to 80 meters in a box plot format. The box plot format shows the sensor measurement at five different ranges up to 80 m. High measurement accuracy is a necessity for reliable sensor fusion

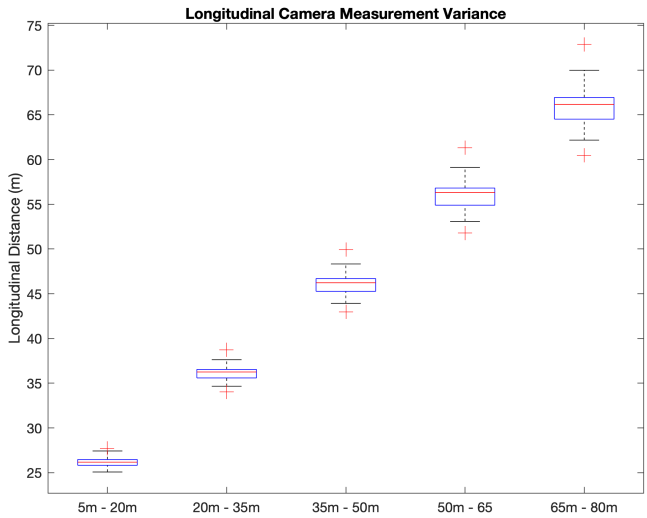


Figure 3.4

Camera Detection Boxplot

systems and accurate tracking results. It is also important to observe each sensor's measurement characteristics so that the sensor fusion state estimation algorithms can be tuned correctly.

The boxplot is a useful plotting tool to show sensor measurement performance over various ranges. The measurement mean is indicated with a red line in each box, measurements in the first and third quartile are above and below the red line within the blue box respectively, and minimum and maximum measurements are shown with the whiskers. Finally, measurement outliers are shown with a red "+".

As expected, the radar sensor outperformed the camera sensor in the range measurements. Cameras are limited by the number of pixels in each image. For a camera to make a range measurement, a relationship between pixel frames must be established [41]. As a vehicle gets

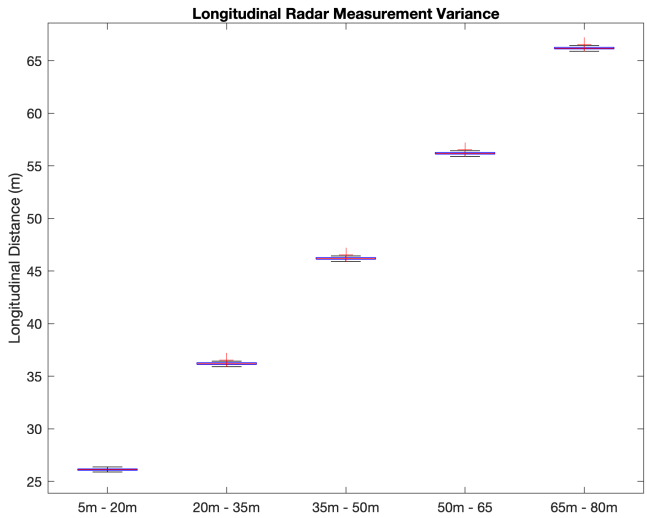


Figure 3.5

Radar Detection Boxplot

further away, the vehicle is represented with fewer pixels and the measurement become less accurate. Figure 3.4 shows the camera measurement deviation rise linearly as the distance increases.

Radar performance is not impacted as harshly by the increase in distance at ranges up to 80 m. Frequency modulated continuous-wave (FMCW) radars radiate rf energy to accurately measure target distance and velocity relative to an automotive camera sensor. With this, the mid-range radar's performance did not decrease linearly as seen in Figure 3.4.

3.3.2 IMM Sensor Fusion Results

Before testing the full sensor fusion system, individual components of the system were tested. The first component testing was the IMM state estimator.

To verify that the IMM was performing correctly, two different simulations were used to initially verify filter functionality. The first functional requirement of the IMM was the model switching

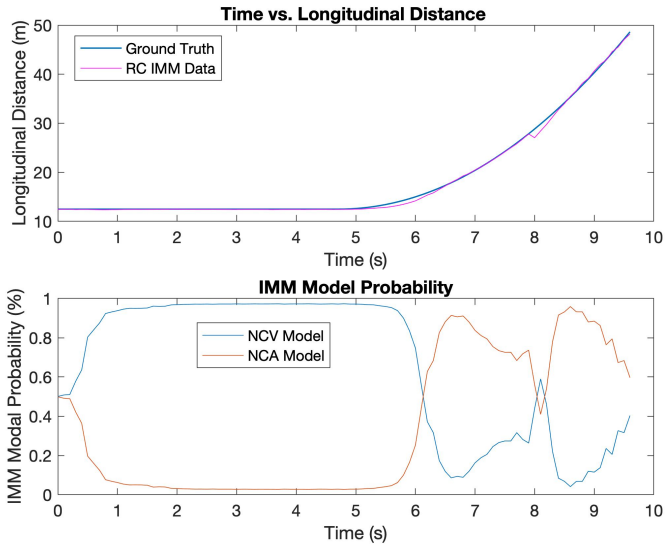


Figure 3.6

IMM Model Switching.

component. As a vehicle performs various maneuvers, the IMM must adjust its model probability to represent the vehicle's current state. Figure 3.6 shows a simple longitudinal maneuver test where a lead vehicle drives at a constant velocity for the first five seconds of the test, then accelerates from time stamps five to ten. This test verified that the models were appropriately switching as the tracked vehicle changes velocities.

In Figure 3.6, the IMM correctly switches between the NCV and NCA modes. In the simulated scenario, the lead vehicle changes motion models at time 5 seconds. At time 5 seconds, the second plot in Figure 7 shows the IMM switching model modes to correctly output from the NCA Kalman Filter.

The second functionality test for the IMM was to compare its output to a single camera system and a single radar system. Three systems were tested: a Radar IMM, a Camera IMM, and an

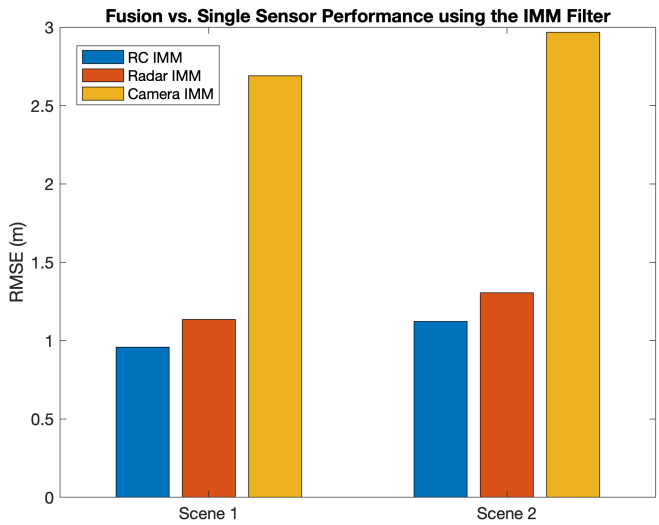


Figure 3.7

Highway RMSE(m) Error Resuts.

RC IMM. The scenarios simulated were two highway driving scenes. In both scenes, the lead vehicle is at a relative distance between 60 – 100 m. The vehicle's maneuvers include lane changes and velocity changes. The results for the simulations are seen in Figure 3.7 where the Euclidian RMSE/m error was used as a metric to analyze system performance. A higher RMSE indicates worse performance.

Figure 3.7 shows that the using both radar and camera sensor measurements, the radar-camera (RC) IMM is able to achieve more accurate results compared to a single sensor system.

Finally, the RC IMM system was tested across 100 driving scenarios. The purpose of this test was to expose the RC IMM system to a wide variety of tests. If an IMM did not perform well in a certain test, issues were addressed by making refinements to the IMM parameters and the testing procedure was restarted to collect measurements across all of the testing scenarios.

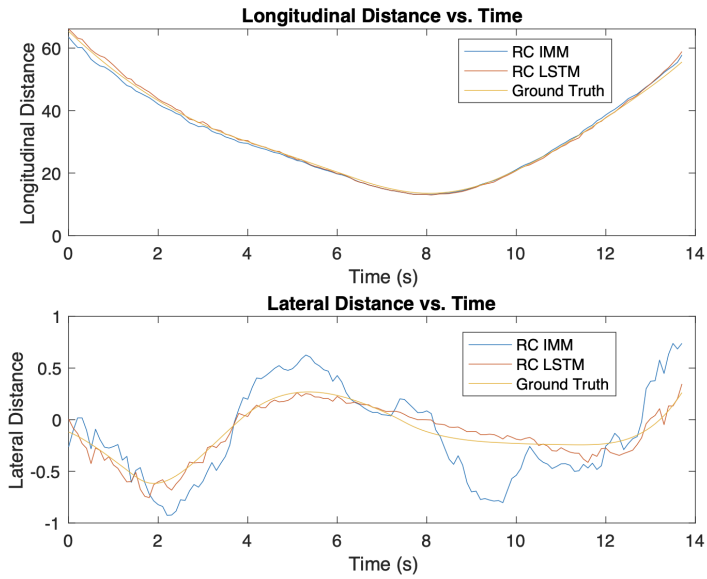


Figure 3.8

Scenario 2: LSTM vs. IMM Performance.

3.3.3 LSTM Sensor Fusion Results

After several iterations of training, tuning, and refinement, the LSTM network was able to achieve consistent sensor fusion results. The final training results were a validation error (RMSE/m) of 0.41 and a mini-batch error of (RMSE/m) 0.29.

Initial testing of the LSTM composed of simple scenes with constant velocity and constant acceleration maneuvers. Once the LSTM was verified to work on simple scenarios, more complex scenarios were introduced. Figure 3.8 shows a simple stop and go test. In this simple scenario, LSTM was able to make accurate predictions.

The LSTM was able to accurately track the lead vehicle in both the lateral and longitudinal direction. The LSTM was also able to outperform the RC IMM in the lateral tracking.

Table 3.4

Scenario Descriptions.

Scenario	**Description**
1	The lead vehicle remains at a constant velocity.
2	The lead vehicle remains stationary while the ego vehicle approaches from behind at 20 mph. The ego vehicle safely decelerates and stops at a safe distance of 5 m.
3	The lead vehicle stops at a stop sign, then accelerates. The ego vehicle stops at the stop sign, then accelerates behind the lead vehicle.
4	The lead vehicle travels in front of the ego vehicle on a a highway. The lead vehicle makes several maneuvers such as lane changes and velocity changes.

To further benchmark and test the LSTM sensor fusion system, four different scenarios were designed to test the LSTM's capabilities. The LSTM system was compared against five different filtering systems. Along with the Radar and Camera IMM from Figure 3.7, a Kalman Filter using a NCV model (CV KF) and an Extended Kalman Filter using a NCV model (CV EKF) were also used to benchmark system.

Table 3.4 describes each scenario and the maneuvers that were made by the lead vehicle. The performance results for each scenario are shown in Table 3.5. Performance results were measured in RMSE/m.

The results from Table 3.7 show that the IMM and the LSTM sensor fusion systems were able to achieve consistent results across all four scenarios. Both the EKF and KF systems excelled at Scenario 1, where the vehicle made maneuvers within the NCV model scope. As the scenarios included more maneuvers such as lateral changes and changes in velocity, the EKF and KF systems suffered a performance hit seen in Scenarios 3 and 4. The camera and radar IMM systems both

Table 3.5

State Estimation Results, Error reported in RMSE(m). R(Radar), CV(Constant Velocity), EKF(Extended Kalman Filter), IMM(Interacting Multiple Model), RC(Radar-Camera), KF(Kalman Filter), LSTM(Long Short-Term Memory)

Scenario	**R CV EKF**	**C CV EKF**	**R IMM**	**C IMM**	**RC CV KF**	**RC IMM**	**RC LSTM**
1	1.1426	3.9067	2.0534	1.8525	1.0518	1.3215	1.3406
2	1.0849	1.5821	0.4159	0.4782	2.2153	1.0812	0.5419
3	1.1858	2.3526	0.1625	0.5458	6.5802	0.8565	0.4486
4	2.4556	4.6987	1.5259	1.7126	15.0106	0.8492	2.1208
Average RMSE(m)	1.467225	3.135025	1.039425	1.147275	6.214475	1.0271	1.112975

had consistent performance, but when introduced to the more complex highway Scenario 4, both systems had degraded performance while the RC IMM was able to leverage measurements from both sensors.

The final testing scenario used to benchmark the LSTM was a scenario with over 1300 timesteps. This was test intended to exploit some of the weaknesses with the LSTM network. As stated in [16], the LSTM networks struggle to maintain prediction accuracy when input sequence lengths exceed 1000 steps. Figure 3.9 shows a comparison between the RC IMM and the RC LSTM systems.

The RC LSTM system struggles to maintain accurate measurements after timestep 600 in the longitudinal direction. In the lateral direction, the RC LSTM system maintains accurate state estimations throughout the duration of the entire sequence and outperforms the RC IMM system in the lateral state estimation. To mitigate this issue, a reset mechanism could be setup to reset the LSTM network when the network begins to drift away from radar-camera measurements. THe

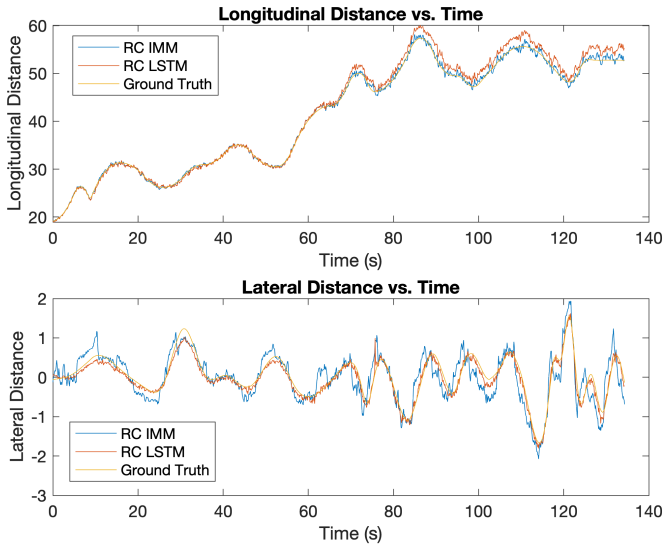


Figure 3.9

Long Sequence LSTM Results.

LSTM netowrk could also be reset regulary at a selected timestep preventing measurement drift after long sequences of input data.

Finally, another weakness in the RC LSTM system is the limited amount of training data. 350 unique scenes were generated to cover a large number of scenarios a vehicle perception system might encounter. Edge cases that were not including in the training date would cause the LSTM's performance to suffer.

3.4 Conclusions and Future Work

In conclusion, this chapter explored two state estimation techniques for sensor fusion system. The first system used a model-based approach using an IMM state estimator. The second system used a data-driven approach with an LSTM network trained on sensor measurement to regress to vehicle position coordinates. After successful training and benchmarking, the RC LSTM system

Table 3.6

Long Sequence Scenario: IMM vs. LSTM.

RC IMM		**RC LSTM**	
Long. RMSE(m)	**Lat. RMSE(m)**	**Long. RMSE(m)**	**Lat. RMSE(m)**
0.46625	0.3200	1.3260	0.1122

and RC IMM system, it can be concluded that both systems outperformed the single filter estimation approaches using a Kalman Filter of Extended Kalman Filter. Across multiple scenarios the RC LSTM outperformed single state estimation techniques such as the Kalman Filter and Extended Kalman Filter algorithms but failed to surpass the performance of the RC IMM system.

Looking into the future, progress on this work will explore novel approaches to incorporate DNNs into model- based algorithms. Mechanisms to predict model modes and corrections as seen in [39] [33] . Another area that will be researched is lane assignment. Using an LSTM to perform lane assignment based on road curvature, vehicle position, and lane boundaries might prove to be beneficial. A predicted lane assignment for each track would be beneficial for an autonomous vehicle's capability in identifying in-path vehicles for ADAS applications such as adaptive cruise control and lane keep assist.

CHAPTER IV

TESTING SENSOR FUSION IN XIL ENVIRONMENTS FOR ADAPTIVE CRUISE CONTROL

4.1 Introduction

This chapter will discuss the engineering process and testing platforms used to bring a radar-camera perception system from initial requirements to full functionality in an ACC system. The sensor fusion is deployed into 2019 Chevrolet Blazer for on-road testing. The testing platforms used to develop and test the sensor fusion system include: model in the loop, hardware in the loop, and vehicle in the loop. An engineering V-Diagram was used to guide the project's development and ensure that requirements are met in four different testing environments: model in the loop(MIL), software in the loop(SIL), hardware in the loop (HIL), and vehicle in the loop(VIL).

4.1.1 Connected and Autonomous Vehicle System Architecture

In the Chevrolet Blazer, two hardware controllers are employed for processing sensor detections and controlling the hybrid vehicle propulsion. An Intel TANK-870Q170 is used to process sensor detections, command longitudinal acceleration, and manage incoming V2X messages. A dSPACE MicroAutoBox(MABx) serves as the hybrid vehicle supervisory controller. This section will focus on the software and hardware layout of the Intel TANK. With heterogeneous sensors, it is required

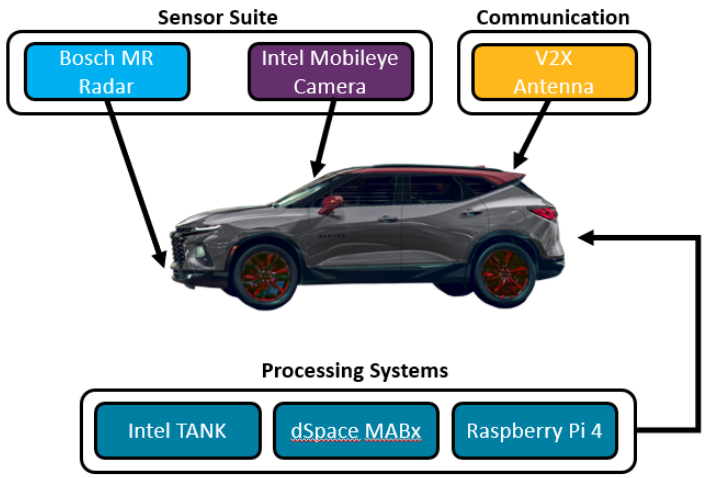


Figure 4.1

Chevrolet Blazer Sensor Suite.

to build a robust sensor fusion algorithm to handle heterogeneous detections. One of the most popular sensor combinations, radar-camera, is employed in this CAV system as seen in Figure 4.1.

4.1.1.1 Sensor Architecture

The Chevrolet Blazer is outfitted two with forward facing sensors: a Bosch MRR Evo 14 Front Radar and an Intel Mobileye 6. Both sensors are feature level sensors, meaning that feature level data is reported from each sensor. Table 4.1 and 4.2 show the Bosch radar and Intel camera sensor specifications. The sensor pair was chosen due to its complementary measurement system. As discussed early, radar and camera sensors are a economical choice for an ACC system. When selecting sensors for an ACC system a critical region should be drawn to define the area where sensor performance is critical. In the critical region the sensor suite should be able to provide: accurate measurements, redundant measurements, lane classification, and multi-object detection.

Table 4.1

Bosch MRR Evo 14 Front Radar.

Specification	**Value**
FOV	90 degrees
Tested Range	0 m - 80 m
Measurements	- Position (m) - Velocity (m/s) - Acceleration (m/s^2) - Deviation (m^2/s^2) - Lane Classification (Enum)

As seen in Figure 4.2, the camera and radar sensor suite covers the critical region, which is objects in the left, right and ego lanes.

4.1.1.2 Sensor Fusion Software Architecture

The software architecture on the Intel TANK was developed using ROS, Python, and MATLAB. As seen in Figure 4.3, the core sensor fusion architecture consists of three layers: sensing, preprocessing, and tracking. The architecture is organized into modular nodes using ROS and Simulink. The architecture is distributed into local processing for each sensor. A global tracker is employed to fuse the detections from the radar and camera trackers.

In the sensing layer, the software is designed to process the CAN messages from each sensor. CAN messages are decoded into usable data formats and published in a ROS network. A preprocessing layer is used to synchronize detections from the asynchronous sensors. The preprocessing layer creates a detection frame every 100 ms. During the 100 ms window, if a detection is received

Table 4.2

Intel Mobileye 6 Camera.

Specification	**Value**
FOV	38 degrees
Tested Range	0 m - 178 m
Measurements	- Position (m) - Velocity (m/s) - Acceleration (m/s^2) - Lane Polynomial - Lane Classification - Object Classification

from a sensor, the packet buffer will store the detection in the frame. At the end of the 100 ms window, the frame is published to its respective local tracker.

In the tracking layer, each sensor has an associated local tracker. A local tracker at each sensor provides several benefits: smoothing out sensor measurements, per-sensor track maintenance rules, and per-sensor data association tuning. Radar and camera sensors have different strategies for generating detections. By employing a local track for each sensor, per-sensor tuning strategies can be employed to create uniform detections before the detections are fused in the global tracker. Once detections are filtered in the local trackers, the global tracker is designed to fuse heterogeneous detections. The output of the global tracker is a track list (with a maximum of three tracks) of surrounding objects such as vehicles, stationary objects, cyclists, and pedestrians. The maximum number of tracks is limited to three tracks due to the assumption that the ACC system shall function on a two lane highway with no more than three vehicles in view of the sensors.

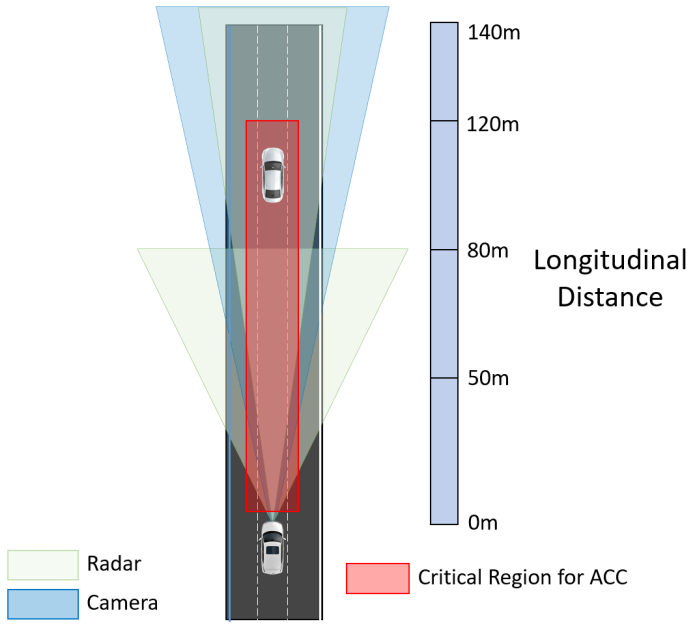


Figure 4.2

Critical Region for Defining ACC Sensor Functionality.

4.2 Engineering Development and Testing Processes

Interfacing with a variety of controllers and developing large and complex software is not a trivial task. To address this set of issues, a development process following the engineering V-Diagram Model, as seen in Figure 4.4 below, was adopted. The purpose of the V-Diagram Model is to provide a systematic approach for requirements development, design, software development, and testing. As opposed to a traditional V-Diagram Model, the V-Diagram Model in Figure 4.4 was modified to support four different testing environments: MIL, SIL, HIL, and VIL. The V-Diagram approach provides a methodology for moving requirements and software through a variety of testing environments and eventually into the vehicle testing environment where the full system is validated. This allows for safe and effective testing in the final platform, VIL. A summary of

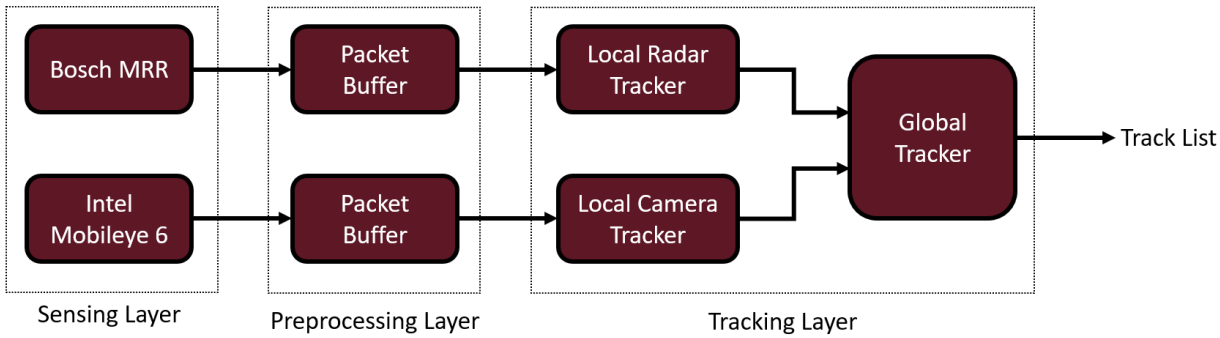


Figure 4.3

Sensor Fusion Architecture.

criteria tested in each environment is seen in Table 4.3. In the following sections, each testing environment will be broken down further into rational and application.

The V-Diagram Model is driven by requirements developed early in the development phase. With ACC as the end application, seven requirements were developed to describe to necessary functionality of the sensor fusion system for ACC to function efficiently and safely. The seven sensor fusion requirements are listed in Table 4.4 below.

4.2.1 Model-in-the-Loop Testing

The sensor fusion software in Figure 4.3 was developed using a model-based design approach. MathWorks' Simulink software was used to develop, test, and build the software on the Intel TANK in the Chevrolet Blazer. In the MIL environment, the tracking algorithm components such as state estimation, track maintenance, data association, and detection transformation were developed. The MIL environment relied on two techniques for testing: scenario simulation and data-replay. Scenario simulation used MathWorks' Driving Scenario Designer[25] to build scenarios. Sensors placed on the virtual vehicle would detect objects in the scenarios. The collected sensor data was

Table 4.3

Environment Criteria.

Environment	Criteria
MIL	This environment deals with fundamental algorithm functionality. Mechanisms such as detection generation and data-replay are used to test algorithms in this environment.
SIL	This environment ensures that generated code (C, C++) functions as intended in the ROS network.
HIL	This environment deals with the communication between controllers.
VIL	This environment tests full system functionality after all safety requirements have been met in previous environments.

used to replay through the sensor fusion algorithm both in real-time and simulation time. Real-time data functionality refers to the built code deployed and running on the autonomous controller. Simulation time is the timestep set in the simulation environment. One advantage of scenario simulation is the ability to design scenarios that would be difficult or dangerous to replicate in the real world. An example of this would be a lead vehicle suddenly coming to a stop. Relative distance and velocity is a key measurement that the sensor fusion system must be able to handle. If the system has too much "lag" or does not track the vehicle correctly, it could cause a major incident.

Data-replay was heavily used in development of the sensor fusion algorithm. An advantage of using data-replay over scenario simulation was the ability to tune the sensor fusion algorithms to real sensor data collected in the VIL environment. In the MIL environment, an example of the workflow used is seen in Figure 4.5. Simulink is used to simulate the sensor fusion model. Logging data in the ROS networks requires the rosbag functionality, which is the logging system in ROS. rosbags can be recording during the data-collection period and replayed into a ROS network. In

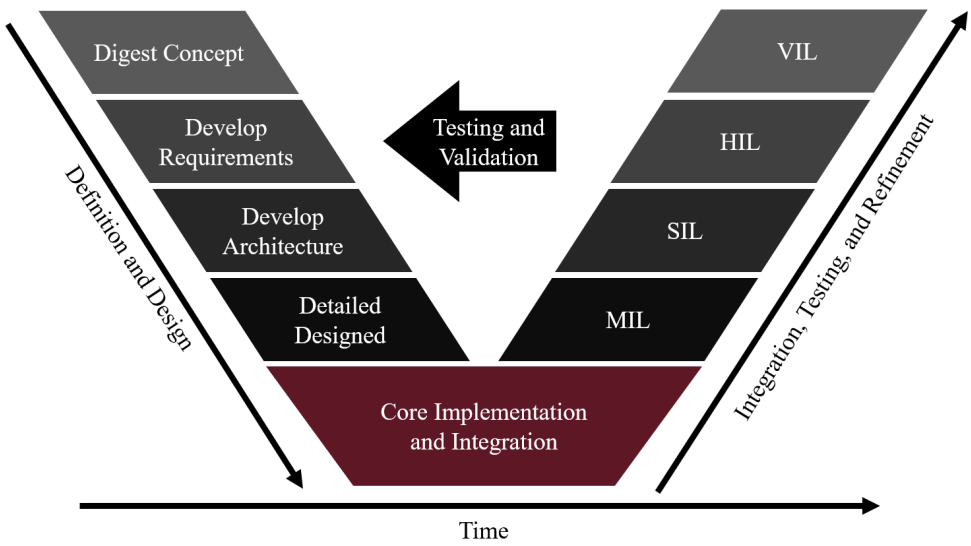


Figure 4.4

Engineering V-Diagram Model.

Simulink, the rosbag replay block is used to replay sensor data into the sensor algorithm. Sensor data is fed into the sensor fusion algorithms similar to how the data enters the sensor fusion system in the VIL environment.

Ground truth data is collected independently on an OxTS RT3000 unit. Since data on the OxTS unit and the ROS system were captured independently, the two datasets needed to be time aligned before further review. To time align ground truth and sensor fusion data, vehicle speed captured on the OxTS unit and vehicle speed capture in the ROS system was cross correlated. The cross correlation process shifts the vehicle speed signals across time and provides a correlation at each time shift. The time shift index with the highest correlation value is used to time align the two datasets. Vehicle speed is used because it an accurate measurement recorded on the OxTS(captured with GPS and IMU sensors) and directly in the vehicle. Once data is time aligned,

Table 4.4

Sensor Fusion Requirements

ID	Requirement Description
1	The sensor fusion system shall be able to track vehicles in longitudinal ranges of 1 m to 120 m
2	The sensor fusion system shall correctly assign lanes to tracks in longitudinal ranges 1 m to 80 m
3	The perception system shall prioritize the closest in-path vehicle for the ACC algorithm.
4	The perception system shall confirm a track after 5 out 6 detections have been assigned to the tentative track.
5	The perception system shall delete a track after 10 out of 10 frames of not having a detection assigned to the track.
6	In ranges of 5 to 80 meters, the perception system shall not switch track IDs.
7	The MAE of a track shall be less than 5% for in-path vehicles within a 5 to 80 meters infront of the ego vehicle.

the sensor fusion system results can be simulated and compared to ground truth data. Cross correlated data is stored as a scenario folder with two datasets: ground truth and sensor data. This workflow allows for rapid development and accurate results (MIL vs. VIL). This workflow also allows for direct sensor comparison (radar vs. camera). In the Simulink model, the camera sensor data input can be disabled, providing a radar only system. The same process can be for a used to simulate camera only system. This approach provides a comparison between camera, radar, and radar-camera systems using the same scenario and dataset.

4.2.2 Software-in-the-Loop Testing

The SIL environment is used to test the full ROS system in a virtualized environment. In the SIL environment, the system is running in real-time, as opposed to the MIL environment.

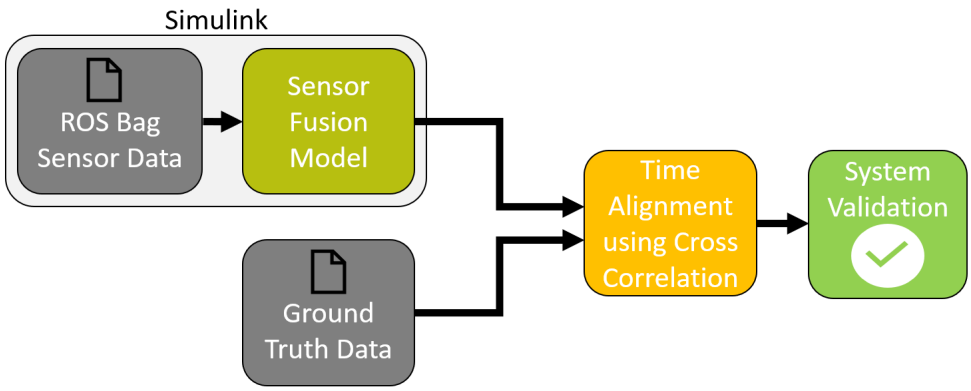


Figure 4.5

MIL Validation Workflow.

Requirements such as compatibility between software components, code generation, and ROS environment variables are all addressed in the SIL environment. Testing in the SIL environment uses data-replay in the form of rosbag files. As discussed in the MIL environment, rosbags replay data in real-time, therefore any timing requirements are addressed in the SIL environment.

Along with the core sensor fusion software and ROS network, other applications such as real-time plotting tools and diagnostics are also developed in the SIL environment. Diagnostic requirements can be tested using fault injection in the rosbag files. In Figure 4.6, a diagram of the SIL environment is seen with build sensor fusion code outputting tracks to a Birds-Eye-Plot tool built in Python.

4.2.3 Hardware-in-the-Loop Testing

The HIL environment is primarily used for testing communication interfaces between controllers in the vehicle. Five communication channels are connected to the Intel TANK and requirements for each communication must be met for safe functionality when the system is deployed into

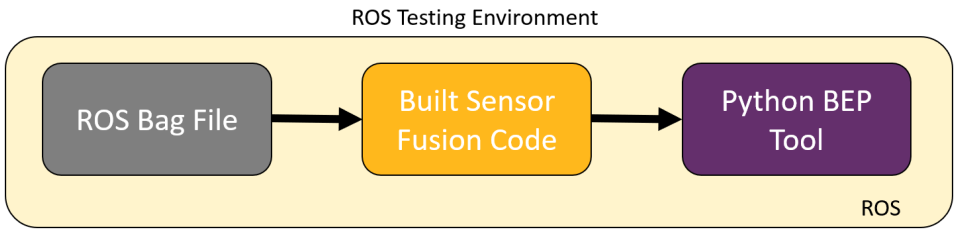


Figure 4.6

SIL Testing Environment Example.

the vehicle. To run HIL simulations, a dSPACE HIL simulator is used to imitate the physical CAN interfaces in the Chevrolet Blazer. As seen in Figure 4.7, three CAN channels from the Intel TANK are used to communicate with sensors/controllers in the vehicle. On the HIL simulator, soft ECUs are used to emulate the radar and camera sensors. The dSPACE MABx controller interfaces with the HIL simulator through multiple CAN channels. In a test case where adaptive cruise control functionality such enabling/disabling is being tested, the HIL simulator would emulate the button status. In this testing scenario, the HIL simulator was used to test the signal chain between the Intel TANK and the vehicle. Requirements that could potentially be dangerous to test in the VIL environment are possible using the HIL simulator and can be performed conveniently in a lab setting.

For analyzing the sensor fusion performance, an OxTS RT3000 GNSS/INS was used to obtain ground truth measurements for a fixed object. To meet the requirements listed in Table 4.4, three scenarios were designed to address each requirement and test the systems functionality. As well as analyzing performance in real-time, ground truth and sensor fusion measurements were logged for

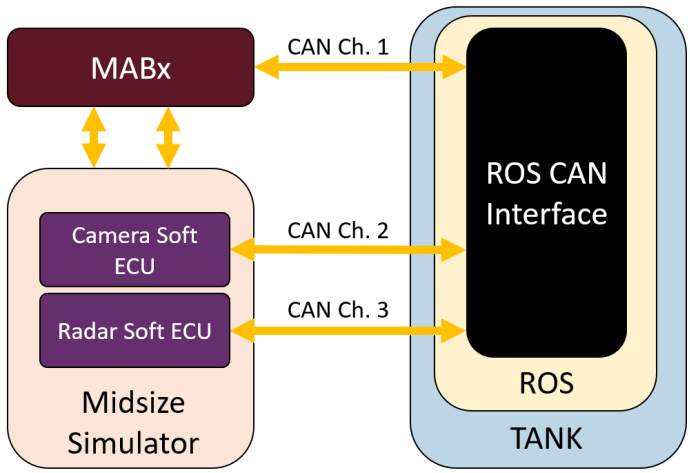


Figure 4.7

HIL Testing Environment Example 1.

data-replay. As seen in Figure 4.8, the system functionality on the TANK is tested using a rosbag file to replay data into the system.

4.2.4 Vehicle-in-the-Loop Testing

In the VIL environment full system functionality is tested. High level system requirements, such as the requirements listed in Table 4.4, are tested in the VIL environment. To ensure optimal use of vehicle time, all controller outputs were logged via a CAN or rosbag logger. To meet the requirements described in Table 4.4, three scenarios were designed to test the functionality of the sensor fusion system and are listed in Table 4.5.

Real-time sensor fusion CAN data is recorded on an OxTS RT3000 unit, which captures ground truth and measured detections. To validate the sensor fusion system, metrics such as mean average error (MAE), RMSE, and number of track drops were all used. Figure 4.9 shows the VIL testing

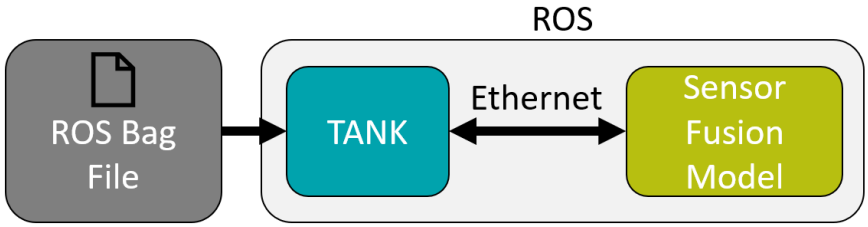


Figure 4.8

HIL Testing Environment Example 2.

environment dataflow in a scenario. A testing limitation was the lack of multiple OxTS units. With only a single OxTS unit, testing was limited to a scenario with only a single static test vehicle.

Each testing scenario was tested at two approach speed: 15 mph and 25 mph. Testing scenarios are also broken down into in-lane and out-of-lane scenarios. The tracking of the lead vehicle's lateral position directly affects the lane assignment. Technology Blvd., seen in Figure 4.10, located in Mississippi State University's research park, was used as a closed-course testing ground due to its 340 m straight away, high quality lanes, and ease of access. Requirements for a closed-course track are: a lead vehicle and a follow vehicle.

All scenarios from Table 4.5 were performed on-road, using real sensor data. Tests were repeated using two different vehicles: a White Ford F250 and a Black Toyota Camry. Testing results for the system are seen in Tables 4.6 and 4.7. Testing procedures were replicated across two vehicles to test if the system could obtain accurate results with different types of vehicles (trucks vs. sedans). In both testing scenarios, the weather was mostly sunny conditions.

In the direct approach scenarios, both vehicles were accurately tracked with less than a 5% longitudinal range error and less than a 2 mph longitudinal range-rate error at speeds of 15 and

Table 4.5

Sensor Fusion On Road Testing Scenarios.

Scenario ID	**Scenario Description**
1	Ego vehicle approaches an in-path vehicle at 10 mph from an initial distance of 250 m.
2	Ego vehicle approaches an in-path vehicle at 20 mph from an initial distance of 250 m.
3	The ego vehicle is positioned 5 m from the lead in-path vehicle. The ego vehicle reverses at 5 mph until the 150 m marker.

25 mph. Figure 4.11 shows a 30 mph direct approach scenario. A limitation of the sensor fusion system is state accuracy at higher relative velocities. As seen in Table 4.7, the average error in the velocity measurement increases as the relative velocity between the ego vehicle and the track vehicle increases.

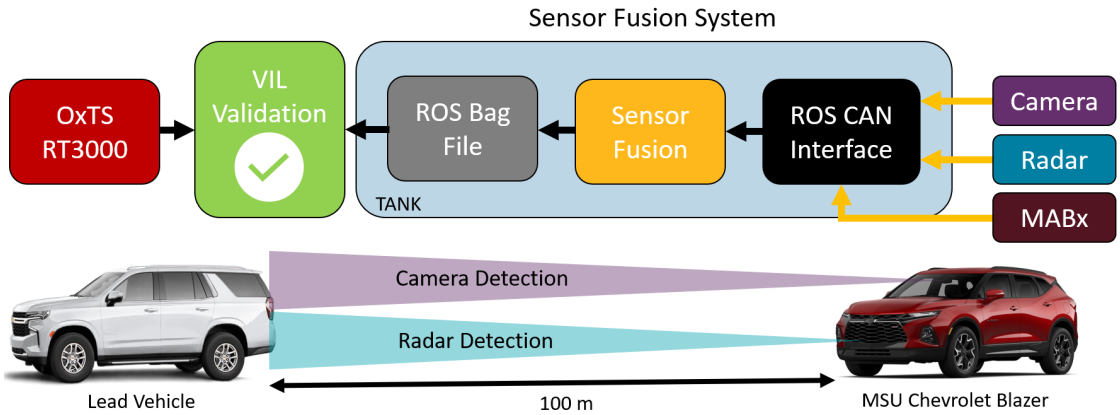


Figure 4.9

VIL Testing Environment Example.

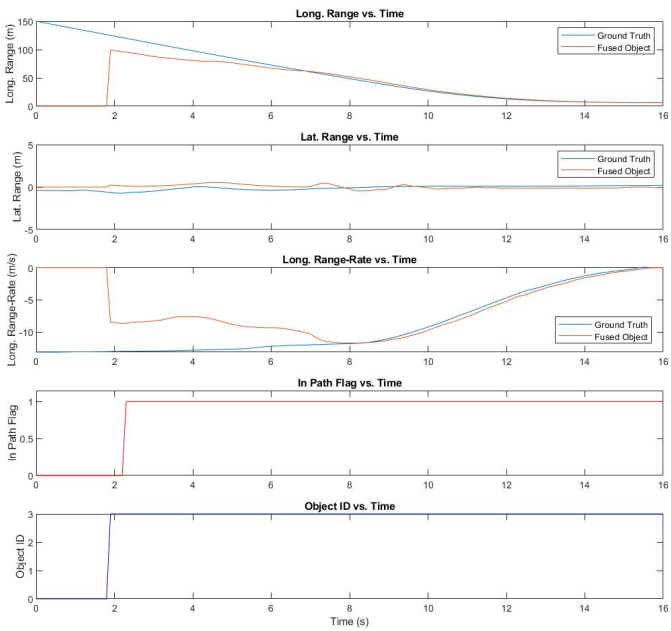


Figure 4.11

VIL Testing Environment Example.

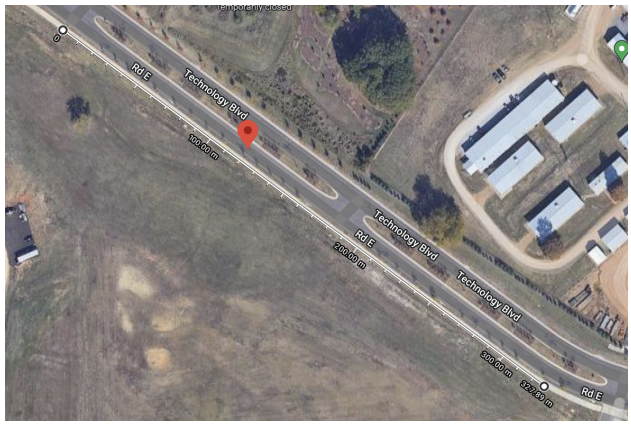


Figure 4.10

Technology Blvd. Sensor Fusion Testing Location.

Five different subplots are shown in Figure 4.11 to describe the tracking performance of the sensor fusion system. In the first and second subplot, the longitudinal and lateral range for both the ground truth and fused object are plotted against time. The in path flag is plotted to show that the lead test vehicle was identified to be in an in-path object at a longitudinal distance of 130 m. The fifth subplot shows the object's assigned ID. In a robust sensor fusion system, the same object should not swap IDs over time. This would mean that the system is capable of tracking the same object over a period of time without "dropping" the track. If the same tracked object is swapping IDs, the system could be suffering from intermediately "dropping" the track and is deleting tracks too quickly. This causes issues to other downstream applications such as ACC where the in path vehicle suddenly disappears and reappears to the system. This would result in unwanted acceleration and braking, causing the driver's ride to be uncomfortable and reducing fuel economy.

Table 4.6

Sensor Fusion Testing Result with Black Toyota Camry.

Scenario	**Average Range Error %**	**Average Range-Rate Error (mph)**	**Long. Range RMSE (m)**	**Lat. Range RMSE (m)**
1	2.54	1.3	1.38	3.09
2	3.82	1.5	1.62	2.91
3	3.66	0.36	0.94	2.91

While static object tests with the OxTS unit provide insight to sensor accuracy, dynamic object tests add more variables such as road curvature, vehicles passing in adjacent lanes, and vehicle kinematics that the sensor fusion system must account for. The dynamic scenario was designed to mimic a highway scenario that might occur when the ACC system is activated. The scenario test site was located at Hail State Blvd., which is composed of over 5 km of highway with high quality lane markings. The scenario was scripted to have a lead vehicle begin approximately 10 m in front of the ego vehicle, accelerating to 45 mph. A 10 m distance is used to emulate the safe distance set for the ACC controller. The ACC controller is still in development and to ensure safe functionality, the minimum safe distance is set to 10 m. As confidence is the system is built, the minimum safe distance will be lowered to 5 m. During the test, the lead vehicle would vary its speed between 40 mph and 50 mph, creating smaller and larger distances to challenge the sensor fusion system. The scenario ends with the lead vehicle coming to a stop and the ego vehicle stopping approximately 10 m behind. Figure 4.12 shows tracking results of the sensor fusion system.

The results from the dynamic scenario shows that the sensor fusion system was able to consistently track the lead vehicle during the entire drive cycle. During the test, the lead vehicle was in-path for the entire duration. The sensor fusion results reflect that a vehicle is in-path even when

Table 4.7

Sensor Fusion Testing Result with White Ford F250.

Scenario	Average Range Error %	Average Range-Rate Error (mph)	Long. Range RMSE (m)	Lat. Range RMSE (m)
1	4.69	0.45	2.93	0.39
2	2.97	0.74	1.96	0.6
3	3.29	0.18	1.68	0.52

road conditions such as curvature and slope were not ideal. It should be noted that during the test, there were multiple occurances of oncoming traffic in the left lane. While plots such as Figure 4.12 show the in-path vehicle, the system is designed for multi-object tracking. Vehicles in the left lane were correctly assigned to the left lane and did not distract or interrupt the system from tracking the lead vehicle.

4.2.5 Performance in MIL vs. VIL

Translating performance from a simulated environment to a real environment is critical for the development process with the overall goal of being able replicate a VIL environment in a MIL simulation. Having an accurate MIL simulation environment allows for rapid algorithm development without taking away vehicle time or resources. To compare the MIL and VIL environments appropriately, a simple approach scenario was used to benchmark the environments. In the scenario, a lead vehicle was stationed approximately 100 m in front of the ego vehicle. Once both vehicles were positioned, the ego vehicle proceeded to approach the lead vehicle at a speed of 10 mph. The scenario was designed and simulated using MathWork's Driving Scenario Designer. In the simulated MIL scenario, ground truth data was collected directly by comparing the ego

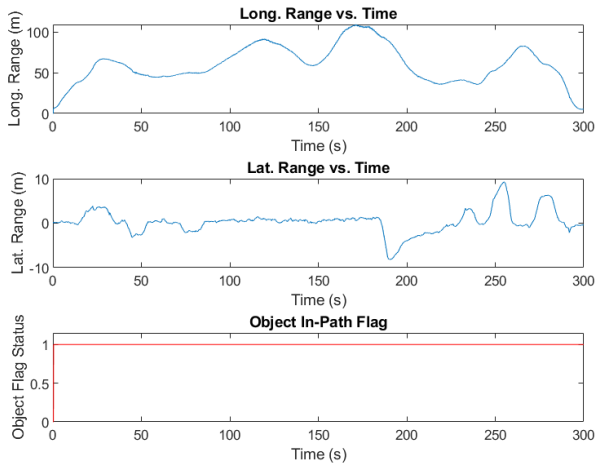
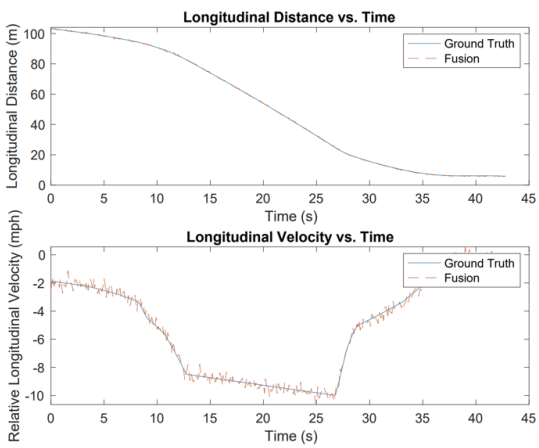


Figure 4.12

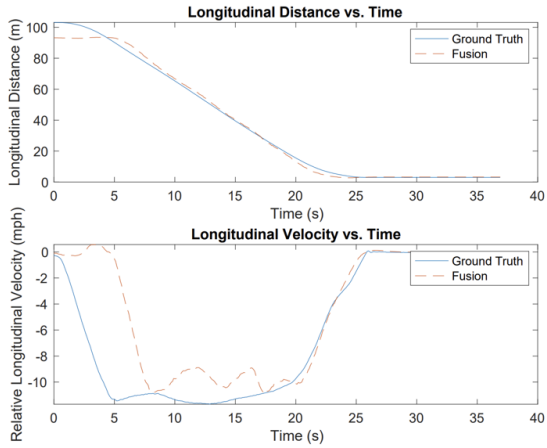
VIL Dynamic Scenario.

vehicle to the lead vehicle. In the VIL scenario, the same sensor fusion system was deployed for real-time functionality. Ground truth data was collected using an OxTS RT3000 unit.

As seen in Figure 4.13, the MIL and VIL testing results are seen for the 10 mph approach scenario. A noticeable difference between the MIL and VIL environments is the sensor measurement characteristics. In the MIL environment, the sensor measurement characteristics were tuned to have less noise compared to the VIL environment. Another difference, which is not shown in the figures, is the radar sensor detection distance. In the VIL environment, the radar is designed to be a mid-range radar with capability of detecting objects at a distance of 80 m. The MIL simulation environment radar sensor was adjusted by increasing the sensor noise and decreasing the detection range from 120 m to 80 m.



MIL Approach



VIL Approach

Figure 4.13

MIL vs. VIL Testing.

4.2.6 Radar vs. Camera Performance

The radar and camera sensors both provide key information to a sensor fusion system. The camera used on the MSU Chevrolet Blazyer, an Intel Mobileye 660, can be seen to provide very accurate longitudinal measurements. The issue with a camera only approach is evident when observing the estimated longitudinal velocity. As seen in Tables 4.6 and 4.7, two different vehicles were used in the same scenarios. In the scenarios with the Black Toyota Camry, the radar sensor generated fewer detections compared to the White Ford F250. In the testing scenarios with the Ford F250, the radar was operating normally and generating detections at a normal range. In the scenarios with the radar operating normally, the velocity measurement is 97 % more accurate compared to the scenarios with the Toyota Camry. While sets of range-rate measurements meet the minimum requirements, it is important to note that the radar sensor is a critical component for accurately measuring longitudinal range-rate.

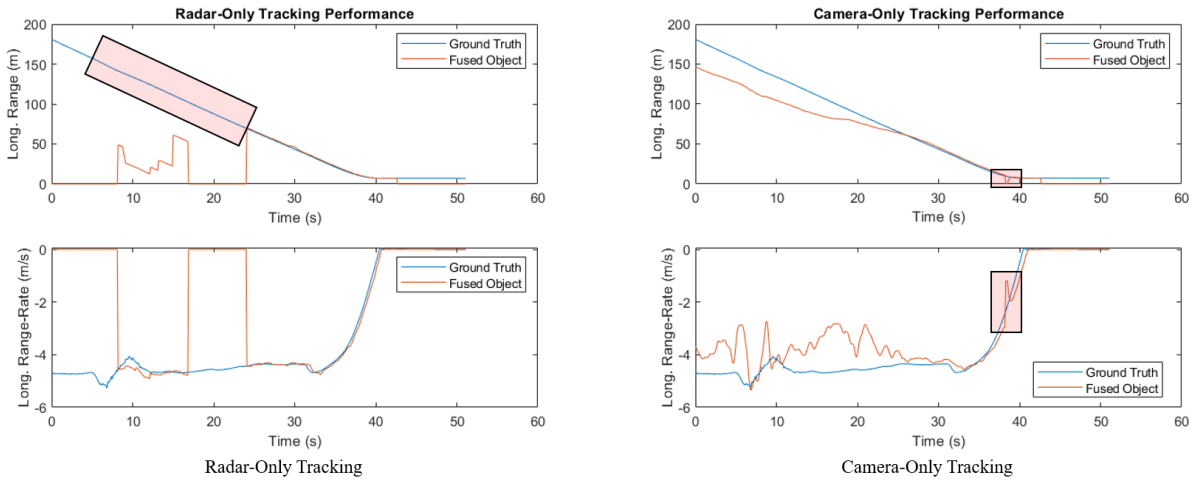


Figure 4.14

Radar vs. Camera Comparison.

Another limitation with the Bosch radar sensor is that is only has mid-range (MR) detection capability. Mid-range sensors automotive sensors are capable of detecting vehicles at distance of 5 m to 85 m where a long-range sensor, such as the Intel Mobileye, is capable of detecting vehicles at distances of 175 m. To test the capabilities of each sensor individually, a MIL simulation was performed where the camera was disabled and the radar was enabled. A second simulation was performed with the radar disabled and the camera enabled. The results of the simulation are highlighted in Figure 4.14. The radar demonstrates accurate tracking at ranges of 5 - 60 m, but does not have the capability for long range detections at 150 m, which is highlighted in red in figure 4.14. The camera is able to track the lead vehicle at 150 m, but the camera is unable to accurately track the longitudinal range and velocity, which is highlighted in red in Figure 4.14. In Table 4.8, a comparison between the radar, camera, and radar-camera sensor fusion system is seen. In all metrics, the radar-camera system outperforms the single sensor systems.

Table 4.8

Camera vs. Radar vs. Camera-Radar Sensor Fusion Results.

Sensor Suite	Average Range Error %	Average Range-Rate Error (mph)	Long. Range RMSE (m)
Camera	7.88	0.63	3.02
Radar	16.82	1.52	27.82
Camera-Radar	4.46	0.39	2.51
Camera % Improvement	**55.42 %**	**47.06 %**	**18.44 %**
Radar % Improvement	**277.13 %**	**289.74 %**	**1008.37 %**

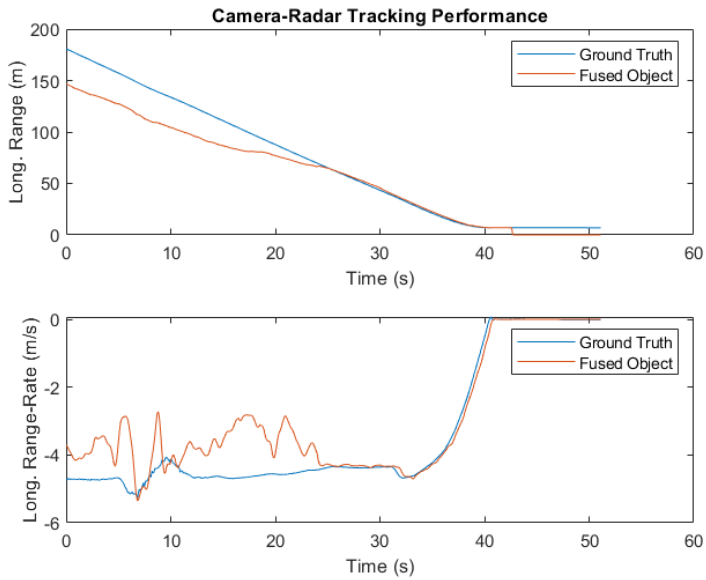


Figure 4.15

Radar-Camera Performance.

Using both camera and radar sensors provide fused result benefits where the full system is able track objects at longer distances (using the camera) and when an object is closer, the system relies

on the radar's measurements since the radar sensor has higher accuracy. The full system approach can be seen in Figure 4.15. Another metric that was not highlighted in the Figure 4.14, but should be noted is lane assignment. In the radar-only system, the lane assignment is not consistent since the system relies on the camera's lane polynomial to project a lane forward. In the camera-radar system, the system takes advantage of the lane assignment that is provided by the camera and the accurate measurements of the radar.

4.3 Sensor Fusion with Adaptive Cruise Control

The final step in system validation is using the sensor fusion system in conjunction with an ACC system. Figure 4.16 shows a high-level dataflow of the ACC system with the sensor fusion and MPC algorithms. A track management component sits between the sensor fusion algorithm and the MPC. In a mult-object fusion system, the in-path object must be identified and its relative longitudinal distance and velocity must be sent the to MPC. The track management component parses the track list for an in-path object and propagates the in-path object to the MPC. If there is not an in-path object, the in-path object fields are set to zero or false, letting MPC know that there is no in-path vehicle follow.

An MPC for ACC from MathWork's Autonomous Driving library was adopted as the primary ACC controller. Before using the MPC in on-road scenarios, several MPC parameters needed to be tuned in simulation. Table 4.9 shows the MPC parameters that were tuned using simulated drive cycles.

Testing the ACC system for on-road functionality occurred over multiple phases. Figure 4.17 below highlights each of the phases used to verify the functionality of the ACC system. As the ACC

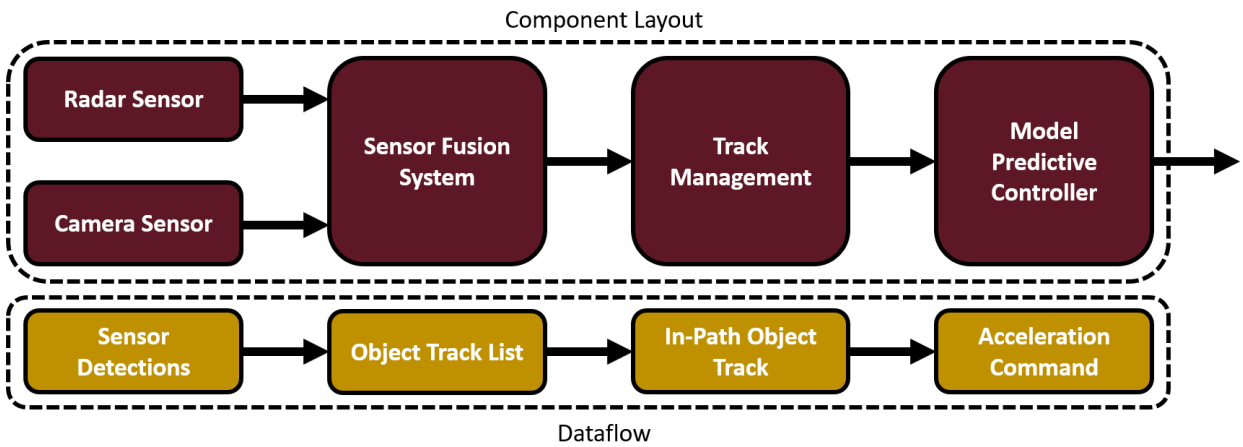


Figure 4.16

ACC System Dataflow.

system completed a phase, more complexity was added to the testing scenarios. The goal of testing over multiple phases was ease the system into more complex scenarios overtime, therefore ensuring the was safe to use. In the first phase, a dynamometer was used to verify that the ACC system could maintain sets speeds with no in-path vehicle. The second phase introduced a simulated vehicle in-path that the ACC system followed while on a dynamometer. The third phase introduced an on-road scenario where the ACC system had to come to a complete stop for an in-path vehicle. This scenario replicated the procedure performed in the sensor fusion approach tests, but with the ACC system commanding vehicle acceleration. In the final phase, a dynamic on-road endurance run was used to demonstrate the full ACC system.

Table 4.9

MPC Parameters.

Parameter	Value	Unit
Default Spacing	6	m
Minimum Acceleration	-5	m/s^2
Maximum Acceleration	5	m/s^2
Prediction Horizon	100	Steps

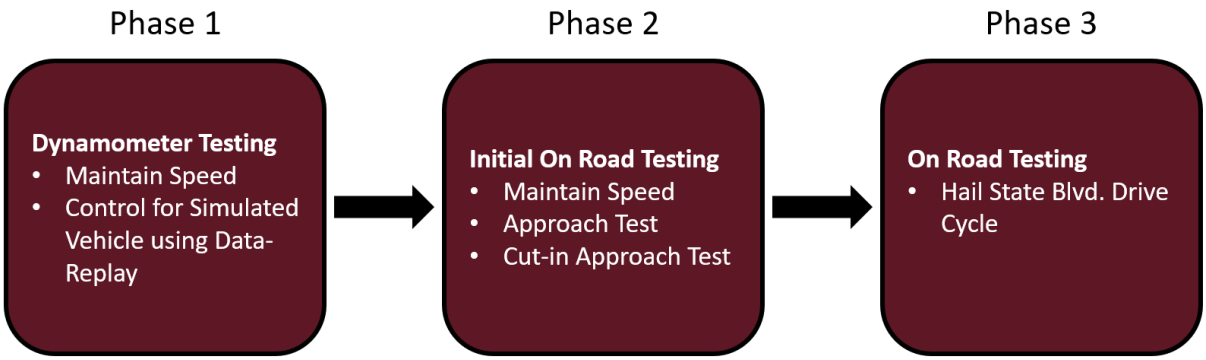


Figure 4.17

ACC On Road Testing Strategy.

The ACC endurance scenario was located at Hail State Blvd. with an in-path vehicle maintaining a velocity of 40 mph. In the ego vehicle, the ACC set speed was set to 45 mph, which forced the ACC system to maintain a safe distance behind the lead-vehicle during the duration of the testing.

Figure 4.18 shows the test results where the ACC system is able to successfully track the lead vehicle throughout the duration of the test as well as maintain a safe distance behind the vehicle.

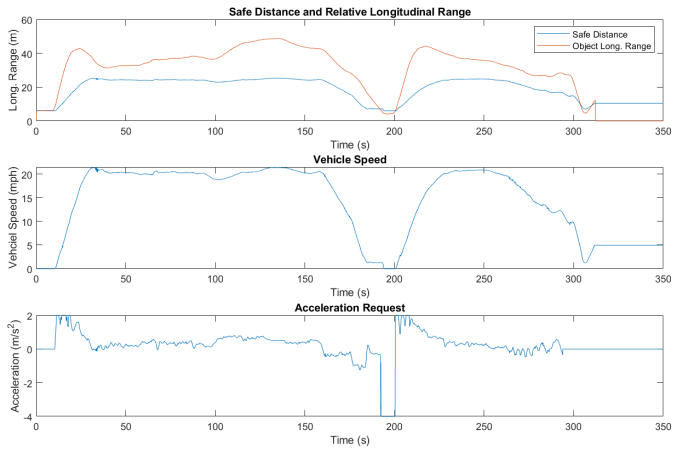


Figure 4.18

Hail State Blvd. ACC On-Road Drive Cycle.

4.4 Conclusions and Future Work

Bringing a sensor fusion system from a set of requirements into a on road vehicle is a challenging task. This section introduces a development and testing methodology on how to bring a sensor fusion system through multiple testing environments and offers novel development workflows for validating requirements across multiple environments. The radar and camera sensor fusion system discussed in the section applied technologies across the MIL, SIL, HIL, and VIL environments to meet all of the system requirements of accurately tracking a lead vehicle for an ACC system. When the sensor fusion system was moved into a real ACC system, there was a seamless transition and the system was able to function safely and comfortably.

Moving into the future, this development methodology can be applied to other sensor fusion systems with different sensor sets. For example, a LiDAR and camera based system can use the workflows presented to collect raw sensor data and ground truth data. Time align the two datasets and perform high fidelity simulations using tools such as Python and Simulink. This process not only introduces rapid mobile development, it also allows for seamless transition into the VIL environment, since the system has been tuned on real sensor data rather than simulated data. There is still a lot of room to improve the existing sensor system by adding sensor that cover blind spots and monitor behind the vehicle. In the addition of more sensors, the practices presented can be applied to validate and improve system tracking capabilities.

4.5 Acknowledgements

I would like to thank the MSU EcoCAR team for enabling this chapter of research. The team leads helped with data collection and implementation of the ACC algorithm discussed in this chapter. I would also like to thank Bosch and Intel, both of whom donated sensor technology to enable this research.

CHAPTER V

TRAFFIC AWARE ADAPTIVE CRUISE CONTROL

5.1 Introduction

This final chapter extends the previously discussed sensor fusion system in the MSU Chevrolet Blazer by adding vehicle-to-intersection (V2I) to the ACC. The ACC system presented in this section not only controls longitudinal velocity for vehicles, but also longitudinal velocity for V2I enabled intersections. This section discusses the DSRC radios used to bring V2X technology into the vehicle, integration of the V2X system, and the testing platforms used to validate the traffic aware ACC system. In previous sections a large emphasis was placed on MIL, HIL, and VIL testing. This section explores component in the loop (CIL) testing, which involves simulating V2X scenarios to achieve seamless integration moving into the VIL environment.

5.2 Methods

5.2.1 Hardware Architecture

The hardware used for the V2X system is a MK5 OBU provided by Cohda Wireless[43]. The MK5 OBU is a DSRC equipped with dual IEEE 802.11p radios and runs Cohda's OBU software which transmits and receives the SAE J2735 V2X [21] message set. The MK5 OBU is connected to the connected and automated vehicle controller, the Intel TANK, which uses the V2X data to autonomously control vehicle longitudinal acceleration. As seen in Figure 5.1, the V2X antenna

Figure 5.1

V2X Antenna Installed on the Mississippi State Chevrolet Blazer.

is installed at the rear of the vehicle. The antenna is capable of receiving the J2735 message set and GPS information.

5.2.2 Software Architecture

On the software side, integrating rudimentary V2X capabilities can be seen in Figure 5.2. A discussed in the previous chapters, in a sensor fusion system for ACC, there must be a component that prioritizes the closest in-path vehicle. To add V2X capability, this software component must be extended to handle upcoming intersections with traffic lights. In Figure 5.2, the prioritization component chooses the closest in-path obstacle. An obstacle could be an object sensed by the sensor fusion system or a upcoming traffic light that is red. The traffic light information is then transformed into an upcoming object. When the a red light occurs, the object information is sent

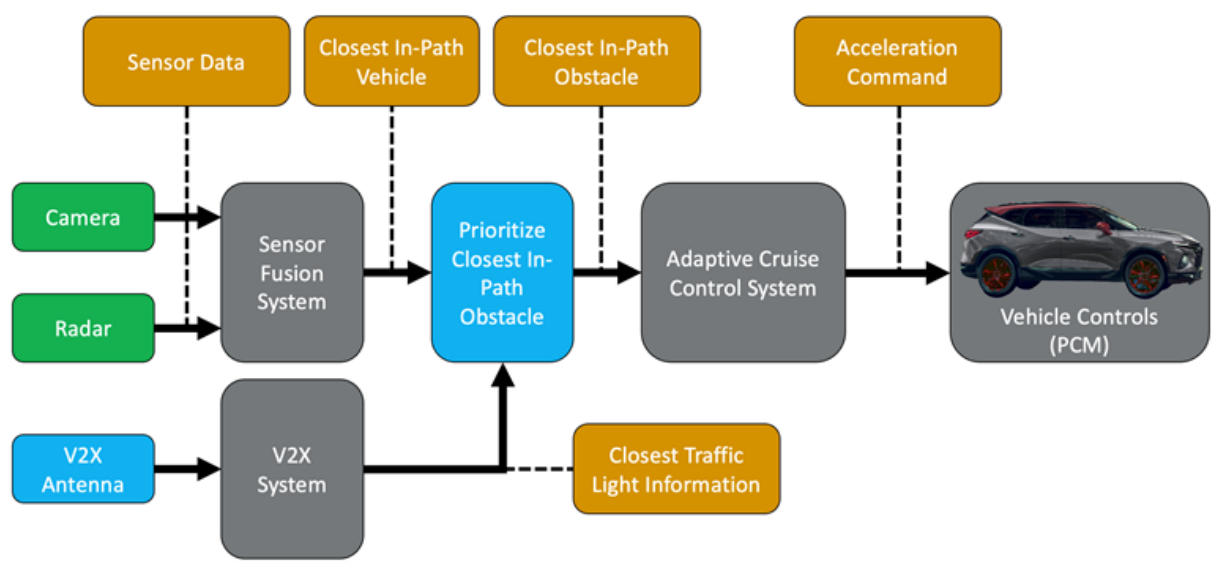


Figure 5.2

ACC with V2X Integration.

an MPC which controls vehicle longitudinal acceleration. This allows the vehicle to autonomously control its longitudinal speed based off of both vehicles and traffic lights.

5.2.3 Green Light Optimization Speed Advisory and Safety Violation Warning

SPaT and Map messages received on an OBU are preprocessed before each message is handed off to the Intel TANK. In the preprocessed stage, the OBU begins by unpacking each message. The OBU calculates which intersections are relevant to the vehicle by associating the vehicle with an ingress lane. The OBU can receive multiple SPaT and map messages, but filtering out non-relevant messages reduces unwanted system confusion.

V2I intersections are composed of an egress and ingress approach lanes. An egress lane is the lane where the vehicle is leaving the intersection. An ingress lane is the lane where the vehicle is approaching the intersection. Figure 5.4 shows a V2I intersection with ingress and egress lanes.

Table 5.1

SVW and GLOSA Data Structure.

Value	Unit
Current Phase	Enumerator
Phase At Arrival	Enumerator
Lane ID	Integer
Time to Change	Seconds
Distance to Event	Meters
Stop Line Latitude	Degrees
Stop Line Longitude	Degrees

If the vehicle is in an egress lane, then the system will ignore the message. When the ego vehicle is in a relevant ingress lane, the OBU will then generate different warnings depending on the ego vehicle's speed and distance to the intersection. If the OBU calculates that the ego vehicle will run a red light at an upcoming intersection, a safety violation warning (SVW) will be generated and sent to the Intel TANK. If the OBU calculates that the ego vehicle is in an approach lane, it will generate a green light optimization speed advisory (GLOSA) warning alerting the vehicle that a relevant intersection is incoming. The data structure for a SVW and GLOSA is seen in Table 5.1. Both message types share a similar data structure, but are generated at different times depending on the vehicle state.

5.2.4 V2X Setup for Component in the Loop Testing

Before taking the V2X system on road, a component in the loop (CIL) testing environment was used to ensure that each hardware and software component was functioning correctly. In Figure 5.3, two MK5 radios are set up approximately 3 m apart. One MK5 is loaded with the OBU

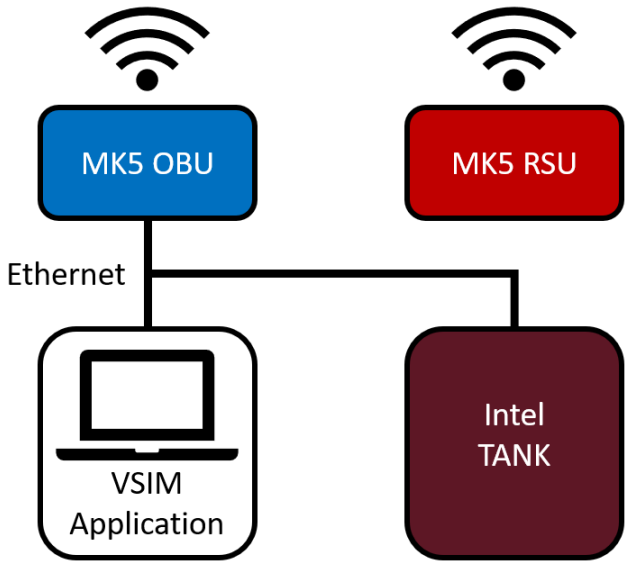


Figure 5.3

V2X Component in the Loop Setup.

firmware and the second MK5 is loaded with road side unit (RSU) firmware. The RSU unit was set to transmit a simple traffic cycle: ten seconds of a "Stop and Remain" (red phase) and fize seconds of "Protected Movement" (green phase).

To simulate an approach vehicle, VSIM, a tool from Cohda Wireless, was used to create the simulated approach. VSIM can also be setup to "spoof" OBU GPS location. This creates a scenario where the OBU thinks it is approaching a V2X-enabled intersection. In Figure 5.4, a picture of the VSIM approach can be seen. During system operation, the RSU would transmit SPaT/Map messages at an interval of One Hz. The messages would be received and processed by the OBU and then sent to the Intel TANK for autonomous longitudinal control.

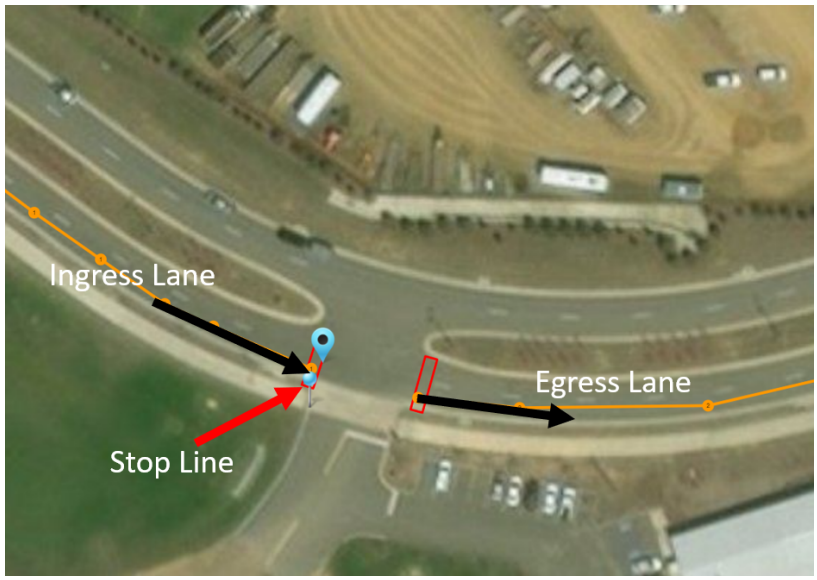


Figure 5.4

Hail State V2I Connected Intersection.

The simulated approach was designed to take place at Hail State Blvd to emulate the same scenario that would be used for on road testing. Figure 5.4 shows the intersection layout with both ingress and egress lanes. By using the CIL setup, many software bugs were fixed in the system before moving to on road testing. This ensured that the system would operate in a predictable and safe manner. The VSIM simulation at Hail State Blvd. is seen in Figure 5.5.

5.3 Results and Discussion

In the CIL environment, the Intel TANK was monitored closely to ensure that data was received, decoded, and transmitted correctly from the MK5. To verify that intersections were being recognized as in path obstacles, an approach test at 20 mph was simulated using VSIM. The results for the test are seen in Figure 5.6, where the intersection is identified as being an in path obstacle when the stop light phase is red and when the phase returned to green, the obstacle

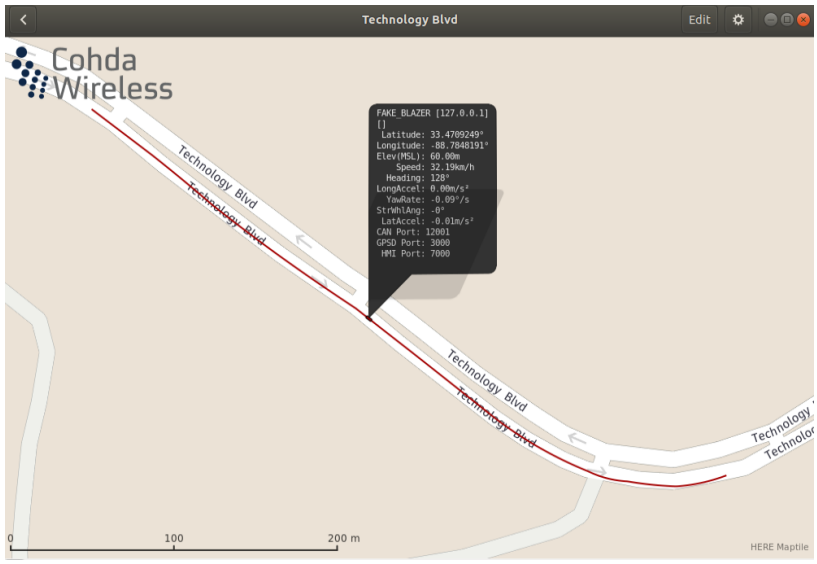


Figure 5.5

VSIM Simulation at Hail State Blvd.

disappears. As seen in Figure 5.6, an SVW is only generated when the OBU sees a red light that the vehicle could run. While the simulated vehicle was not using acceleration commands, testing in the CIL environment verifies: custom SPaT/Map messages, inter-controller communication, and over the air (OTA) communication between the OBU and RSU radios.

Moving into the VIL environment, the same approach scenario as seen in Figure 5.5 was used. In the VIL environment, the ACC system was activated and was enabled to stop for any in-path obstacles. The results for the VIL testing are seen in Figure 5.7 for a 20 mph scenario and Table 5.2. Table 5.2 shows the different system responses when approaching the intersection at varying speeds. The vehicle's traffic aware ACC system was able to function correctly for the three scenarios listed in Table 5.2. As vehicle approach speed increased, the SVW alert was generated at greater distances, giving the vehicle more time to stop before arriving at the intersection. A

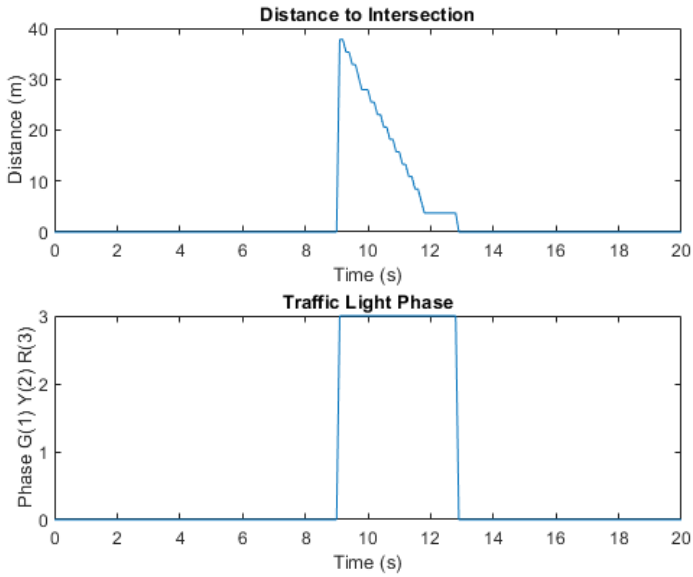


Figure 5.6

CIL 20 mph Approach Test Results.

pass/fail metric was used to judge the vehicle's system response. In all three approach scenarios, the vehicle successfully stopped for the intersection, waited for the correct traffic light phase, and resumed set ACC speed when the correct phase was displayed.

5.4 Conclusion and Future Work

There are many opportunities to develop the V2I system further. The first opportunity would be to use GLOSA alerts instead of SVW alerts. A primary issue for the SVW alert system is lack of information when the vehicle is approaching. As the vehicle approaches an intersection with SVW alerts enabled, a SVW alert is only generated when the OBU calculates that the vehicle will run the proceeding red light. On the other hand, a GLOSA alert is generated when the approach

Table 5.2

Safety Violation Warning Testing Results.

Scenario	Distance for SVW Alert	System Response
20 mph Approach	25.3 m	Stop and Go
30 mph Approach	37.8 m	Stop and Go
35 mph Approach	42.5 m	Stop and Go

vehicle is in an approach lane. This gives the vehicle continuous information about the upcoming traffic light's status, giving the system more time to optimize its approach.

Optimizing for multiple V2I intersections (also known as a connected corridor), has been accomplished in simulation by an MPC presented in [42]. Bringing an MPC that optimizes for multiple V2I intersections to a real-time system presents many challenges. The first challenge is implementing the MPC from [42]. The MPC from [42] was optimized for simulation, where the traffic light phasing was known for the entire connected corridor. In a real world scenario, only current traffic light phase patterns will be known for the upcoming connected corridor. Deploying a real-time connected corridor also is presented as a challenge. V2I technology is not widely used in traffic intersections, therefore a series of custom intersections must be created and synchronously managed.

In conclusion, V2X technology allows for vehicles to optimize vehicle control with high quality data. It brings many safety aspects to drivers and pedestrians through V2I and V2P message sets. While there are many applications, there are also a few downfalls of V2X technology that must be discussed. The first downfall of V2X technology is the requirement for all intersections and vehicles to have V2X-enabled radios. This means bring the Department of Transportation (DOT)

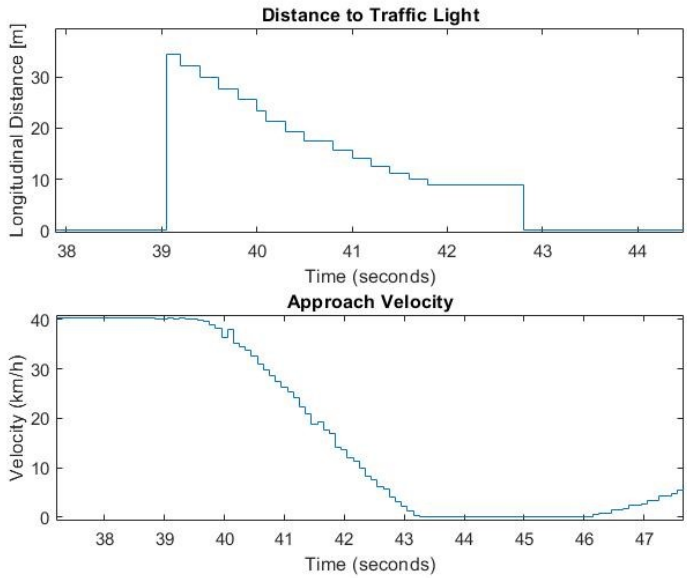


Figure 5.7

VIL Traffic Light Approach at 20 mph.

and industry onto the same page. The second downfall for V2X technology is bad-actors posing as V2I intersections which could compromise a vehicle's autonomous systems.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

In summary, this thesis research covers a wide scope of automotive perception through either sensing or communication technology. An LSTM neural network was developed to perform automotive sensor fusion for a radar-camera system. The LSTM-based sensor fusion was able to perform in a similar manner compared to traditional sensor fusion methods such as Kalman Filtering. During the development several challenges were encountered while developing and training the neural network. The LSTM network required a lot of data before the network began to converge during training. Several data normalization techniques were used to ensure that the neural network trained in an optimal manner.

This thesis also discusses the engineering process to bring a radar-camera sensor fusion for ACC from system requirements to full on road system functionality in a 2019 Chevrolet Blazer. By using the engineering V-Diagram to guide system development through the MIL, SIL, CIL, HIL, and VIL testing platforms, a robust sensor fusion system was developed for ACC that met all initial system requirements. Along with validating the sensor fusion system's performance on road using an OXTS RT3000, the system was also used for ACC in an on road drive cycle. The sensor fusion not only maintained a vehicle track for the duration of the drive cycle, the system provided accurate information to the MPC used to command acceleration. This allowed the MPC to behave in an

optimal manner, without causing any major issues during the on-road test. During the development process of the radar-camera sensor fusion system, the Bosch Radar had issues and did not report detections as stated by the documentation. This limited the sensor fusion system's performance. Moving into the future, this research is being extended with a Delphi ESR 2.5 automotive radar.

By using the engineering process discussed above, a V2X system was integrated and tested to bring V2I capability to the Chevrolet Blazer's ACC system. The V2X system was integrated into the existing sensor fusion system and tested using the CIL and VIL testing platforms. In final test scenarios, the traffic aware ACC system was capable of stopping for a V2I-enabled intersection and autonomously resuming speed when the intersection signals provided the correct signal.

Moving into the future, there are several optimizations that should be made to the traffic aware ACC system. The first optimization is maintaining a safe distance in the ACC application. As seen in testing results, the ACC system struggled to keep up with the lead vehicle and did not command enough acceleration to minimize the error between the relative longitudinal distance and safe distance. Along with optimizing the ACC system's ability to follow a lead vehicle, the ACC system needs to be able to handle multiple V2I intersections and optimize its approach through each intersection. As seen in [42], an MPC was used to optimize a simulated vehicle's position by commanding acceleration through a connected corridor. While connected corridor optimizing has been done in simulation, the optimization has not been completed on road using a real vehicle. By adding real world dynamics to the system, optimizing a vehicle's approach through a connected corridor becomes a nontrivial task. Moving into the final months of the EcoCAR Mobility Challenge, the Mississippi State EcoCAR team hopes to accomplish this using the V2X system in the Chevrolet Blazer.

REFERENCES

[1] "New extension of the Kalman filter to nonlinear systems," *Signal Processing, Sensor Fusion, and Target Recognition VI*, vol. 3068, 1997, p. 182.

[2] "String stability analysis of adaptive cruise controlled vehicles," *JSME International Journal, Series C: Mechanical Systems, Machine Elements and Manufacturing*, vol. 43, 2000, pp. 671–677.

[3] "A Kalman Filtering Tutorial for Undergraduate Students," *International Journal of Computer Science and Engineering Survey*, vol. 08, no. 01, 2017, pp. 01–18.

[4] "Cellular-V2X communications for platooning: Design and evaluation," *Sensors (Switzerland)*, vol. 18, 2018, pp. 1–22.

[5] "Multi-Sensor Fusion in Automated Driving: A Survey," *IEEE Access*, vol. 8, 2020, pp. 2847–2868.

[6] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of DSRC and Cellular Network Technologies for V2X Communications: A Survey," *IEEE Transactions on Vehicular Technology*, vol. 65, 2016, pp. 9457–9470.

[7] R. Amari, M. Alamir, and P. Tona, "Unified MPC strategy for idle-speed control, vehicle start-up and gearing applied to an automated ma," 2008, vol. 17.

[8] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time," *IEEE Transactions on Control Systems Technology*, vol. 19, 2011, pp. 707–714.

[9] ASIRT, "Association for Safe International Road Travel,", 2018.

[10] H. A. P. Blom and Y. Bar-Shalom, "The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients," *IEEE Trans. Automat. Contr.*, vol. 33, no. 8, 1988, pp. 780–783.

[11] S. Campbell, N. O'Mahony, L. Krpalcova, D. Riordan, J. Walsh, A. Murphy, and C. Ryan, "Sensor Technology in Autonomous Vehicles : A review," *29th Irish Signals and Systems Conference, ISSC 2018*, 2018.

[12] F. Castanedo, "A review of data fusion techniques," *Sci. World J.*, vol. 2013, 2013.

[13] E. Charoniti, G. Klunder, P.-P. Schackmann, M. Schreuder, R. D. S. Schwartz, D. Spruijtenburg, U. Stelwagen, and I. Wilmink, "Environmental Benefits of C-V2X for 5GAA-5G Automotive," 2020, p. 77.

[14] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Transactions on Control Systems Technology*, vol. 15, 5 2007, pp. 566–580.

[15] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, 5 1989, pp. 335–348.

[16] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, 2000, pp. 2451–2471.

[17] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, 2020, pp. 362–386.

[18] S. Hochreiter, "Long Short-Term Memory," *vol.*, vol. 1780, 1997, pp. 1735–1780.

[19] D. Hrovat, S. D. Cairano, H. E. Tseng, and I. V. Kolmanovsky, "The development of Model Predictive Control in automotive industry: A survey," 2012, pp. 295–302.

[20] IEEE, "IEEE 802.11-2020 - IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (,".

[21] S. International, "Surface Vehicle," *SAE International*, vol. 4970, 2018, pp. 1–5.

[22] S. Jones, A. Huss, E. Kural, A. Massoner, A. F. Parrilla, L. Allouchery, and N. Wikström, "V2x based traffic light assistant for increased efficiency of hybrid and electric vehicles," *VDI Berichte*, vol. 2016, 2016, pp. 167–177.

[23] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering, Transactions of the ASME*, vol. 82, no. 1, 1960, pp. 35–45.

[24] B. D. Kim, C. M. Kang, J. Kim, S. H. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2018, March 2018, pp. 399–404.

[25] MathWorks, "Driving Scenario Designer,".

[26] MathWorks, "Introduction to Track Logic,".

[27] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, 1957, pp. 32–38.

[28] Z. Radosavljevic, "A study of a target tracking method using Global Nearest Neighbor algorithm," *Vojnoteh. Glas., no*, vol. 2, 2006, pp. 160–167.

[29] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, 2016, pp. 779–788.

[30] I. Revised, T. Related, D. A. Systems, O.-r. M. Vehicles, R. This, S. A. E. O.-r. A. Driving, I. S. O. Tc, and J. W. Group, "SURFACE VEHICLE," 2021.

[31] G. Ripaccioli, A. Bemporad, F. Assadian, C. Dextreit, S. D. Cairano, and I. V. Kolmanovsky, "Hybrid Modeling, Identification, and Predictive Control: An Application to Hybrid Electric Vehicle Energy Management,".

[32] R. Roriz, J. Cabral, and T. Gomes, "Automotive LiDAR Technology: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[33] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, *Model-based deep learning*, arXiv, 2020.

[34] B. Siciliano, O. Khatib, and F. Groen, "The DARPA Urban Challenge: Autonomous Vehicles in City Traffic (Springer Tracts in Advanced Robotics, 56),".

[35] Y. Sun, Z. Wang, Y. Wang, C. Zhang, K. Fu, and J. Ye, *Fusion Recurrent Neural Network*, arXiv, 2020.

[36] A. Taoudi, M. S. Haque, C. Luo, and R. F. Follett, "Design and Optimization of a Mild Hybrid Electric Vehicle with Energy-Efficient Longitudinal Control," *SAE International Journal of Electrified Vehicles*, vol. 14, 2 2021, pp. 55–78.

[37] S. Ullman, "The Interpretation of Structure From Motion,", 2019.

[38] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An overview of autonomous vehicles sensors and their vulnerability to weather conditions," *Sensors*, vol. 21, no. 16, 2021, pp. 1–22.

[39] K. Vedula, M. L. Weiss, R. C. Paffenroth, J. R. Uzarski, and D. Richard, "Maneuvering Target Tracking using the Autoencoder-Interacting Multiple Model Filter," .

[40] Z. Wang, W. Ren, and Q. Qiu, "LaneNet: Real-Time Lane Detection Networks for Autonomous Driving," 2018.

[41] Z. Wang, Y. Wu, and Q. Niu, "Multi-Sensor Fusion in Automated Driving: A Survey," *IEEE Access*, vol. 8, 2020, pp. 2847–2868.

[42] N. Wikström, A. F. Parrilla, S. J. Jones, and A. Grauers, "Energy-Efficient Cooperative Adaptive Cruise Control with Receding Horizon of Traffic, Route Topology, and Traffic Light Information," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, 5 2019.

[43] C. Wireless, "MK5 OBU,".