Driver Assistance System and Feedback for Hybrid Electric Vehicles

Using Sensor Fusion

by

Venkatesh Balaji

A Thesis Presented in Partial Fulfillment
of the Requirement for the Degree
Master of Science

Approved September 2019 by the
Graduate Supervisory Committee:

Lina J. Karam, Chair
Antonia Papandreou-Suppappola
Hongbin Yu

ARIZONA STATE UNIVERSITY

December 2019

ABSTRACT

Transportation plays a significant role in every human's life. Numerous factors, such as cost of living, available amenities, work style, to name a few, play a vital role in determining the amount of travel time. Such factors, among others, led in part to an increased need for private transportation and, consequently, leading to an increase in the purchase of private cars. Also, road safety was impacted by numerous factors such as Driving Under Influence (DUI), driver's distraction due to the increase in the use of mobile devices while driving. These factors led to an increasing need for an Advanced Driver Assistance System (ADAS) to help the driver stay aware of the environment and to improve road safety.

EcoCAR3 is one of the Advanced Vehicle Technology Competitions, sponsored by the United States Department of Energy (DoE) and managed by Argonne National Laboratory in partnership with the North American automotive industry. Students are challenged beyond traditional classroom environment in these competitions, where they redesign a donated production vehicle to meet emission standards and improve energy efficiency while maintaining the features that are attractive to the customer, including but not limited to performance, consumer acceptability, safety, and cost.

This thesis presents a driver assistance system interface that was implemented as part of EcoCAR3, including the adopted sensors, hardware and software components, system implementation, validation, and testing. The implemented driver assistance system uses a combination of range measurement sensors to determine the distance, relative location, & the relative velocity of obstacles and surrounding objects together with a computer vision algorithm for obstacle detection and classification. The sensor system and vision system were tested individually and then combined within the overall system. Also, a visual and audio feedback system was designed and implemented to

provide timely feedback for the driver as an attempt to enhance situational awareness and improve safety.

Since the driver assistance system was designed and developed as part of a DoE sponsored competition, the system needed to satisfy competition requirements and rules. This work attempted to optimize the system in terms of performance, robustness, and cost while satisfying these constraints.

ACKNOWLEDGEMENTS

I would like to acknowledge the invaluable guidance and support of my thesis advisor Dr. Lina J. Karam, in helping me shape this thesis and implement it. I would like to express my heartfelt gratitude to Dr. Antonia Papandreou-Suppappola and Dr. Hongbin Yu for agreeing to serve on my committee and providing valuable suggestions.

I would like to thank Dr. Lina J. Karam and Dr. Abdel Ra'ouf Mayyas for allowing me to be a part of the EcoCAR3 team. I much appreciate the support provided by EcoCAR3 team members Patrick Phillips, Hayden Hostetler, Mitchell Brown, Brandon Larson, Kevin White, and Monica Hsu. A special thanks to my friends and team members Andrew Agrusa, Jared Lovesee, Mattie Whitt, and Emily Arnold, for their constant support and encouragement throughout my journey in the team. My sincere gratitude to Charan Prakash and Tejas Borkar for their help in this project. My heartfelt and sincere thanks to a great friend, Prajwal, for his presence and support throughout this journey.

An exceptional thanks to General Motors, Department of Energy, Argonne National Lab, and all the other sponsors of the EcoCAR3 project and our GeneralMotors adviser Chris Stubbs.

I am incredibly grateful to my parents Balaji and Jayashree, my uncle Dr. Raveendran Rengaswamy, and my elder brother Naresh and his wife for encouraging me to join the project and boosting my confidence through it.

Above all, I thank God for showing me the right path, right people, and right adviser without which this thesis would not have been possible.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Chapter 1

INTRODUCTION

This chapter presents the motivations behind the work in this thesis and briefly describes the contributions and summarizes the organization of the thesis.

1.1 Motivation

One of the integral parts of today's lifestyle, transportation has taken many forms, such as public transportation, private transportation, pooling, etc. Cars play a significant role in transportation. Focusing on the US market, according to [1], 276.1 million cars and light trucks are registered and declared as Vehicle in Operation (VIO) in the year 2018. This number has seen a steady increase over the last few years. With such an increase in vehicles, safety issues have also increased. Situations such as vehicle accidents have led to severe injury, or permanent impairment, or fatality. According to [2], 37,133 deaths have occurred in the year 2017 due to motor vehicles, which is majorly due to distraction while driving. The National Highway Transportation Safety Administration (NHTSA) defines distracted driving as any activity that diverts attention from driving. The activities include talking or texting on the phone, eating or drinking, talking to people in the vehicle, accessing the infotainment system, or anything that takes the driver's attention away from the task of safe driving [3]. To avoid the increasing toll of fatalities caused due to road accidents and improve the safety of transportation, the automotive industry is investing mainly in Advanced Driver Assistance Systems (ADAS) and vehicle autonomy. This includes connected vehicles and semi-autonomous drive modes. The target is to implement full freedom without the need for a human driver's intervention. In today's cars, many systems such as lane departure warning, collision

avoidance, automated brake assist, to name a few, are implemented. These are safety systems that perform the control action based on the situation to avoid an accident.

ADAS and autonomous vehicle research have been increasing exponentially in the past few years. This has increased the need for engineers in the industry to develop, test, and deploy these systems in manufactured vehicles. As an effort to offer hands-on training to students and make them industry-ready, the Department of Energy (DoE), in collaboration with General Motors and Argonne National Laboratories, designed an Advanced Vehicle Technology Competition (AVTC) called EcoCAR3 [4]. In EcoCAR3, students must convert a Chevrolet Camaro car into a hybrid vehicle to improve the energy efficiency, while maintaining the features that make it attractive to the customer: performance, consumer acceptability, safety, and cost. This included powertrain design, mechanical fabrication, control algorithm development, and Advanced Driver Assistance System (ADAS) development.

1.2 Contributions

This thesis presents a driver assistance system that was implemented as part of EcoCAR3, including the adopted sensors, hardware and software components, developed algorithms and system implementation, validation, and testing. The implemented driver assistance system uses a combination of range measurement sensors to determine the range, relative location, and the range rate of obstacles, together with a computer vision algorithm for obstacle detection and classification. In this work, cars are the only obstacles to be detected. The sensor system and vision system were tested individually and then combined within the overall system. Also, a visual and audio feedback system was designed and implemented to provide a front collision warning to the driver, based on the obtained data about the target vehicles, as an attempt to enhance situational awareness and safety.

1.3 Thesis Organization

The organization of this thesis is as follows. Chapter 2 presents background relevant to ADAS with a brief history of ADAS, SAE definitions of different levels of autonomy, color spaces used, a background on object localization, performance evaluation metrics, and communication protocols. Chapter 3 describes the overall implemented ADAS interface, including requirements, sensor selection process, and a block diagram of the developed system. Chapter 4 describes the vehicle detection algorithm and the algorithm for sensor data acquisition and processing. Chapter 5 presents the details of how the system was developed and tested in stages. Chapter 6 concludes the thesis by summarizing the contributions of this work as well as proposing possible future directions of research.

Chapter 2

HISTORY AND BACKGROUND

This chapter presents background information relevant to this work. Section 2.1 provides a history (in the order of occurrence) of crucial developments in ADAS. Section 2.2 describes the different levels of Autonomy as defined by the Society of Automotive Engineers (SAE). Section 2.3 briefly explains the color spaces used in this thesis. Section 2.4 introduces to the concept of localization, and its importance. Section 2.5 explains about ground-truth and how ground-truth data was developed for this thesis. Section 2.6 describes the performance metrics used to evaluate the performance of the object detection algorithm. Section 2.7 introduces the Controller Area Network protocol and parts of the message frame. Section 2.8 presents Serializer and De-serializer pair and its advantages in transmitting image data. Section 2.9 describes Low-Voltage Differential Signaling. Lastly, Section 2.10 briefly describes the Image Signal Processing pipeline that converts raw image sensor data to a perceivable image.

2.1 History of Advanced Driver Assistance System

Research on ADAS has existed since the 1950s. Cadillac developed forward collision warning [5] in the late 1950s. It was implemented in Cadillac Cyclone, a prototype vehicle that used Radio Detection and Ranging (RADAR) technology to identify the obstacles ahead. The idea was viewed to be costly for mass production. Major safety systems such as Anti-Lock Braking System (ABS) [6], Electronic Stability Control were developed in the 1970s and early 1980s. In 1995, a team of scientists and engineers at Hughes Research Laboratories in Malibu, California, demonstrated the first modern, forward collision avoidance system. The technology was marketed as Forewarn [7], a

system based on RADAR was readily available at Hughes Electronics, but not commercially elsewhere. Research in ADAS increased exponentially over the years. It led to the introduction of advanced systems such as the Traction Control System [8], Adaptive Cruise Control [9], and many other active safety systems in the vehicles manufactured. Forward collision assist, automatic parking, adaptive front light, which partially controlled the vehicle or functionality, were introduced in between 2000-2014 [10]. Some of these systems are mandated in the vehicle models nowadays.

With the exponential rise of research in the field of ADAS, many ideas and algorithms were proposed to make the ADAS computationally less intense and bringing the response time of the system to as low as possible. For instance, a lane departure identification system was proposed by Vijay Gaikwad and Shashikant Lokhande [11], in which a Piece-wise Linear Stretching Function (PLSF) is used to improve the contrast level of the Region Of Interest (ROI). The ROI is split into two sub-regions, and lane markers are detected by applying the Hough transform [12] in each sub-region independently. This segmentation approach improves the computational time required for lane detection. A distance-based departure measure is computed at each frame and compared with a threshold. A warning is sent to the driver when the departure measure exceeds the threshold. This algorithm identifies lane departure only using three lane-related parameters based on the Euclidean distance transform to estimate the departure measure. The use of the Euclidean distance transform, in combination with the PLSF, keeps the false alarm around 3% and the lane detection rate above 97% under various lighting conditions.

Traffic sign recognition was an essential part of ADAS. This system comes handy when the driver is unable to notice the correct traffic sign due to movement. This proposed system for traffic sign recognition [13] has two steps: detection and recognition. The color features of the pixels in the detection step are used to detect candidate re-

gions. Next, the cascaded Feedforward Neural Networks with Random Weights (FNNRW) [14] classifiers are utilized for shape and content recognition. This system has an accuracy of 91% with a running time of 40ms.

To study driver's behavior to an incident, an instrumented vehicle, the Japan Automobile Research Institute-Augmented Reality Vehicle (JARI-ARV), was developed to reproduce realistic traffic accident and conflict scenarios without endangering the driver [15]. JARI-ARV was used to study the driver's response to a situation that was encountered, to study the human factors and response.

Another important system that helps drivers park in different situations and check for obstacles close to the vehicle is the 3-Dimensional surround view system [16]. This system uses four fisheye lens [17] cameras to capture images. The captured images are used to generate a 3-D surround view of the driving environment according to the pattern of image acquisition, camera calibration, image stitching, and scene generation.

Original Equipment Manufacturers (OEM) such as General Motors and Mazda have introduced augmented reality windshields that display vital information such as traffic signs and vehicle speed. Jaguar Land Rover has introduced a concept windshield that displays active road region for drivers to avoid lane departure, distance to a target object.

Hybrid vehicle architecture and Electric vehicle architecture then entered the automotive market, and there was a transition towards autonomous driving. Many automotive manufacturers such as Tesla, Inc., Faraday Future, BMW, Mercedes Benz, and others are currently working towards advanced systems and driverless transportation. Tesla Autopilot [18] is a well-known example that shows research done in this field. Also, ride-share companies such as Uber Technologies Inc., Lyft, and tech companies such as Google LLC, Intel Corporation, and Apple Inc., have entered this market and are ruling with cutting edge technology.

2.2 Society of Automotive Engineers Levels of Autonomy

Society of Automotive Engineers (SAE) is a U.S.-based association and standards developing organization for engineering professionals, with emphasis on transport industries such as automotive, aerospace, and commercial vehicles [19]. SAE determines the intelligence level and automation capabilities of the vehicle by categorizing in one of the levels from 0 to 5, as shown in Table 2.1. SAE's level of driver automation indicates the minimum capability for each level. Here, the system refers to the driver assistance system or a combination of driver assistance systems or automated driving systems [20].

Table 2.1: Level of Automation [20] as Defined by the SAE

SAE Level	SAE Name	SAE Definition	Steering, acceleration, deceleration control	Monitoring of driving environment	System Capability
0	No Automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when "enhanced by warning or intervention systems".	Human driver	Human driver	Not Applicable
1	Driver Assistance	The driving mode-specific execution by a driver assistance system of "either steering or acceleration/deceleration".	Human driver and system	Human driver	Some driving modes
2	Partial Automation	The driving mode-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration.	System	Human driver	Some Driving Modes

3	Conditional Automation	The driving modespecific performance by an automated driving system of all aspects of the dynamic driving task with the expectation that a human driver will respond appropriately to a request to intervene.	System	System	Some Driving Modes
4	High Automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task without the expectation that a human driver will respond appropriately to a request to intervene.	System	System	Many Driving Modes
5	Full Automation	The full-time performance by an automated driving system of all aspects of the dynamic driving task under all roadway and environmental conditions that can be managed by human driver.	System	System	All Driving Modes

2.3 Color Spaces

A color space is a geometrical representation of colors in a space that allows the specification of colors as a tuple of three or four numerical values called color components. The various color spaces are mainly divided into the following groups [21]:

1. Primary spaces

   Composed of three primary colors and assumes that it is possible to match any color by mixing appropriate amount of the three primary colors.

2. Luminance-chrominance spaces

   Composed of one luminance component and two chromaticity.

3. Perceptual spaces

   Quantifies human color perception by utilizing intensity, hue, & saturation components.

4. Independent axis spaces

   This group is obtained from different statistical methods.

In this section we will discuss about two color planes used in this thesis, (R, G, B) and (L*, u*, v*) defined by Commission Internationale de l'Eclairage (CIE) and (Y', Cb', Cr').

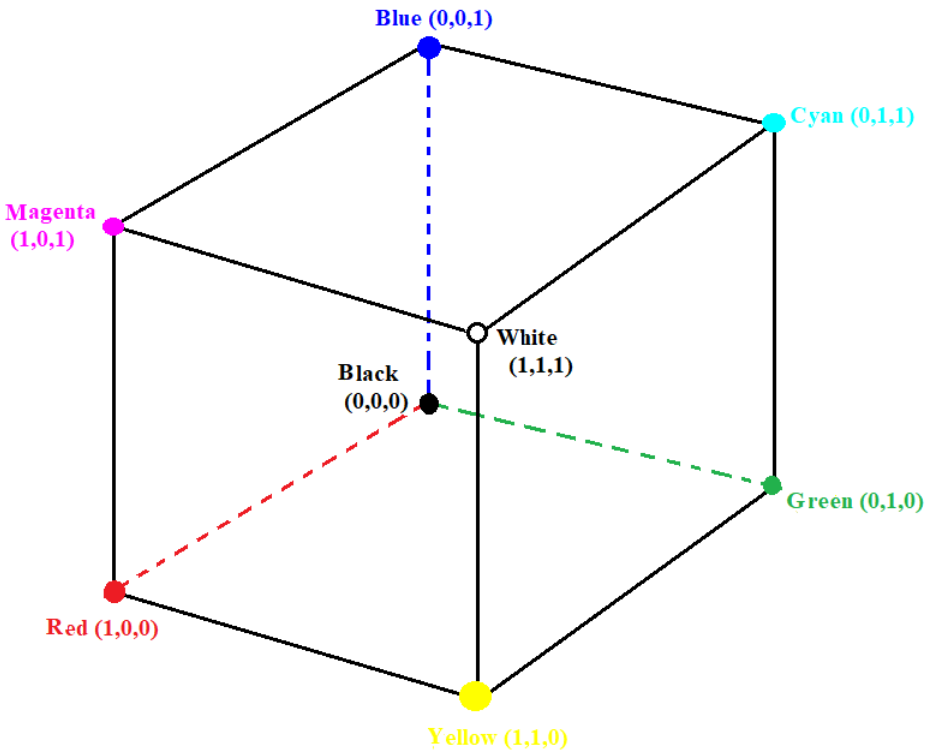

Figure 2.1: RGB Color Cube.

2.3.1 CIE (R, G, B) Color Space

CIE (R, G, B) is a color space composed of three primary colors Red, Green, and Blue. This color space falls under the Primary spaces category, which assumes that any color can be represented as a combination of the primary colors. CIE (R, G, B) is derived from color-matching experiments that used red, green, and blue with wavelengths of 700.0nm, 546.1nm, and 435.8nm, respectively [21].

Figure 2.1 shows RGB color cube with Red, Green, Blue, White, Black, and combinations of two primary colors. The origin point is black color where Rc=Gc=Bc=0 and reference point is white where Rc=Gc=Bc=1. Here Rc, Gc, and Bc are the coordinates.

Achromatic axis is the line segment joining the Black and White points. The intersection of the color cube and the plane represented by the equation Rc+Gc+Bc=1 is an equilateral triangle known as the Maxwell triangle [21]. The primary colors are given by (1,0,0), (0,1,0) and (0,0,1) for Red, Green and Blue respectively.

2.3.2 (L, u*, v*) of CIELUV Color Space*

The (R, G, B) color spaces present some major drawbacks [21]:

1. The tristimulus values and chromaticity coordinates can be negative as it is not possible to match all the colors by additive mixture with a real primary space.

2. There is a large number of (R, G, B) color spaces with different characteristics as they are device dependent.

3. The tristimulus values depend on the luminance, which is a linear transformation of the primary color components.

Hence, CIE defines CIE (X, Y, Z), a color space with the imaginary primary colors (virtual or artificial) to overcome the problems of the primary spaces. However, the Eu-

clidean distances evaluated in the (R, G, B) or (X, Y, Z) color spaces do not correspond to the color differences that are perceived by a human observer [21]. CIELUV color space was developed to overcome this situation. CIELUV is a perceptually uniform color space where L represents the lightness or luminance, and u* and v* are chromaticity coordinates. This color space falls under the category of Luminance-chrominance spaces. The luminance and chrominance components of CIELUV are represented in (X, Y, Z) by the below equations.

$$L^* = \begin{cases} (116 \times \sqrt[3]{\frac{Y}{Y^w}}) - 16 & \text{if } \frac{Y}{Y^w} > 0.008856 \\ 903.3 \times \frac{Y}{Y^w} & \text{if } \frac{Y}{Y^w} \leq 0.008856 \end{cases} \tag{2.1}$$

Where X^W , Y^W and Z^W are the tristimulus values of the reference white.

$$u^* = 13 \times L^* \times (u^{'} - u^{'W}) \tag{2.2}$$

$$v^* = 13 \times L^* \times (v^{'} - v^{'W}) \tag{2.3}$$

$u^{'} = \frac{4X}{X+15Y+3Z}$, $v^{'} = \frac{9X}{X+15Y+3Z}$, $u^{'W}$ and $v^{'W}$ are chrominance components of $u^{'}$ and $v^{'}$ for the reference white respectively.

The reverse transformation is represented by equations

$$Y = \begin{cases} Y_n.L^*(\frac{3}{29})^3 & \text{if } L^* \leq 8 \\ Y_n.(\frac{L^*+16}{116})^3 & \text{if } L^* > 8 \end{cases} \tag{2.4}$$

$$X = Y \times \frac{9u^{'}}{4v^{'}} \tag{2.5}$$

$$Z = Y \times \frac{12 - 3u^{'} - 20v^{'}}{4v^{'}} \tag{2.6}$$

2.3.3 (Y', C'_b, C'_r) Color Space

(Y', C'_b, C'_r) is an international standard for digital video and image coding. Y' is the luminance component, and C'_b and C'_r are the chrominance components. The chromi-

nance components are defined by the equations

$$C_b^* = a_1 \times (R - Y) + b_1 \times (B - Y) \tag{2.7}$$

$$C_r^* = a_2 \times (R - Y) + b_2 \times (B - Y) \tag{2.8}$$

Where a_1, a_2, b_1 and b_2 are coefficients specific to the norms used, standards, or commissions.

The luminance Y' is represented by equation

$$Y^{'} = (0.299 \times R') + (0.587 \times G') + (0.114 \times B') \tag{2.9}$$

International Telecommunication Union (ITU) recommends the use of (Y', Cb', Cr'), as color space independent of the primaries and the reference white and is also used by video and image compression schemes such as MPEG and JPEG with the coefficient values $a1 = 0$, $a2 = 0.713$, $b1 = 0.564$ and $b2 = 0$ according to ITU-R BT.601-7, 2007 [22]. According to ITU-R BT.709-5, 2002 [23] (Y', Cb', Cr') is used for digital coding of high-definition television with coefficients $a1 = 0$, $a2 = 0.635$, $b1 = 0.534$ and $b2 = 0$.

2.4 Object Localization

Object localization is a technique of locating the detected object in a frame. The location of the object is obtained in the form of a bounding box around it. The usual parameters for the bounding box are one vertex, width, and height of the box. Figure 2.2 illustrates object localization. It is important to localize the detected object to know its exact location in the frame, and this information can be further used to fuse the range estimation sensor data. Object localization depends on object detectors, which detect features in a frame. A feature is a piece of information such as shapes, edges, points, or objects in a frame, which is relevant for solving the computational task related to a particular application. The most common reason to choose features over a pixel-based

Figure 2.2: Object Localization With Bounding Box.

system is that features can act to encode ad-hoc domain knowledge, which is difficult to learn using a finite quantity of training data and operates much faster than a pixel-based system [24].

Features can be obtained by detector algorithms such as Haar-like features, a cascade of classifiers. Localization can be achieved using many techniques such as a sliding window or sliding window combined with Adaboost [25]. This section elaborates on some localization techniques.

A primary approach to object detection and localization is a sliding window is a rectangular region that “slides” across an image. The window is designed with a fixed width and height based on the application. Usually, the window region is taken, and an image classifier is applied to determine if the window has an object of interest. This method is computationally costly as object classifier computes feature in one region to

determine the presence of the object and moves to another region to follow the same process.

Another visual object detection framework was developed by Paul Viola & Michael Jones [24]. This framework uses:

1. Integral images, which is obtained by summing the pixels above and to the left of a particular pixel. This image type accelerates the process of rectangular feature computation.

2. AdaBoost, which allows the selection of a small number of important features for classification. This avoids a large number of unwanted features and focuses on critical features by constraining the weak learner so that each weak classifier returned can depend on only a single feature. This results in each stage of boosting to select a weak classifier.

3. A combination of complex classifiers in a cascade structure that focuses on important regions of interest, increasing the speed of detection drastically.

2.5 Ground Truth

Ground truth is the accuracy of the training set's classification in a supervised learning technique. Ground truth information is the information about the target to be detected for comparison with the actual detection to evaluate the performance of the detector.

In images, ground truth labeling is, marking the target object in the image frame. The simplest way to mark the target is to draw a rectangular box around the target. The information stored as ground truth is the location of the target object in a particular frame. The localization data determined by an object detector is compared with the ground truth data, to evaluate the performance.

The primary tool utilized to label ground truth is the MATLAB Ground Truth Labeler App. The Ground Truth Labeler app can be used to label ground truth data in a video or a sequence of images. Rectangular ROIs can be labeled in a sequence of images or video, frame-by-frame. The application also offers automation of the labeling process, where different algorithms can be used to obtain the rectangular ROIs automatically, and the user can accept the obtained result. The ROI labels are used to define locations of objects, such as cars, pedestrians, and lane markers in an image frame. The scene labels are used to define the entire image frame's condition, such as sunny or cloudy, or to mark events such as intersections.

To label ground truth, the below steps have to be performed:

1. A new labeling session is created and an image sequence or video is imported.

2. A Region Of Interest (ROI) label, and a scene label (if needed) is created. This Label has a unique name, which can be used to label a particular target.

3. ROI label is drawn around the target of interest. Automation of labeling the car can also be done, using the algorithms available in the ground truth labeler app, including but not restricted to, Point Tracker. Labeling can also be done manually frame-by-frame by navigating through the frames.

4. After labeling the video or image sequence partially or fully, the labels can be imported to workspace or stored in a file. This would be an object with all the ROI and scene labels as a timetable.

2.6 Performance Evaluation Metrics

Performance evaluation is the process of evaluating the performance of a supervised learning technique by comparing the output of the system with ground truth in-

formation. Multiple quantitative metrics are available to quantify the performance of a detection model.

2.6.1 Intersection over Union (IoU)

Intersection over Union (IoU), also known as Jaccard Index [26] or Jaccard Similarity Coefficient, is a comparison between the similarity and diversity of the sample. As the name suggests, it is the ratio of the intersection of two samples over the union of the same samples.

To evaluate the performance of an object detector that localizes the target, IoU can be computed using the bounding box obtained by ground-truth labeling and predicted bounding box using the below equation.

$$IoU = \frac{Box_{gTruth} \cap Box_{predicted}}{Box_{gTruth} \cup Box_{predicted}} \tag{2.10}$$

Here Box_{gTruth} is the bounding box obtained by labeling the ground truth, and $Box_{predicted}$ is the bounding box output from the object detector.

IoU is a value between 0 and 1, both included, and an IoU close to 1 indicates better object detection and localization. To evaluate the performance of a detection algorithm, usually an IoU of 0.5 or higher is considered, and this value is called threshold.

2.6.2 Detection Conditions

Detection condition depends on the class of the target being detected. In this work, the target is cars present in the image frame, so there is only one class of targets. Considering car as the target object, detection conditions can be classified into four categories defined as follows:

1. True Positive (TP): When a car is present and detected, the detection is called true positive. Bounding box obtained from detection, with an IoU above threshold, is considered TP.

2. False Positive (FP): When a car is not present but detected, the detection is called false positive. Bounding box obtained from detection, with an IoU below threshold, is considered FP.

3. True Negative (TN): When a target that does not belong to the class of cars is present and detected, the detection is called the true negative. This is not applicable in this work as there is only one target class.

4. False Negative (FN): When a car is present but not detected, then this is called false negative. In this case there is no bounding box obtained from detection.

2.6.3 Precision

Precision measures how accurate the predictions of the object detection model is. It is the ratio of true object detection to the total number of objects detected. Below equation shows the computation of Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.11}$$

Higher precision score indicates more likelihood of the detected targets being correct.

2.6.4 Recall

Recall, also known as Detection Rate is a measure to qualify the object as detected. Recall is computed as follows:

$$\text{Detection Rate} = \frac{\text{True Positives}}{\text{Total number of ground truths}} \tag{2.12}$$

To compute the recall at a given IoU value for the entire dataset, the recall of individual sample is computed and averaged.

2.7 Controller Area Network

The Controller Area Network (CAN) is a communication protocol that allows Electronic Controller Units and Sensors to communicate with each other in automotive applications. It is a low cost, robust communication created by Robert Bosch in the year 1986. History of CAN development can be found in [27].

Every ECU connected to the network is known as a node & each node requires a

1. Host processor, which decrypts the received message and decides what message should be transmitted.

2. CAN Controller, usually part of the microcontroller, that stores the received serial bits from the start, until entire message is received.

3. Transceiver that converts the bitstream from CAN bus level used by CAN controller and vice versa.

Since many nodes are connected to the bus at the same time, not all nodes transmit data at once. The process of a particular node occupying the bus is called arbitration. A bit-wise arbitration takes place which determines the priority of the node that has to occupy the bus. Bit value 0 is dominant, and bit value 1 is recessive. Arbitration is nondestructive, meaning the node that wins the priority continues to transmit, while the node that transmitted recessive bit and failed to occupy the bus waits for its turn to transmit the data. A dominant bit always overwrites a recessive bit on a CAN bus. Figure 2.3 shows two nodes connected to the bus with the architecture of each node.

A CAN message frame (2.0A format), as shown in Figure 2.4 has 9 parts explained below

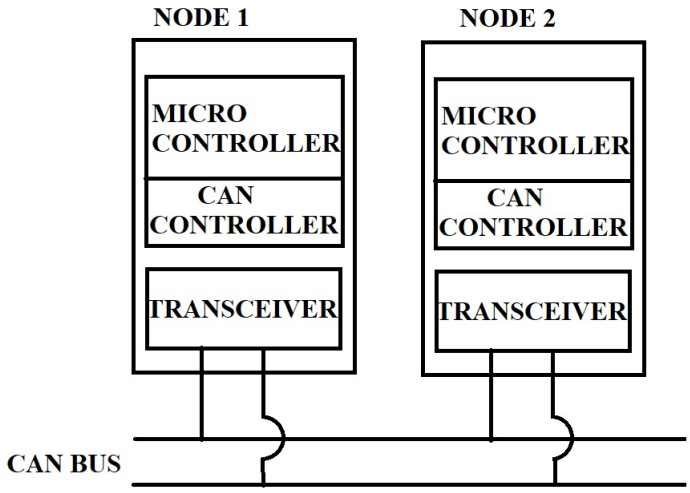

Figure 2.3: CAN Bus Nodes and Its Internal Architecture.

SOF 1bit	ID 11bit	RTR 1bit	r0 1bit	DLC 4bit	DATA 64bit	CRC 16bit	ACK 2bit	EOF 7bit

Figure 2.4: CAN Message Frame (CAN 2.0A).

1. SOF: Start Of Frame (SOF) is 1-bit information that marks the start of a message. A dominant 0 informs the arrival of a message to other nodes.

2. CAN ID: The Standard CAN 11-bit identifier establishes the priority of the message. The lower the binary value, the higher its priority.

3. RTR: Remote Transmission Request (RTR) bit indicates a request of information from other nodes.

4. r0: Reserved bit (for possible use by future standard amendment).

5. DLC: The 4-bit data length code (DLC) contains the number of bytes of data being transmitted.

6. Data: Up to 64 bits of application data may be transmitted.

7. CRC: The 16-bit (15 bits plus delimiter) cyclic redundancy check (CRC) checks data integrity.

8. ACK: Acknowledgement from every node receiving an accurate message. A node receiving an accurate message overwrites this recessive bit in the original message with a dominant bit. ACK is 2 bits, one is the acknowledgment bit, and the second is a delimiter.

9. EOF: This End Of Frame (EOF), 7-bit field marks the end of a CAN frame.

Although the CAN message frame has nine parts, the most important parts that are logged and shown to users are CAN ID, DLC, and Data. Another CAN message format, also known as 2.0B, supports both 11 bit (standard) and 29 bit (extended) identifiers.

CAN Flexible Data-Rate (CAN FD) released in early 2012, is an extended version of CAN, in which each message frame can hold up to 64 bytes of information, unlike classic CAN message that holds up to 8 bytes of information.

The Socket CAN [28] package is an implementation of CAN protocols for Linux. CAN is a networking technology that has widespread use in automation, embedded devices, and automotive fields. The CAN socket API has been designed as similar as possible to the TCP/IP protocols to allow programmers, familiar with network programming, to learn how to use CAN sockets easily.

Received CAN messages can be filtered based on the message ID using the following syntax

struct can_filter {

canid_t can_id;

canid_t can_mask;};

where can_id is the message ID and can_mask is the mask A filter matches, when

$< received\_can\_id > \& mask == can\_id \& mask$

which is analogous to known CAN controllers hardware filter semantics. 0 to n receive filters for each open socket can be set separately as:

struct can_filter rfilter[2];

rfilter[0].can_id = 0x123;

rfilter[0].can_mask = CAN_SFF_MASK;

setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));

CAN_SFF_MASK means the mask of a Standard Frame Format (11 bit message ID) should be set. If the message ID matches with the filter can_id, those messages will be received and messages with other message ID will be discarded.

2.8 Serializer : De-Serializer Pair (SerDes)

Serializer and De-serializer pair (also known as SerDes [29]) is used to serialize the data from the image sensor for serial transmission and de-serialise the data at receiver end. The principle is shown in Figure 2.5.

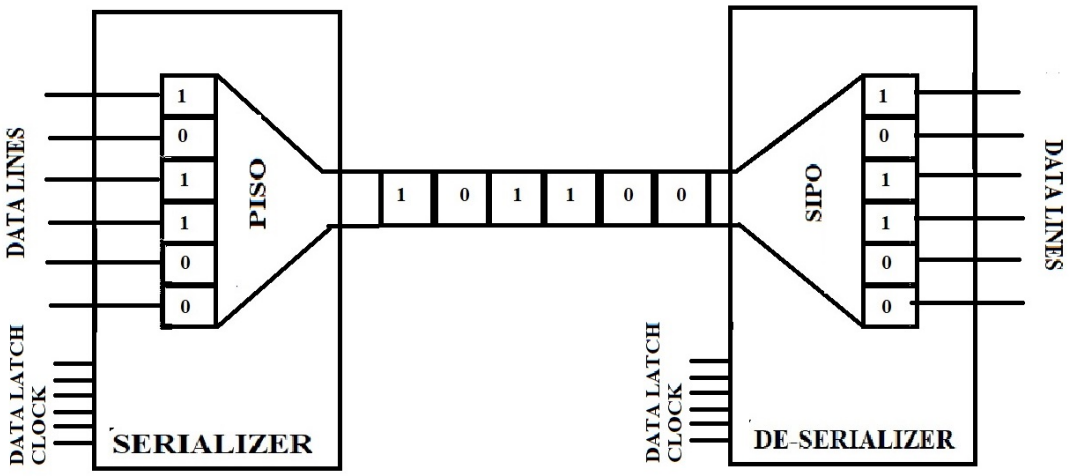

Figure 2.5: SerDes Principle.

The setup typically consists of a Parallel Input, Serial Output block that has a parallel clock input, a set of data input lines, and input data latches, and a Serial Input, Parallel Output block typically has a receive clock output, a set of data output lines and output data latches. The co-axial (COAX) cable that connects the serializer and de-serializer is a differential pair cable, and bitstream is transmitted by Low-Voltage Differential Signaling (explained in Section 2.9).

The input latch and clock in the serializer are used to synchronize the parallel bitstream to serial bitstream. At the receiver, the same clock is used to identify the serial bit stream and convert it to a parallel bit stream. The latch in the de-serializer is to create an appropriate delay to reconstruct the parallel bitstream.

The purpose of this setup is to reduce the number of input/output pins between the sensor and the processor/receiver. Serial data transmission also has the advantage of higher signal to noise ratio and less reconstruction error with long-range transmission.

2.9 Low-Voltage Differential Signaling

Low-Voltage Differential Signaling (LVDS) is an efficient way of transmitting bit values. Figure 2.6 shows the LVDS setup. At the transmitter side, there are four transistors A, B, C, and D driven by a current source I_A. At the receiver side, the voltage V_O is the received signal. R is the impedance resistor used to obtain the differential voltage.

Table 2.2: LVDS Truth Table.

Bit value	A	B	C	D	V_O
1	1	0	0	1	$+0.35V$
0	0	1	1	0	$-0.35V$

Table 2.2 shows the excitation of the transistors to transmit bit 0 and 1. When bit value 1 needs to be transmitted, transistor A and D need to be closed, so the current

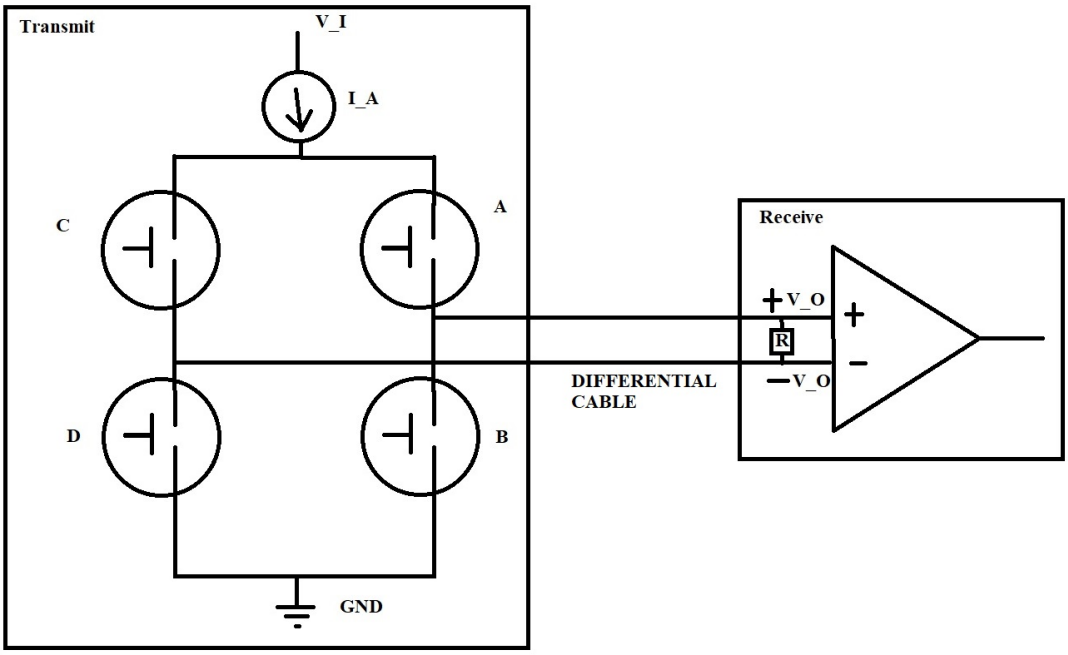

Figure 2.6: LVDS Setup.

goes through A-R-D-GND, where GND is ground. When bit value 0 needs to be transmitted, transistor B and C need to be closed, so the current goes through C-R-B-GND. At the receiver, +0.35V or -0.35V is received when bit 1 and 0 are transmitted, respectively.

Since the signal is a differential voltage, transmission noise will not affect the data. This is because bit 1 and 0 are distinguished based on the voltage polarity in the receiver. This makes LVDS effective for long-range transmission.

2.10 Image Signal Processing

Image Signal Processing (ISP) is an important component in camera data acquisition and processing. ISP is used to obtain human perceivable images from raw data obtained from an image sensor.

The photodiodes in an image sensor have different color filters: red, green, and blue (RGB) in a pattern called Bayer filter (shown in Figure 2.7). As each photodiode records the color information for exactly one pixel of the image, without an image processor, there would be two green pixels next to each red and blue pixel.

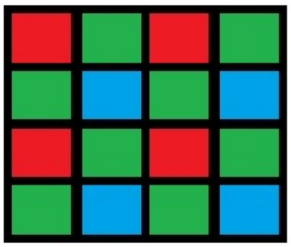

Figure 2.7: Top View of 8x8 RGBG Bayer Pattern.

To obtain a human perceivable image, the raw image data goes through a process [30]. Raw image data from an image sensor is converted to digital format at the sensor level, and the digital version of raw data is transmitted to the processing unit. At the processing unit, the raw data undergoes white balancing. White balancing removes unrealistic color casts so that white objects are rendered white in the image. Then Demosaicing is performed. Demosaicing is the process of interpolating the raw information to estimate the color intensity in each pixel. For instance, in a Green pixel of Bayer filter, the information about Red and Blue is obtained by using the information from neighboring Red and Blue pixels. Demosaicing is performed in by various methods such as simple interpolation or pixel correlation with an image [31] [32] [33]. Further, to obtain a better image quality, noise in the image data is reduced, and the image is sharpened by detecting the edges and reproducing them smoothly. Finally, color space conversion is performed to obtain a YUV image.

Figure 2.8 shows the general process required to obtain an RGB image from raw image sensor data.

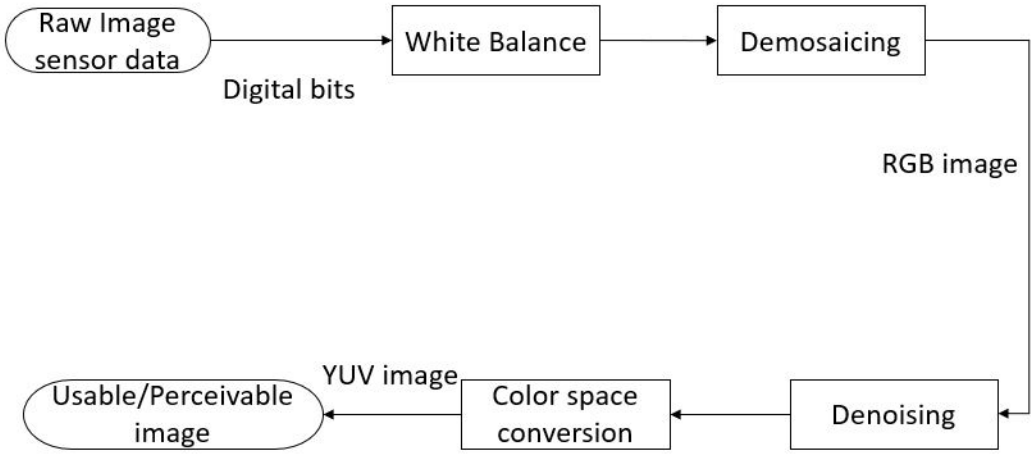

Figure 2.8: ISP Pipeline.

Chapter 3

SYSTEM OVERVIEW

This chapter provides an overview of the designed and implemented advanced driver assistance system. Section 3.1 lists the requirements of the driver assistance system (the data that the system must be capable of determining, for a target vehicle). Section 3.2 outlines the various factors that impacted the range measurement sensor selection, and provides a specification table that lists all the sensors considered as part of the selection process. Section 3.3 presents information about the hardware and interface that are used to implement the ADAS, along with a top level description of the system, block diagram of the components and Field of View (FoV) diagram.

3.1 Requirements

In this section, the term "ego vehicle" refers to the considered vehicle with ADAS. Any other vehicle is referred to as the "target vehicle". The lane in which the ego vehicle is driven is known as the "host lane". The lanes that are adjacent to the host lane are referred to as the "right lane" or "left lane". A target vehicle can be in an adjacent lane or in the host lane.

Target vehicles must be detected using image processing and the following data [34] about any vehicle ahead of the ego car must be determined:

1. Longitudinal Range: distance between the lines passing laterally through the centers of target & ego vehicle (measured as A in Figure 3.1).

2. Lateral Range: distance between the lines passing longitudinally through the centers of target & ego vehicle (measured as B in Figure 3.1).

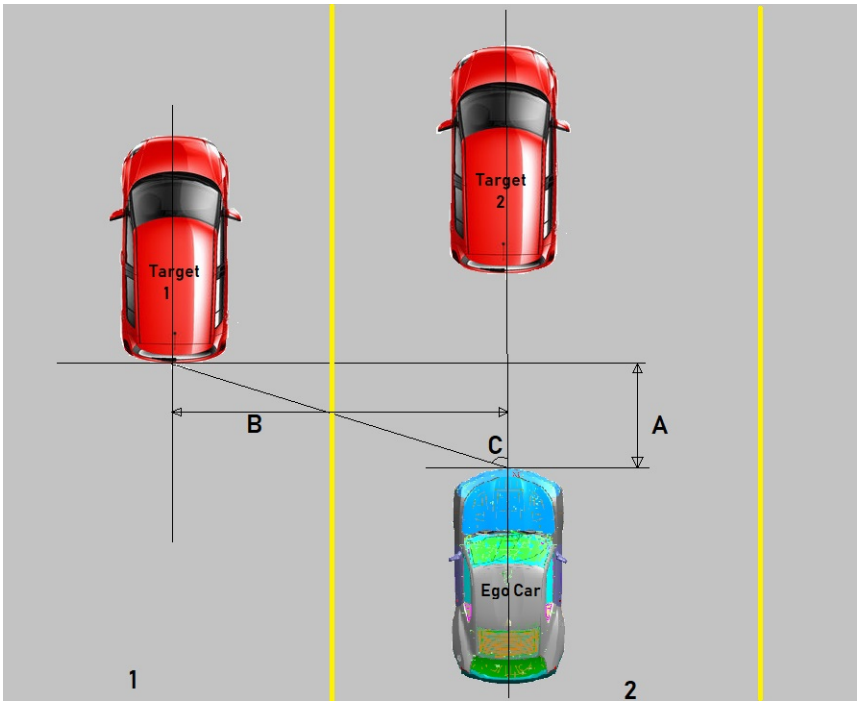

Figure 3.1: Top View Representation of the Requirements. A: Longitudinal Distance, B: Lateral Distance and C: Azimuth Angle of Target 1 With Respect to Ego Vehicle.

3. Azimuth angle: angle at which the target is located with respect to the ego vehicle (measured as C in Figure 3.1).

4. Lane Position: the lane within which the target vehicle is with respect to the ego vehicle. The lane position of the target is the host lane if the target is in the same lane as that of the ego vehicle or left/right lane if the target is in adjacent lanes.

5. Range Rate: velocity of the target vehicle with respect to the ego vehicle.

The above data corresponding to a given target vehicle must be transmitted to the vehicle's Controller Area Network (CAN) so that the test vehicle is aware of the detected obstacle and its dynamic properties. A top view illustration of the requirements is shown in Figure 3.1, where A is the longitudinal distance, B is the lateral distance, C is the Azimuth angle of Target 1 with respect to the Ego Car. 1 and 2 refer to the left

lane and host lane, respectively. Lane position, Range and Range rate estimation will be discussed in Chapter 4.

3.2 Sensor Selection

Selecting a sensor for range measurement was a major part of this work as multiple technical and logistical factors affected the selection process. The selected sensor must not only be capable of delivering the required performance, but also should be compatible with the rest of the hardware and interface. Also, the sensor selection involved the consultation of mechanical engineers to analyze mechanical and structural compatibility.

Factors that influenced sensor selection are discussed below.

3.2.1 Sensor Capability

Various properties such as Field of View (FoV), communication interface, and other physical attributes of sensors were studied carefully to evaluate the performance and usability of the sensors. The FoV refers to the region of the scene that can be "seen" by the sensor.

Table 3.1 shows various sensors and key properties that were analyzed as part of the sensor selection process. Sensors with an adequate detection range, Field of View (FoV), light weight, smaller form factor, multiple communication interfaces and lower cost were favored for the proposed ADAS implementation.

Table 3.1: Sensor Specifications.

	Sensor (Sensor Type) Manufacturer Cost	Range Data Refresh Field of View (FoV)	Physical Size Weight Recommended placement	Communi -cation Interface	Environments Physical qualities
1	HDL-64E [35] (LIDAR) Velodyne Lidar $75,000	120m/ Rotation Rate: 5 Hz-20 Hz/ 360^0 x 26.9^0	283 x 203 x 180 mm/ 12.7 Kg/ Roof	100 Mbps Ethernet	Lighting matters for detection. Rotating scanner.
2	HDL-32E [36] (LIDAR) Velodyne Lidar $10,000	80-100m/ Rotation Rate: 5-20 Hz/ 360^0 x 41.3^0	85 mm D x 144 mm H/ 1.0kg/ Roof	100 Mbps Ethernet	Lighting matters for detection. Rotating scanner.
3	VLP-16 [37] LiDAR Puck (LIDAR) Velodyne Lidar $8,000	100m/ Rotation Rate: 5âĂŞ20 Hz/ 360^0 x 30^0	103 mm D x 72 mm H/ 830g/ Roof	100 Mbps Ethernet	Lighting matters for detection. Rotating scanner.
4	UXM-30LX-EW [38] (LIDAR) Hokuyo Automatic Co., Ltd. $5,165	30m/ Rotation Rate: 20 Hz/ 190^0	124 x 126 x 150 mm/ 800g/ Roof	Ethernet 100BASE-TX	Lighting matters for detection. Rotating scanner.
5	UST 10LX [39] (LIDAR) Hokuyo Automatic Co., Ltd. $1700	30m/ Rotation Rate: 40 Hz/ 270^0	50 x 50 x 70 mm/ 130g/ Roof	Ethernet 100BASE-TX	Lighting matters for detection. Rotating scanner.

6	ARS441 [40] 76/77 GHz Wavelength (RADAR) Continental $3400	250m, 70m, 20m/ 16.7Hz/ $18^0, 90^0, 150^0$	137 x 91 x 31 mm/ 295g/ Grille or Front fasica	CAN	Immune to ambient light. No moving parts.
7	Leddar Evaluation Kit [41] (LEDDAR) LEDDAR TECH $300	50m/ 100Hz/ 45^0 x 7.5^0	114 x 76 x 46 mm/ 265g/ Grille or Front fasica	RS-45 and CAN bus interfaces	Immune to ambient light. No moving parts.
8	Leddar One [42] (LEDDAR) LEDDAR TECH $115	40m/ 140Hz/ 3^0 beam	3 x 50.8 mm D/ 14g/ Grille or Front fasica	3.3 V UART or RS-485	Immune to ambient light. No moving parts.
9	Leddar Vu8 [43] (LEDDAR) LEDDAR TECH $690	100m/ 100Hz/ horizontal:20^0 vertical:0.3^0	70 x 35.2 x 67.5 mm/ 110g/ Grille or Front fasica	USB, CAN, serial	Immune to ambient light. No moving parts.
10	Leddar Vu8 [43] (LEDDAR) LEDDAR TECH $690	40m/ 100Hz/ horizontal:100^0 vertical:0.3^0	73 x 35.2 x 62.3 mm/ 128.5g/ Grille or Front fasica	USB, CAN, serial (UART/RS- 485)	Immune to ambient light. No moving parts.
11	OV10635 [44] (CAMERA) NXP Semiconductors $300	15 - 30 frames/second 60^0 x 40^0	25 x 25 x 31 mm/ 128.5g/ Front windshield	Serial over COAX	Poor performance in bad weather. No moving parts.

3.2.2 Structural Modifications of Vehicle

According to [45], the following structural modifications were prohibited or restricted:

1. cutting any structural part of the vehicle body structure;
2. making modifications to vehicle chassis structure that may affect safety or vehicle dynamics;
3. implementing a new vehicle structure that could affect safety or vehicle dynamics;
4. implementing a structural design that violates any competition rule.

Structural modifications were restricted at the roof top. Although mounting sensors on the vehicle's roof was not completely restricted, heavy sensors might lead to undesirable long-term structural modifications. For this reason, heavy sensors that required roof top mounting were avoided.

3.2.3 Logistical Factors

Many factors such as cost of the sensor, wait time for sensor shipment, technical support from the manufacturer also affected the selection of range estimating sensors.

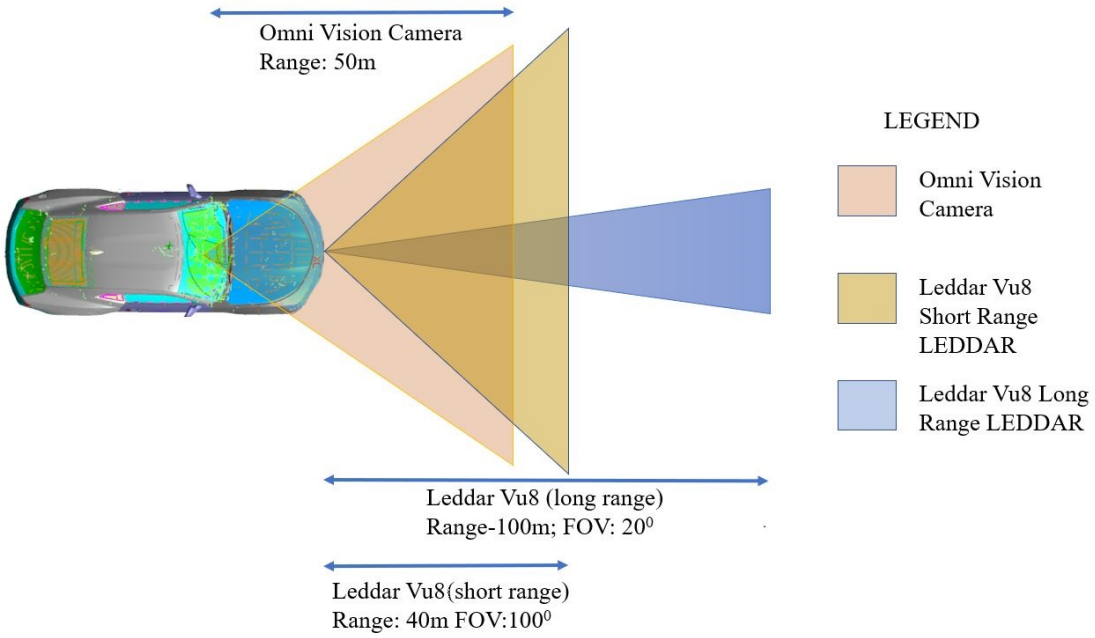

Figure 3.2: Top View of Sensors' Field of View.

3.3 System Hardware Description

To meet the requirements mentioned in Section 3.1, range measurement sensors were used along with camera to detect vehicles. Figure 3.2 illustrates the field of view (FoV) of all sensors used to implement the ADAS. Two Leddar Vu8 sensors (Row Items 9 and 10 in Table 3.1) with different range and FoV configurations are used to cover the front region. An OmniVision OV10635 camera (Row Item 11 in Table 3.1) is used for obstacle detection using image processing. The FoV cones originate from the location where the corresponding sensor is mounted.

The block diagram of the implemented system is shown in Figure 3.3. The system consists of four major blocks as follows:

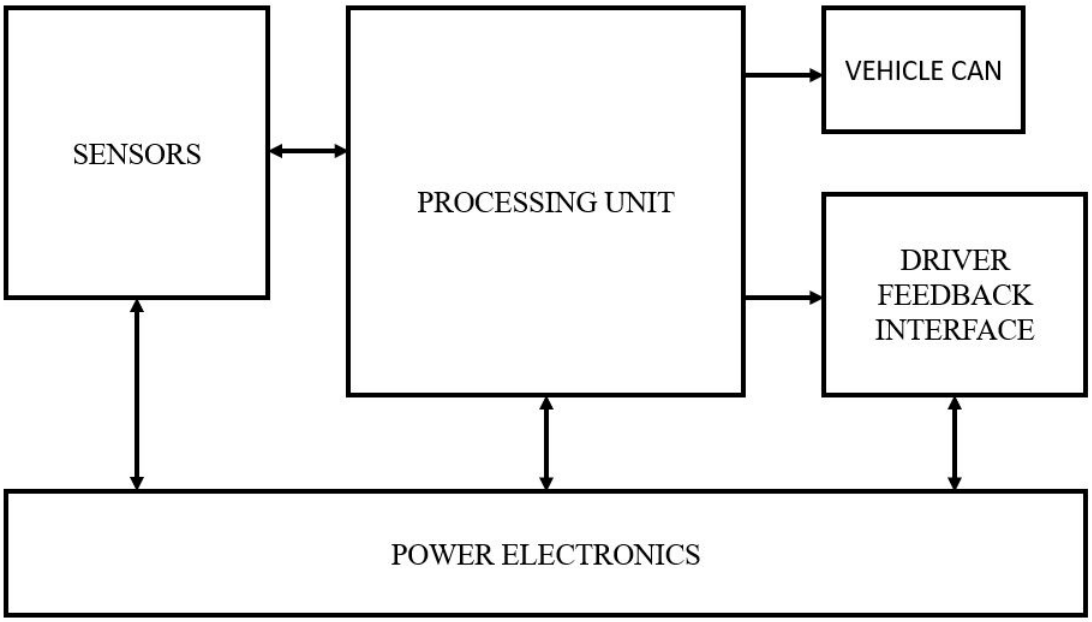

Figure 3.3: Block Diagram of the System.

3.3.1 Sensors

Sensors include an Omnivision OV10635 camera for object detection & two Leddar Vu8 for range measurement. The OV10635 camera has a MAX96705 [46] serializer that serializes the data and is connected to the processing unit through a coaxial cable with FAKRA connector [47] and a Maxim MAX9286 Deserializer. Section 2.8 explains the process of data serialization & de-serialization. FAKRA (also known as Fachkreis Automobil, a German standard) connectors are SubMiniature version B (SMB) [48] based automotive-grade connectors. Figure 3.4 shows the components in the camera (image sensor, de-serializer and connector).

Leddar Vu8 sensors are connected to the processing unit through the Controller Area Network (CAN) ports using twisted pair.

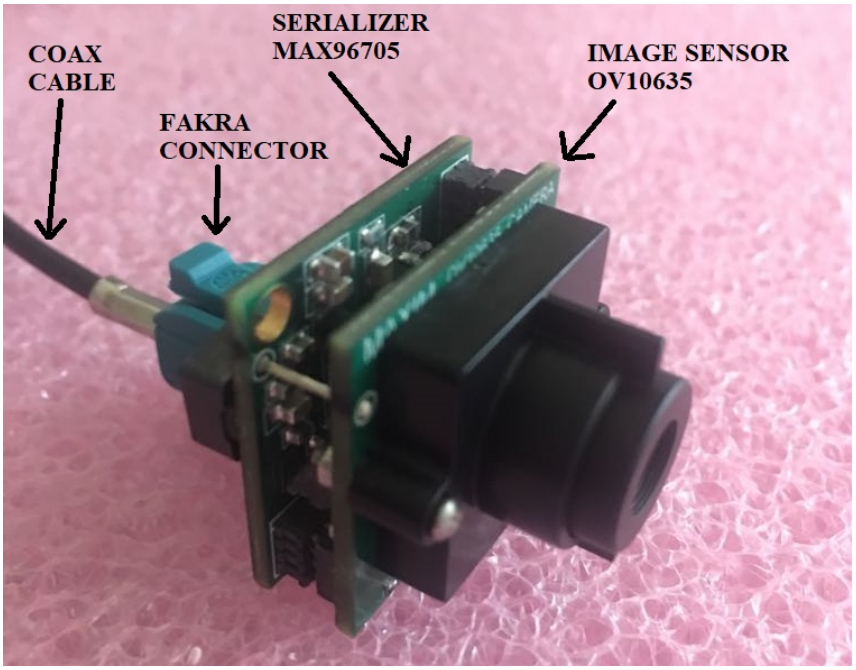

Figure 3.4: Components of the Camera.

3.3.2 Processing Unit

The processing unit is the NXP S32V234 (Treerunner), a vision processor manufactured by NXP Semiconductors. Treerunner is equipped with an ARMÂő CortexÂő-A53 quadcore, 64-bit CPU, and an ARM Cortex-M4, 32-bit CPU. It supports various communication protocols such as Universal Asynchronous Receiver Transmitter (UART), Inter Integrated Circuit (I2C), Ethernet, Flexible Data rate Controller Area Network (FD-CAN), FlexRay and Local Interconnect Network (LIN). It also has an Image Signal Processing (ISP, purpose explained in Section 2.10) that supports 2x1 or 1x2 MegaPixel at 30 frames/sec and two APEX2-CL Image cognition processor, each of which comprises two Array Processing Unit (APU) cores [49]. A MAX9286 Gigabit multimedia serial link (GMSL) deserializer, which supports up to four cameras connected through 50Ω resistance COAX connection, was used to de-serialize the image information received from the camera for the serial output interface.

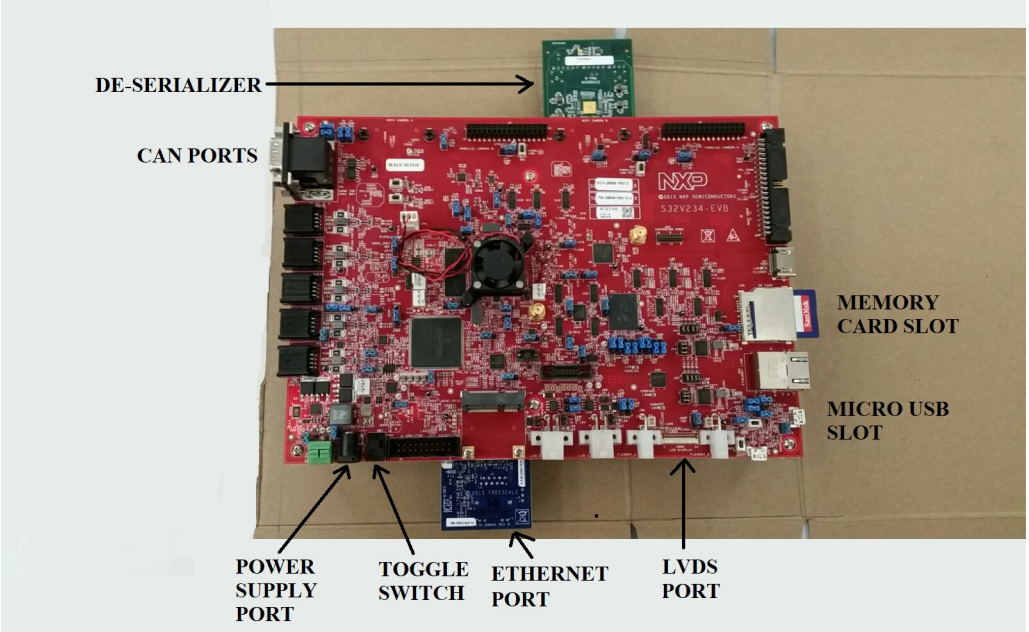

Figure 3.5: Front View of NXP S32V234 (Treerunner), With Labels of All the Ports Used. Picture Taken While Designing Enclosure.

As shown in Figure 3.5, Treerunner contains 2 CAN ports one of which is used to communicate with two Leddar Vu8 sensors and the other port is used to transmit the information to the vehicle CAN bus. Appendix B provides more details about the CAN message from Vu8 sensors. MAX9286 has a Mobile Industry Processor Interface-Camera Serial Interface (MIPI-CSI) connector, through which it is connected to TreerunnerâĂŹs MIPI-CSI port.

Figure 3.7 shows how the camera is connected to Treerunner through the Maxim De-serializer The power requirements of Treerunner is 12V with a maximum of 4A current while functioning at full capacity. The data transmitted from Treerunner to the vehicle CAN is received by the ETAS ES910, hybrid controller, which is used to synchronize and control the combustion engine and electric motor. This unit also controls the driver feedback interface.

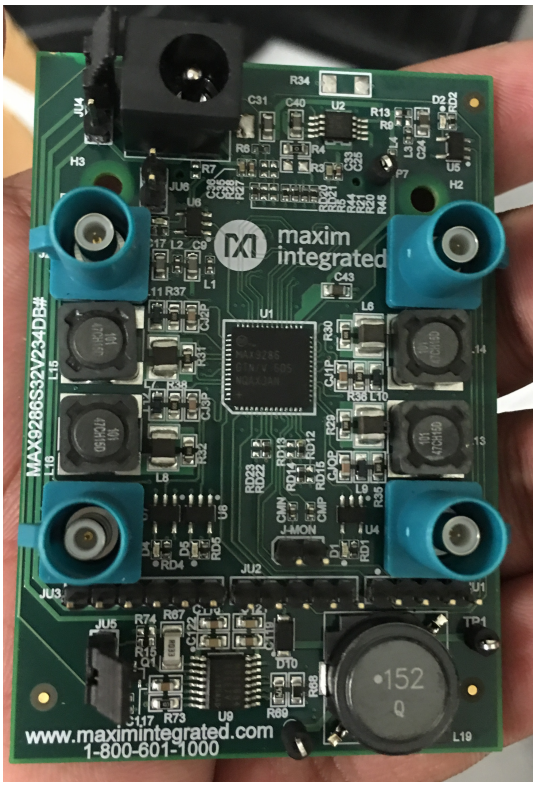

Figure 3.6: Maxim Integrated MAX9286 De-Serializer That Expands the Camera Inputs to Treerunner. Picture Taken While Designing Enclosure.

3.3.3 Driver Feedback Interface

The designed driver feedback interface uses LEDs and a Buzzer, and is controlled by ETAS ES910, which generates a control signal based on the transmitted ego-to-target distance data, to provide visual and audio warning to the driver when the distance between the ego vehicle and the target vehicle in the host lane is below a critical threshold.

3.3.4 Power Electronics

Components used to control the power to all the sensors, processing units and feedback interface elements fall under this block. These components were used to ensure the safe and proper functionality of all the power sensitive hardware.

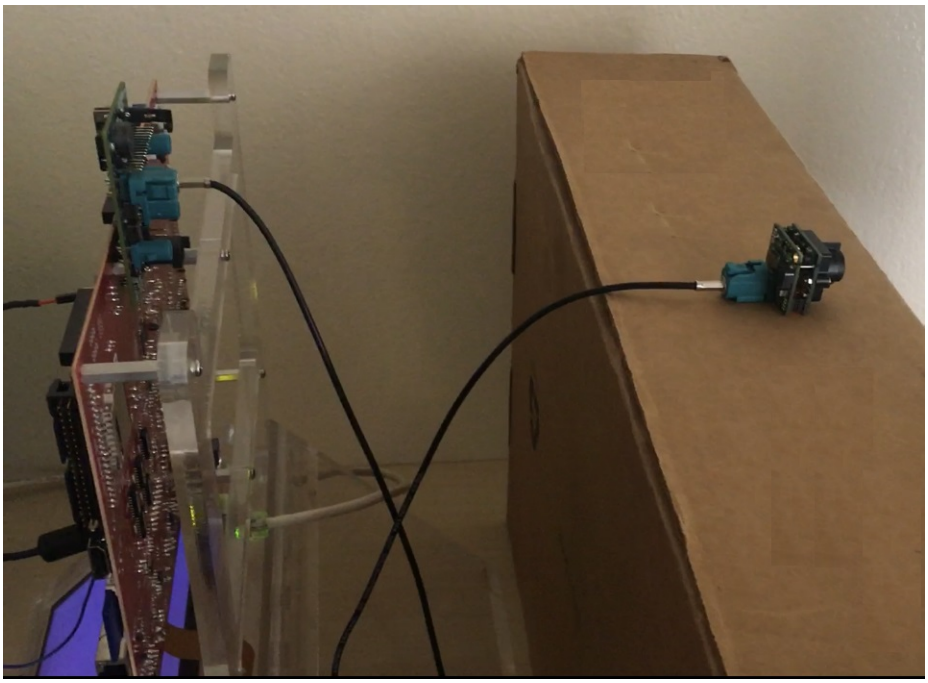

Figure 3.7: Picture of Setup Showing Camera Connected to Treerunner Through the MAX9286 De-Serializer.

Chapter 4

ALGORITHMS AND SOFTWARE FOR OBSTACLE DETECTION

This chapter presents the algorithms and software implementation of the two considered ADAS tasks: 1) Vehicle detection using camera feed, and 2) Sensor data acquisition and processing using NXP S32V234. Section 4.1 covers the details of the vehicle detection algorithm and implementation. Section 4.2 describes the CAN data frame including the request transmitted from the host and the response from the sensor, decoding of CAN data bytes and computation of dynamic data such as range, azimuth and lane position. Section 4.3 describes the data transmission from the NXP Treerunner board to the vehicle CAN bus.

4.1 Vehicle Detection

Vehicle detection is one of the main requirements of an ADAS system. One aim of obstacle detection is to categorize the localized obstacle. In this work, the only targets to be identified are vehicles. This task is achieved using an algorithm called Aggregated Channel Feature (ACF) [50].

A block of the ACF detection framework is shown in Figure 4.2. The ACF detection framework has five major steps including pre-processing, channel feature computation, feature aggregation, flattening and boosting. In the pre-processing step, the input image I (Figure 4.2(a)) of width W and height H is smoothed (Figure 4.2(b)) using a low pass filter ($[1,2,1]/4$). Smoothing is needed to suppress irrelevant fine scale details and image noise [51]. It also determines the scale of subsequently computed gradients [52]. This is followed by a feature extraction step (Figure 4.2(c)), in which several types of features are extracted from the smoothed image; each feature type is referred to as a

channel [52]. Ten feature channels are computed, including normalized gradient magnitude, histogram of oriented gradients (6 channels in total, each channel represents one of the six gradient orientation bins), and LUV color channels. The ACF method extracts features at several scales by forming a feature pyramid for the considered image. A feature pyramid is a multi-scale representation of an image wherein features (i.e., channels) are computed at various scales. A higher scale corresponds to lower spatial resolution. Scales are sampled evenly in log-space, starting at s = 1, corresponding to the original input image spatial resolution, with typically 4 to 12 scales per octave. An octave is the interval between one scale and another with half or double its value. In the ACF method, a fast feature pyramid computation method is adopted.

The computed channel features are then aggregated by first dividing the channels into 4 x 4 blocks and pixels in each block are summed and smoothed to obtain lower resolution channels (Figure 4.2(d)). The feature channels computed in all scales, are then vectorized and provided to decision trees. Finally boosting is used to train and combine decision trees over these features to distinguish object from background and a multi-scale sliding-window approach is employed. Non-maximal suppression (NMS) [53] is used to remove multiple bounding boxes for the same object and obtain a final bounding box that represents a target.

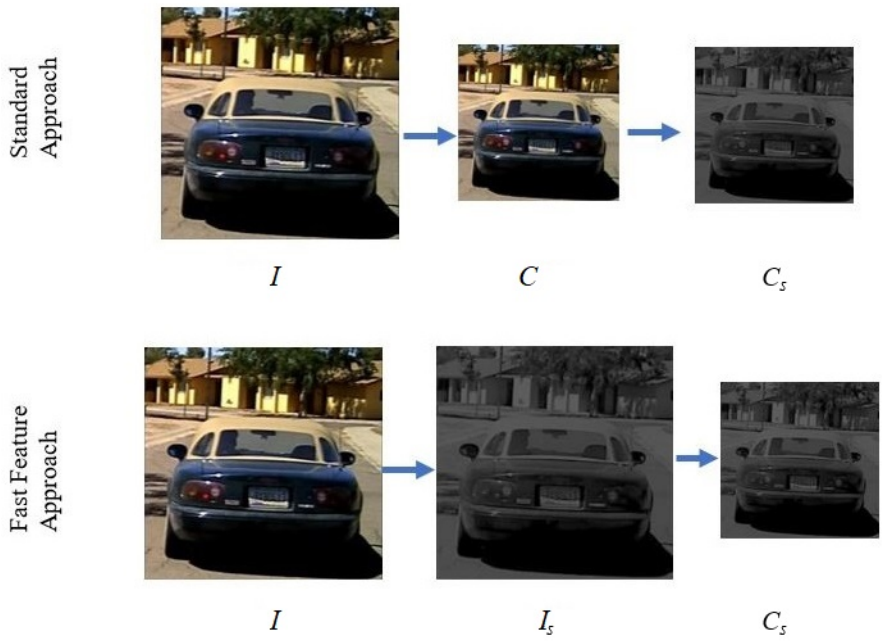

Figure 4.1: Standard Approach vs Fast Feature Approach of Feature Computation. Top: For Each Scale s, Input Image I Is Re-Sampled at Scale s to Produce I_s and Its Feature Channel(s) C_s Is (Are) Computed. Bottom: The Feature Channel(s) C of an Input Image I Is (Are) Computed at Scale s=1, and Feature Channels C_s at Other Scales, Is (Are) Produced Through Scaling and Transformation of C to Approximate C_s.

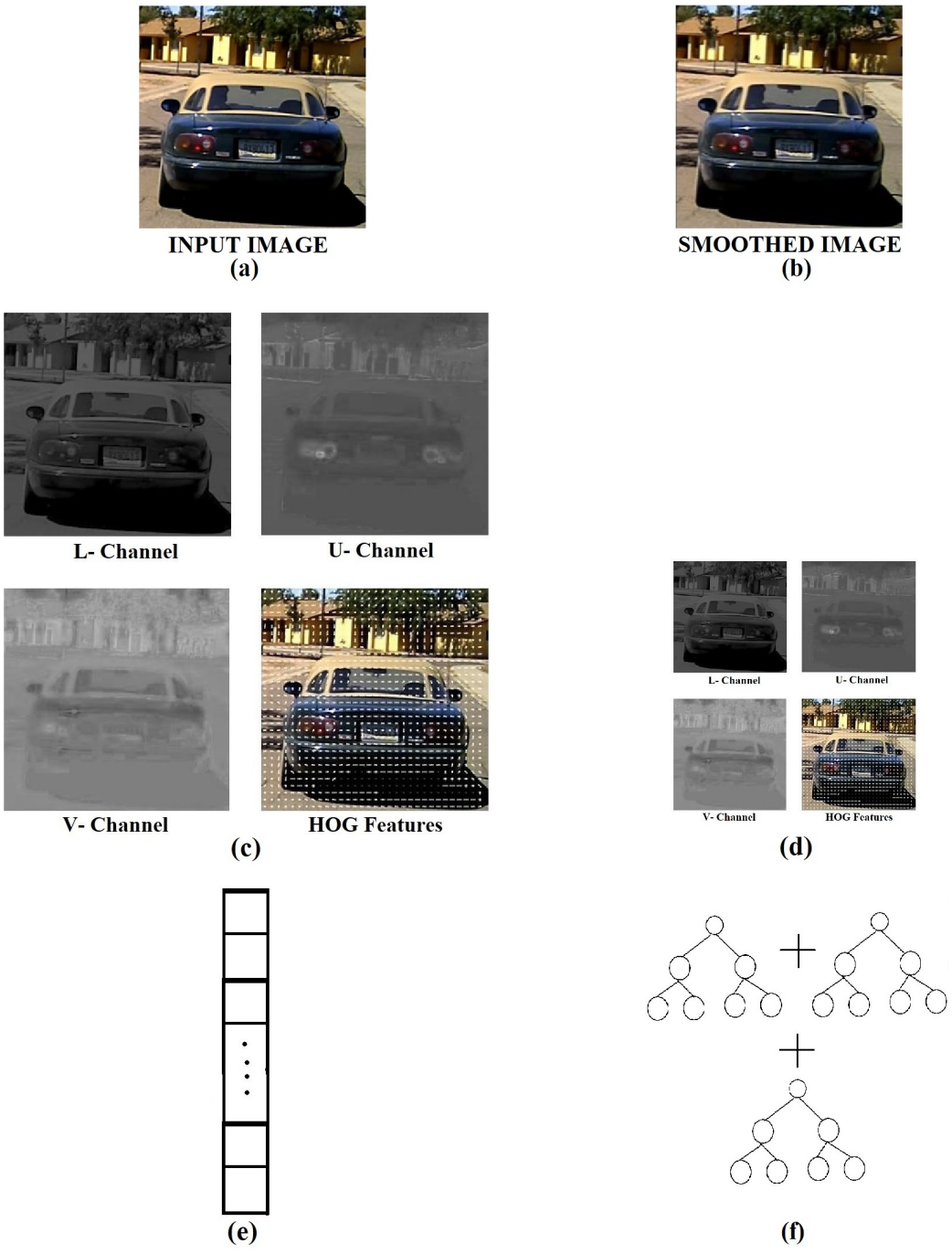

Figure 4.2: Illustration of the Main Steps of ACF. (a) Input Image. (b) Smoothed Image. (c) Computed Channels (Normalized Gradient Magnitude Channel Not Shown). (d) Aggregated Low Resolution Channels. (e) Flattening. (f) Boosted Tree.

Figure 4.3: Output Frame With the Detected Bounding Box Plotted Around the Target Vehicle (Same Sample Image as Shown in Figure 4.2).

Figure 4.1 illustrates the fast feature pyramid computation approach through fast feature scaling, where the feature channels are computed at the lowest scale (s=1) and then these features are scaled to produce feature channels at various scales in the octave, without the need to extract features at each scale. Such approach offers a good trade-off between speed and accuracy.

The output obtained after all the computation is a list of bounding box coordinates with corresponding confidence scores. The obtained bounding box is of the format [*x, y, width, height*], where x and y are the coordinates of the top left vertex of the bounding box in the image coordinate system. This bounding box information can be utilized to plot a graphical box on the image frame to visually observe the location and the overlap of detection. Another use of this bounding box is to evaluate the performance of the object detection algorithm, by computing performance metrics such as IoU as discussed in Section 2.6.1, and categorizing the detection as true positive or false positive. Figure 4.3 shows a sample image with the computed bounding box plotted around the target.

The Leddar Vu8 sensor's field of view (FoV) is split into 8 segments as shown in Figure 4.4. Communication between the Leddar Vu8 and the Treerunner was established using CAN. Consider an obstacle to be present in segment 0. The sensor senses the obstacle and reports the distance D value as shown in Figure 4.4 and the segment in which the obstacle is present, which in this case is segment 0. This message is received as a CAN message as further discussed below. A request message is sent via CAN to the Leddar Vu8 and the sensor responds with the data acquired at that instance.

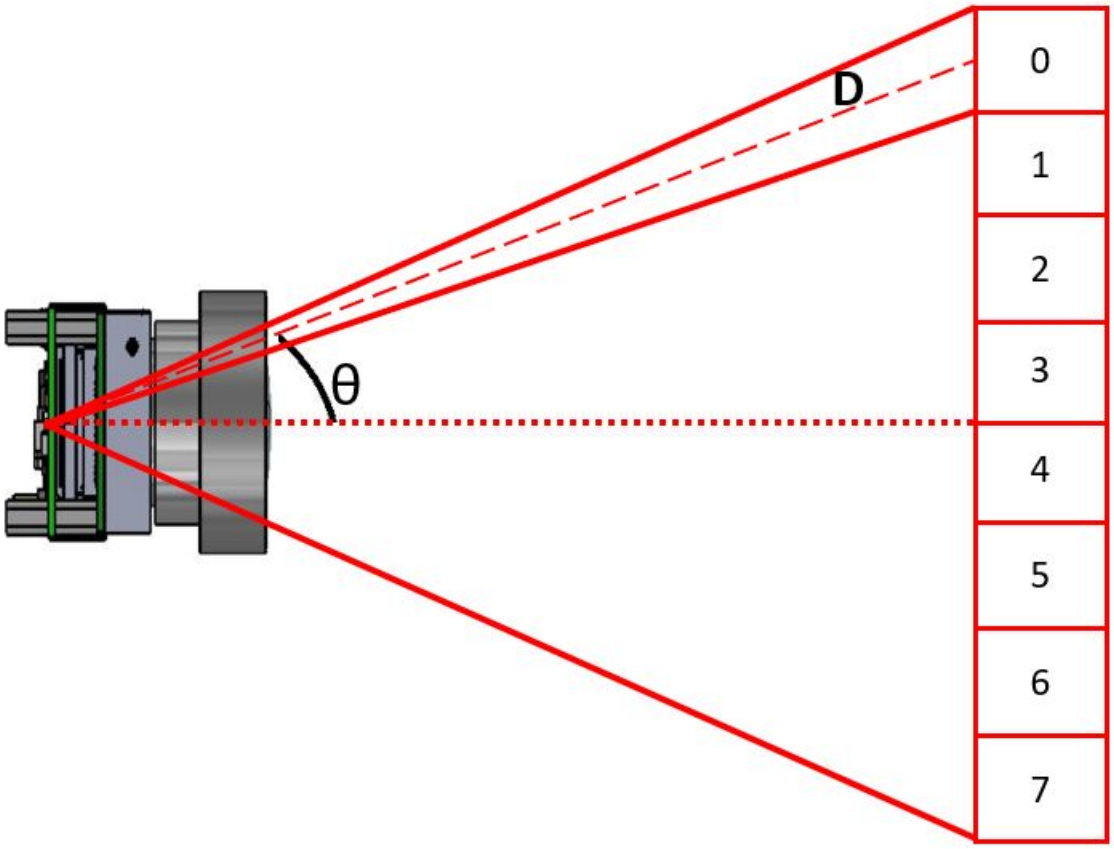

Figure 4.4: Top View of the Leddar Vu8 FoV Segments and How the Sensor Returns the Distance of an Obstacle Found in a Segment (e.g., Segment 0).

Table 4.1: Default CAN IDs for Request Messages (R_x) to and Response Messages (T_x) From the Leddar Vu8.

Sensor	Message ID (Hex)	Message ID (Decimal)	Direction	Data Type
Leddar Vu8 Long Range (S1)	0x740	1856	Rx	Request from a host
	0x750	1872	Tx	Answer to a host request
	0x751	1873	Tx	Number of detection messages
	0x752 & over	1874 and above	Tx	Detection messages
Leddar Vu8 Short Range (S2)	0x780	1920	Rx	Request from a host
	0x790	1936	Tx	Answer to a host request
	0x791	1937	Tx	Number of detection messages
	0x792 & over	1938 and above	Tx	Detection messages

For the long-range Leddar Vu8 sensor (S1 in Table 4.1), the CAN bus interface uses two default message IDs: 1856 (0x740) and 1872 (0x750) [54] for request message from the Leddar Vu8 and response message from the host, respectively. These message IDs can be modified using the Leddar Configurator (Appendix C). If multiple Leddar Vu8 sensors are connected to a single host computer, over the same CAN bus, the message IDs for both sensors must be configured to a different value for each sensor. This prevents the CAN bus data of a sensor from being overwritten by another sensor and also helps in distinguishing the sensor data. Table 4.1 shows the IDs of request and response messages used for both the long-range and short-range Leddar Vu8 sensors. For the ease of implementation, default CAN IDs were used for the long-range Leddar Vu8 sensor, and message IDs for the short-range Leddar Vu8 sensor were configured

using the Leddar Configurator. Setting unique message IDs to each sensor also helped to implement CAN filters based on message IDs. These filters help in extracting the required messages based on their IDs. The message frame follows the little-endian format where the message interpretation must be done from Byte 7 to Byte 0. For convenience the bytes of each message will be numbered from 0, i.e., Byte 0 is first and least significant byte, and Byte 7 is the eighth and most significant byte of a message. A request message from the host to the sensor and a stream of response messages from the sensor to the host are shown below. This data was logged while the sensor was pointed towards a retro reflective board located 14m from the sensor.

Request from the host to the sensor:

0x740 : 02, 01, 00, 00, 00, 00, 00, 00

Response message stream from the sensor to the host:

0x750 : 02, 01, 00, 00, 00, 00, 00, 00

0x751 : 08, 64, 00, 00, 41, 8D, 26, 00

0x752 : 79, 05, 8F, 00, 01, 00, 07, 00 : detection of channel #7

0x753 : 77, 05, D3, 00, 01, 00, 06, 00 : detection of channel #6

0x754 : 77, 05, EC, 00, 01, 00, 05, 00 : detection of channel #5

0x755 : 76, 05, F9, 00, 01, 00, 04, 00 : detection of channel #4

0x756 : 77, 05, F2, 00, 01, 00, 03, 00 : detection of channel #3

0x757 : 77, 05, ED, 00, 01, 00, 02, 00 : detection of channel #2

0x758 : 77, 05, B9, 00, 01, 00, 01, 00 : detection of channel #1

0x759 : 78, 05, 9F, 00, 01, 00, 00, 00 : detection of channel #0

More details about the request and response message are provided below.

Request message from host to the sensor

A request message is sent from the host to the sensor to initiate or terminate response

messages from the sensor. 0x740 (in hexadecimal) or 1856 (in decimal) is the message ID. The first byte (Byte 0) describes the main function and the rest of the message bytes are used as arguments. The bytes will be described using the aforementioned example.

Byte 0: the value is 0x02. In this case, the request type is to send the detection once, meaning one detection frame (data from all 8 segments) will be sent by the sensor to the host. Byte 0 values used in our application are 0x01 (stop sending detection to host), 0x02 (send one frame of detection to host) and 0x03(send detection continuously, until host requests to stop). For other byte values and corresponding function refer to Appendix B.

Byte 1: the value is 0x01. This is to request the response in a multiple-messages format, meaning the detection in each channel of the sensor will be sent by a message with a unique message ID (e.g., 0x759, 0x758 or 0x757). If the value of Byte 1 is set to 0x00, then all the segment data messages from the sensor will have the same address (i.e., 0x753, 0x754 ... 0x758, 0x759 will be replaced by 0x752).

Byte 2 to Byte 7: the value is 0x00. These bytes do not take any other value in the request message.

Response message stream from the sensor to the host

The response message from the sensor to the host is a message stream with response to host's request followed by detection messages. The first message with message ID 0x750 (1872) is the response to the host's request (0x740). If the request was successful, the response message (0x750) is an echo of the request message (0x740).

The next message with message ID 0x751 (1873) specifies the number of detections, LED power in percentage and the time stamp. Values from the aforementioned example will be used for illustration.

Byte 0: indicates the number of detections. In the considered example, it is 08 which means there are 8 detections. This number represents the total of number of objects

detected per segment in that frame.

Byte 1: indicates the LED power in percentage. In this example, 64 in decimal converts to 100%, meaning the LED transmit power is set to 100%.

Byte 2 and 3: These are reserved bytes with value 0x00 (hex).

Byte 7- Byte 4: indicate a time stamp. The time stamp is expressed as the number of milliseconds since the module was started. In the above example, 0x00268D41 in decimal converts to 2,526,529ms.

Messages after the sensor message with ID 0x751, i.e., messages with ID 0x752 (1874) and above, correspond to the detections of the sensor. Each 8 byte message corresponds to a detection in a particular segment. Each of these messages have the information arranged as follows:

Byte 7 and 6: the segment in which the sensor has detection.

Byte 5 and 4: is the flag information. Flag 0x01 means the measurements are valid. Flag 0x09 means the received signal is above saturation level. The measurements are valid but have lower accuracy and precision.

Byte 3 and 2: indicate the amplitude of light received by the sensor from obstacle.

Byte 1 and 0: indicate the distance of the obstacle from the sensor.

For the message with ID 0x752 the values are shown in Table 4.2.

The rest of the messages can be decoded similar to the message with ID 0x752.

From the obtained sensor data, the following computations can be done (Figure 4.4):

- Lateral range L_1: D*sin(θ)
- Longitudinal range L_2: D*cos(θ)

Table 4.2: Hex Values Obtained From One CAN Response Message Frame 0x752 and Its Corresponding Byte Location and Decimal Values.

Byte Location	Byte Value (Hex)	Byte Value (Decimal)	Description
Byte 7 and 6	00 07	7	Seventh segment of the sensor's horizontal FoV
Byte 5 and 4	00 01	1	Valid detection
Byte 3 and Byte 2	00 8F	143	Amplitude of light from obstacle
Byte 1 and Byte 0	05 77	1399	Distance in centimeters.

- Range Rate: $(L_{2,f2} - L_{2,f1})$*f, where $L_{2,f2}$ is the longitudinal range in frame 2, $L_{2,f1}$ longitudinal range in frame 1, f is the frame rate, and frame 1 and frame 2 are two consecutive frames.

The azimuth angle and lane position (left, center or right) of the obstacle depends on factors such as location of the sensor and the segment in which the obstacle is located. For this reason, the sensor is placed in the center of the ego car. The lane position of the vehicle is determined by the sensor FoV and the longitudinal distance (L_2). The long range Leddar Vu8 with 20^0 is used to identify farther obstacles. Figures 4.5 and 4.6 illustrate how the lane position of an obstacle vehicle is determined using the Leddar Vu8 sensor with 100^0 and 20^0 FoV, respectively. The segments shown in Figures 4.5 and 4.6 are based on trigonometric computation. Tables 4.3 and 4.4 list how the lane position of a vehicle is determined based on the segment and longitudinal distance (L_2) detected by the sensor. If the obstacle is in segment 0, and is detected by Vu8 with 100^0 horizontal FoV, then the azimuth angle of that obstacle is 43.75^0 to the left of the ego vehicle.

The longitudinal distance (L_2) computed for all segments, in a frame, is used to determine if an object has occupied multiple segments. When the detection frame is obtained (example shown above as response message stream from the sensor to the host), distances in all the segments are decoded. The adjacent segments are grouped according to Tables 4.3 and 4.4. For instance, if an object is detected in segments 3 and 4 of the long range Vu8 sensor, and is at 40m longitudinal distance (L_2 obtained from segments 3 and 4) from the ego vehicle, the object is classified as single object which has occupied host lane and is at 40m longitudinal distance (L_2) from ego vehicle. In such cases the azimuth angle of the detected object with respect to ego vehicle, is half of the difference between the azimuth angle of the left segment and the azimuth angle of the right segment. For the above example, the azimuth angle will be $(2.5^0 - 2.5^0)/2 = 0^0$.

The short range Vu8 sensor is majorly used to identify objects at closer range, in the left and right lane, as shown in Table 4.3. The long range Vu8 sensor is used to detect objects in the host lane as well as objects in the left and right lane above 20m longitudinal distance (L_2), as shown in Table 4.4.

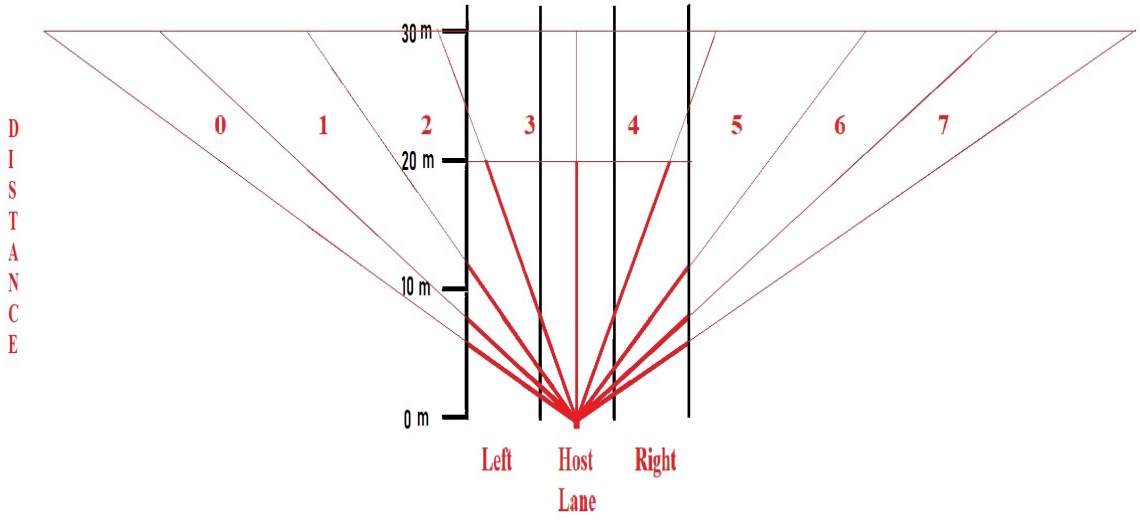

Figure 4.5: FoV of Leddar Vu8 100^0 Sensor.

Table 4.3: Data Utilization From Leddar Vu8 100^0 Sensor.

Longitudinal distance (L_2) (m)	Left Lane	Host Lane	Right Lane
0-10	Segments 0 and 1	Segments 3 and 4	Segments 6 and 7
10-20	Segment 2	-	Segment 5

Table 4.4: Data Utilization From Leddar Vu8 20^0 Sensor.

Longitudinal distance (L_2) (m)	Left Lane	Host Lane	Right Lane
0-10	-	Segments 3 and 4	-
10-20	-	Segments 3 and 4	-
20-30	Segments 0 and 1	Segments 3 and 4	Segments 6 and 7
30-40	Segments 0, 1 and 2	Segments 3 and 4	Segments 5, 6 and 7
40-50	Segments 1 and 2	Segments 3 and 4	Segments 5 and 6
50-60	Segment 2	Segments 3 and 4	Segment 5
60-70	-	Segments 3 and 4	-

4.3 Data Transmission From Treerunner to Vehicle CAN

The computed target distance both lateral (L_1) and longitudinal (L_2), lane position and azimuth angle are transmitted to the vehicle CAN bus from the Treerunner through CAN. CAN messages with ID 0x441, 0x442 and 0x443 are generated and transmitted through CAN channel 2 of the Treerunner (CAN channel 1 is used to communicate with the Leddar Vu8 sensors), to the ETAS module, which is the hybrid vehicle controller that controls the driver feedback indicators (Section 3.3.2). Message IDs 0x441, 0x442 and 0x443 represent information of the targets present in left, host and right lanes as determined by the Leddar Vu8 sensors, respectively.

The message is of length 8 bytes and the data organization is shown in Figure 4.7

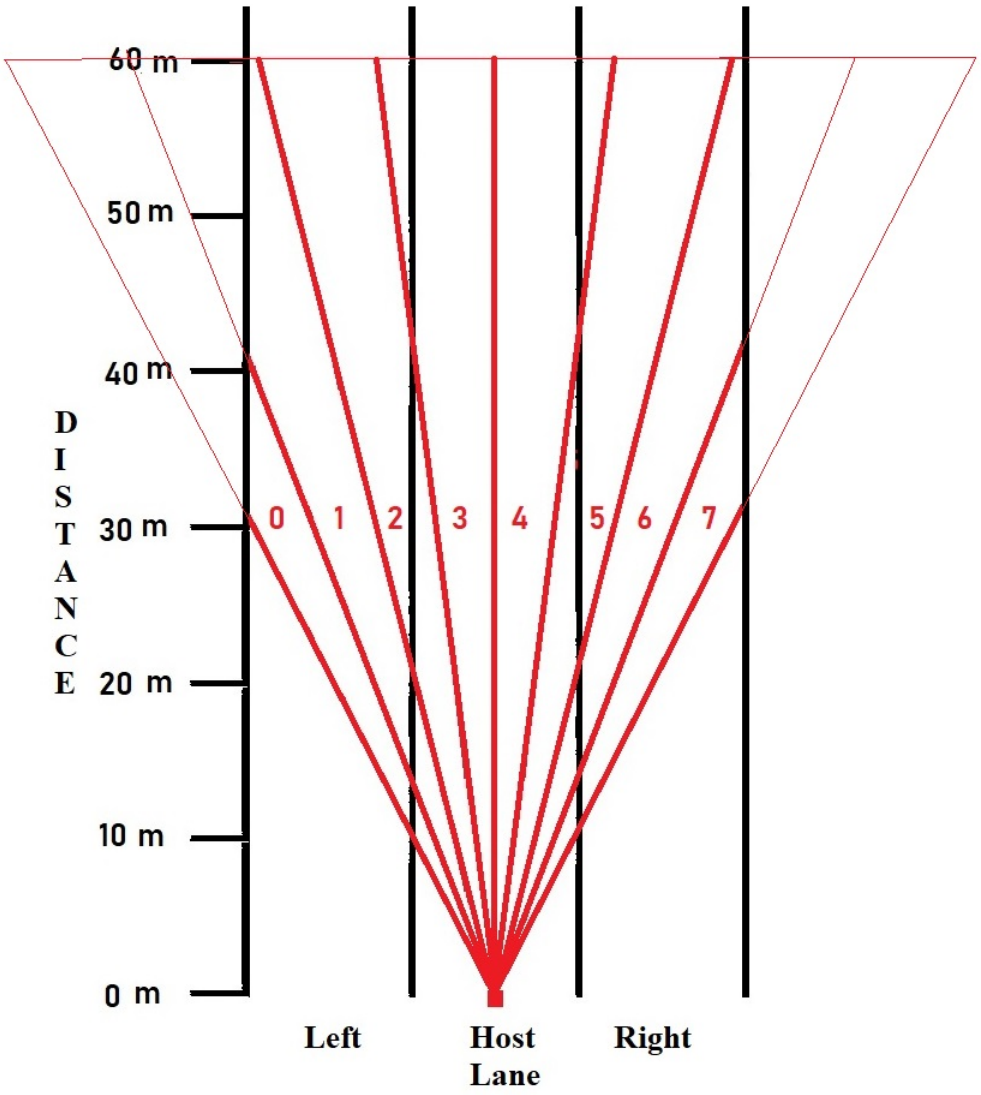

Figure 4.6: FoV of Leddar Vu8 20^0 Sensor.

In Figure 4.7, each color code represents the length in bits of each data value and also shows the location of corresponding Most Significant Bit (MSB) and Least Significant Bit (LSB). For instance, the Lateral Range data value occupies 10 bits of the message and its LSB is at bit 30 of the 64 bit message while its MSB is at bit 23 of the message. This message also follows the little endian byte order.

Bit	0	1	2	3	4	5	6	7
Byte								
0	Range Rate (MSB) 7	6	5	4	3	2	1	0
1	15	14	13	(LSB) 12	Lane Position (MSB) 11	(LSB) 10		
2	Lateral Range (MSB) 23							
3		(LSB) 30						
4	Longitudinal Range (MSB) 39							
5				(LSB) 44	Azimuth Angle (MSB) 43			
6						(LSB) 50		
7								

Figure 4.7: CAN Message Format Used to Transmit Data From Treerunner to Vehicle CAN.

Chapter 5

IMPLEMENTATION AND RESULTS

This chapter presents details about the initial prototyping of the object detection system using the Aggregated Channel Feature algorithm [50] [55], sensor integration, and the software development and test conducted to analyze the performance of the developed system. Section 5.1 describes the implementation of the object detection algorithm using MATLAB. Section 5.2 presents details about the enclosure design and integration of hardware. Section 5.3 describes the implementation of the overall algorithm in Treerunner, as well as the testing strategy and achieved output.

5.1 Initial Prototyping and Sensor Testing

In this section, details about the initial implementation of the object detection algorithm and about the initial testing of the Leddar Vu8 sensor are discussed. The initial testing helped in configuring many algorithm-related parameters in order to improve the performance and accuracy of the overall system.

5.1.1 Object Detection Prototyping

As a proof of concept, the Aggregated Channel Feature detector was tested using MATLAB and the Piotr's Computer Vision toolbox [55]. OV10635 camera was mounted under the rear view mirror of a test car (âĂŸ96 Honda Accord) and was connected to the NXP S32V234 (Treerunner) board, as shown in Figure 5.1. Using this OV10635 and Treerunner setup, a dynamic test video was recorded such that, in each video frame, there was at least one target vehicle being followed. Figures 5.2 and 5.3 show sample frames from the recorded test video.

Figure 5.1: Test Setup to Capture Video Data.

Attributes of the recorded video and number of targets in each frame are shown in Table 5.1. The number of video frames in total is 900, corresponding to a duration of 30 seconds, where 353 frames have two targets in each frame and 547 frames have three targets in each frame. This gives a total of 2347 targets in the recorded 30-second video.

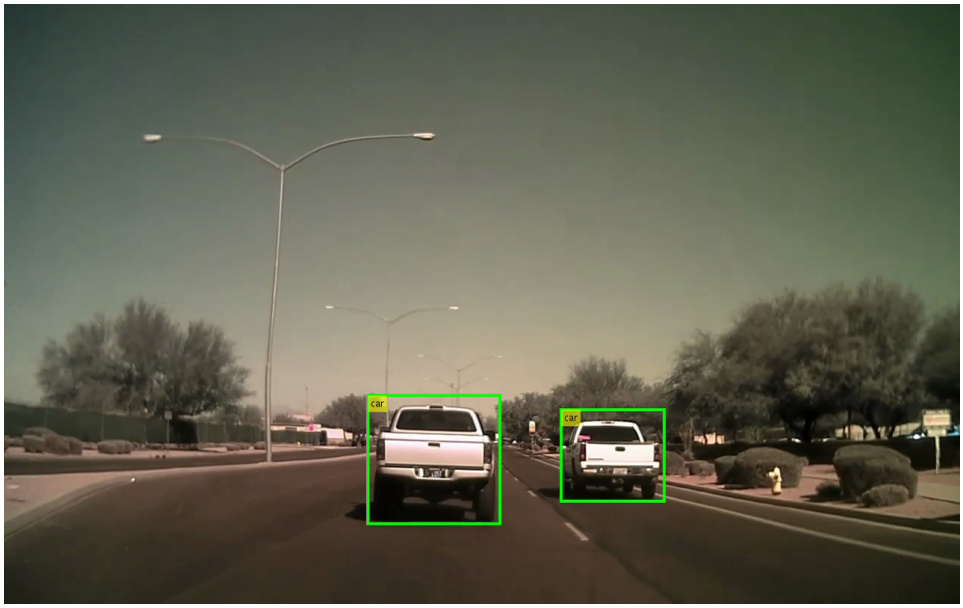

Figure 5.2: Ground-Truth Annotated Frame From the Recorded Test Video. This Frame Has Two Targets Ahead.

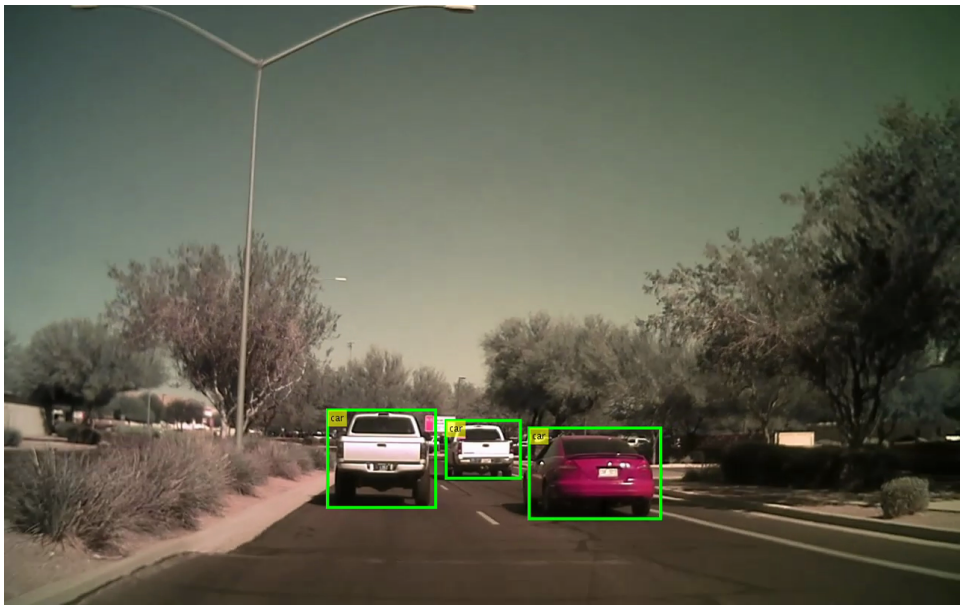

Figure 5.3: Ground-Truth Annotated Frame From the Recorded Test Video. This Frame Has Three Targets Ahead.

The recorded video was used to label the ground-truth using MATLAB's Ground Truth Labeler as explained in Section 2.5, and a video with all the labeled ground-truth information was stored. Here, the ground-truth information represents the Region of Interest in the image, that is occupied by the target to be detected, vehicles in this case. For instance, if there are two vehicles present in an image of the video sequence, the ground-truth information is the manually plotted bounding box that encloses the region of the image containing the vehicle that needs to be detected.

Table 5.1: Description of the Data Collected Using OV10635 and Treerunner.

Attributes	Values
Ego vehicle used	Honda Accord 1996
Duration	30 seconds
Time of capture	3 pm to 4 pm
Weather condition	Clear sunny
Total Frames	900
Total number of targets	2347
Minimum number of targets in a frame	2
Number of frames with 2 targets	353
Number of frames with 3 targets	547

Figure 5.4: Target Vehicles With Bounding Boxes Detected by the Trained Detector.

A MATLAB script was written to apply the ACF detector, whose implementation is provided by the Piotr's Computer Vision Toolbox, to each frame of the video and to store the resulting bounding box with corresponding confidence score for each detection. Figure 5.4 shows how the bounding box is plotted around the corresponding target. The bounding box for "car 2" is bigger than the actual vehicle. This is because the algorithm combines multiple overlapping bounding boxes to get one output box for a particular object.

Table 5.2: ACF Object Detection Performance Metrics Computed on the Video Data Recorded Using the OV10635 and Treerunner Setup.

Metrics	Detector	Comment
True Positive (TP)	1564 of 2347	1564 targets were detected successfully out of the 2347 total targets
False Positive (FP)	247	False detections that do not correspond to true targets.
True Negative (TN)	N/A	Not applicable as non-vehicle objects are not labelled
False Negative (FN)	783 of 2347	783 targets were not detected out of the 2347 total targets
Precision	86.3611%	TP/(TP+FP)
True Positive Rate (Recall)	66.6383 %	TP/(TP+FN)
False Negative Rate	35.1%	FN/(TP+FN)
False Discovery Rate	13.5%	FP/(TP+FP)
True Negative Rate	N/A	Can be computed only when True Negative is available

To evaluate the performance of ACF for object detection, each detected bounding box was compared with the ground-truth bounding boxes for the recorded video, to compute the IoU (as explained in Section 2.6) with a 50% threshold. The resulting classification statistics are shown in Table 5.2. The True Positives (TP) and False Negative (FN) add up to the total number of ground-truth targets in the dataset.

5.1.2 Initial Testing of the Leddar Vu8

The Leddar Vu8 sensor was initially tested separately using a target retro reflective board as demonstrated in Figure 5.5. This was done to configure the sensor parameters (Appendix A) using the Leddar Configurator (Appendix C), to fit the system requirements (Section 3.1). To test the maximum distance up to which the sensor detected an obstacle, the sensor was kept stationary and the target was moved away from the sensor along a straight line. To test the functionality of various segments of the sensors' FoV, the target was moved along the horizontal FoV of the sensor. This helped identify the right values for sensor parameters such as Accumulation, Oversampling, and Points of the Vu8.

Figure 5.5: Leddar Vu8 Test Setup. Retro Reflective Board Used as Target to Test the Range of the Leddar Vu8 Sensor.

Accumulation or signal accumulation is the process of accumulating the reflected light beam to increase the signal intensity.

Oversampling is the process of sampling a signal above the Nyquist rate, to improve resolution and reduce reconstruction error.

Points is the number of base sample points. Points determine the maximum detection range. Higher points improves the detection range of the sensor.

5.2 Hardware Integration

Integrating the ADAS hardware involved three major steps, which were selecting mounting locations, enclosure design and electrical integration.

5.2.1 Selecting Hardware Location

Selecting the location for mounting the hardware, involved several factors such as the physical properties and capabilities of the hardware. Locations for sensors were selected to maximize the sensing performance.

The OV10635 OmniVision camera, whose video output was used as input to the vehicle detection algorithm, was mounted behind the rear view mirror as shown in Figure 5.6. Since the camera was mounted inside the vehicle, the enclosure that was designed to protect the camera did not need to be designed to provide Ingress Protection (IP) [56].

Two Leddar Vu8 sensors were mounted in the front grille as shown in Figure 5.7. This helped avoid any physical obstruction for the LED beam propagation and provided effective use of the sensors' field of view.

The NXP S32V234 (Treerunner board) was mounted in the trunk of the car in a custom designed case. The design included slots on the sides that allowed proper connec-

Figure 5.6: OV10635 Mounted on the Test Vehicle.

tion of wires from sensors and power source to corresponding ports. Figure 5.8 shows the Computer Aided Design of the NXP S32V234 case.

5.2.2 Enclosure Design

The enclosures for the NXP S32V234 board, OmniVision camera and Vu8 sensors were custom made by the mechanical engineers, who were involved in the EcoCAR3 project, and the designs were completed using Computer Aided Design (CAD). An initial version of the enclosure was 3D printed for each hardware, so that modifications could be made to the design to converge to a final design. After finalizing all the modifications on the enclosure, such as slots for cable pass through and mounting features, the final enclosure was 3D printed. The structural properties of the enclosures are presented in Table 5.3, while CAD renderings of the final design for the S32 board enclosure, camera housing and Vu8 sensor enclosure are shown in Figure 5.8, Figure 5.9 and Figure 5.10, respectively. The enclosure for Treerunner holds the de-serializer board (Section 3.3) in it. The side walls of the enclosure can be modified based on the require-

Figure 5.7: Leddar Vu8 Mounted on the Test Vehicle.

Table 5.3: Detailed Characteristics of the Custom Designed Hardware Enclosures. L, B and H Stands for Length, Breadth and Height, Respectively.

Hardware	Housing Dimension(inch)	Housing Material
NXP S32V234	L- 10.5; B- 10.5 H- 1.5	Acrylic for the face of the board PLA Plastic for the sides
Omnivision OV10635	L- 2; B- 2; H- 2	PLA Plastic
Leddar Vu8	L- 2.9; B- 1.6; H- 2.5 (Vu8 100^0) L- 2.8; B- 1.4; H- 2.7 (Vu8 20^0)	PLA Plastic

ment of ports to be accessed in the S32V234 board. The camera enclosure is designed to enclose all the sides of the camera leaving the lens unobstructed and the design further accounts for the camera connector.

The enclosure design for the Vu8 sensors was critical as the horizontal field of view played a major role in the design. To avoid obstruction in the transmit path of light from

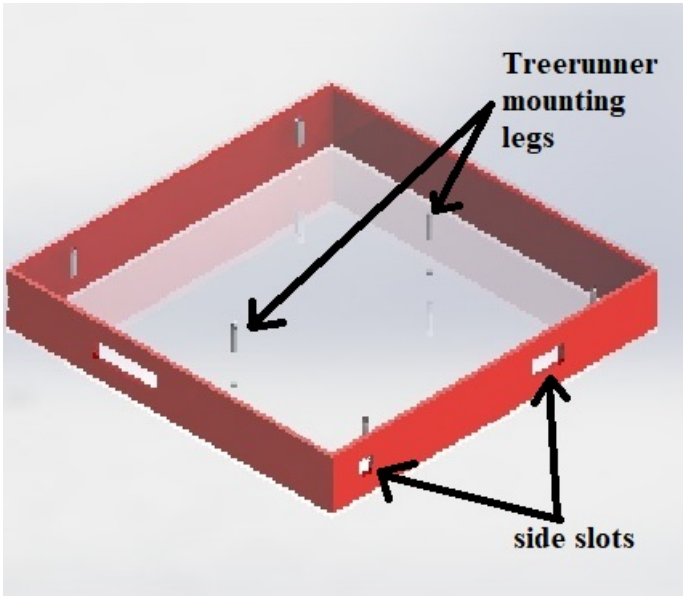

Figure 5.8: Treerunner's Enclosure.

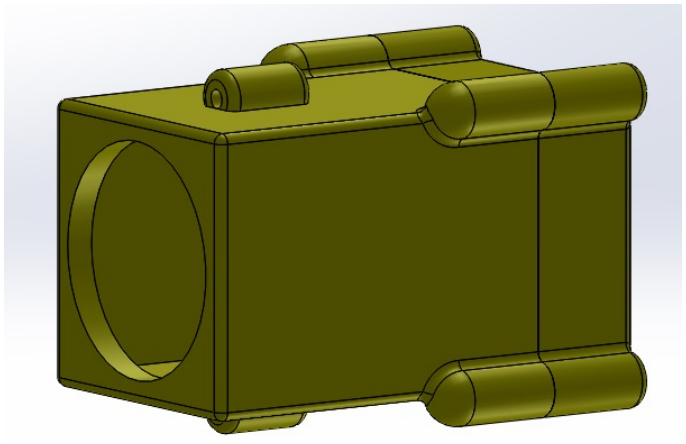

Figure 5.9: OV10635 Camera's Enclosure.

Figure 5.10: Leddar Vu8 Sensor's Enclosure.

sensor, the enclosure was designed to closely fit the sensor as shown in Figure 5.10 with a clear plastic cover on the front side, to protect the sensor from external contact.

5.2.3 Electrical Integration

All the sensors and Treerunner were connected to the low voltage battery of the vehicle through proper voltage regulation circuitry. This was to ensure all the components were supplied with the required voltage and no component gets damaged due to high voltage spikes.

5.3 Software Development and Testing

Software was developed in modules and were integrated together to obtain the final package. The first module (Algorithm 1) was the implementation of the Aggregated Channel Feature framework for the object detection task, and the second module (Algorithm 2) implemented the Leddar Vu8 data acquisition and processing using CAN.

Algorithm 1: ACF-based detection pseudo-code.

1 Input image (frame size 1280x800) is transformed from YUV (YCbCr) format to RGB format and resized to half the size of the original image.;

2 Resized image is converted from 24 bits per pixel to 32 bits per pixel, by adding an alpha channel [57] and is converted to LUV format.;

3 The channel pyramid (as explained in Section 4.1) of the LUV image is computed. The detector is applied to the computed feature pyramids, to obtain bounding boxes and associated confidence score.;

4 Obtained bounding boxes are plotted on the input image and displayed.;

Algorithm 2: Data Acquisition from the Leddar Vu8 sensor and Processing.

1 Open raw CAN socket and declare filters (explained below) based on message ID to receive messages from one sensor at a time.

2 Request message (message IDs shown in Figure 5.11) is sent to the sensors whenever a measurement is needed and the response messages (described in Section 4.2) from the sensors are read from Channel 1. To read one sensor at a time, filters (described in previous step) are set. The messages from each sensor are acknowledgement to the request message from host (message ID 0x750 for long range and 0x790 for short range Vu8), number of detection (message ID 0x751 for long range and 0x791 for short range Vu8) and detection messages (message ID 0x752 for long range and 0x792 for short range Vu8).

3 Response messages from each sensor are stored separately and converted from received hexadecimal values to decimal values (described in Section 4.2) to calculate lateral, longitudinal distance and azimuth angle of detected target.

4 Data from different segments of the Vu8 sensors are combined according to Tables 4.3 and 4.4 to obtain target lane position.

5 CAN message frame with transmit message IDs 0x441, 0x442 and 0x443, shown in Figure 5.12 is generated using the processed data and is transmitted from the NXP S32V234 to the ETAS ES910 (hybrid controller) for the distance estimation to reach vehicle CAN bus. Message IDs 0x441, 0x442 and 0x443 represents target information corresponding to left, host and right lane, respectively.

More details about the filters (Step 1 of Algorithm 2) are provided below.

For the short range Leddar Vu8, the ACK_ID_SHORT is 0x790, $RESP\_ID\_SHORT$ is 0x791, and DET_ID_SHORT is 0x792. The detection data is requested in a single message format. In this way, all the detection messages will have the message ID 0x792. For this case, the filters are declared as follows:

$rfilter1[0].can\_id = ACK\_ID\_SHORT$;

$rfilter1[0].can\_mask = CAN\_SFF\_MASK$;

$rfilter1[1].can\_id = RESP\_ID\_SHORT$;

$rfilter1[1].can\_mask = CAN\_SFF\_MASK$;

$rfilter1[2].can\_id = DET\_ID\_SHORT$;

$rfilter1[2].can\_mask = CAN\_SFF\_MASK$;

For the long range Leddar Vu8, the ACK_ID_LONG is 0x750, $RESP\_ID\_LONG$ is 0x751, and DET_ID_LONG is 0x752. The detection data is requested in a single message format. In this way, all the detection messages will have the message ID 0x752. For this case, the filters are declared as follows:

$rfilter2[0].can\_id = ACK\_ID\_LONG$;

$rfilter2[0].can\_mask = CAN\_SFF\_MASK$;

$rfilter2[1].can\_id = RESP\_ID\_LONG$;

$rfilter2[1].can\_mask = CAN\_SFF\_MASK$;

$rfilter2[2].can\_id = DET\_ID\_LONG$;

$rfilter2[2].can\_mask = CAN\_SFF\_MASK$;

To improve the execution speed and accuracy, a region of interest was specified in the input frame. This region of interest was obtained by cropping out the top 200 rows of the image to ignore the sky region in the image. This avoided feature computation in areas where vehicles could not be found. Multiple bounding boxes for a single target were grouped to form a single bounding box using an OpenCV function *groupRect-*

angles()[58] with an overlap threshold of 80%. This function takes as input rectangles that need to be grouped together, a group threshold representing the minimum possible number of rectangles minus 1 (if a cluster consists of rectangles less than the group threshold, that cluster is rejected), an overlap threshold that is used to group two or more rectangles if these overlap by more than the specified overlap threshold (set to 80% in this implementation).

To communicate with the Leddar Vu8 sensors, both sensors were connected to the Treerunner board through CAN Channel 1, as shown in Figure 5.11, and CAN Channel 1 was configured to a baud rate of 115200 bits per second. To transmit the data to the vehicle bus, the Treerunner board was connected to the ETAS module through CAN Channel 2, as shown in Figure 5.12, and CAN Channel 2 was configured to a baud rate of 115200 bits per second.

Since both the Leddar Vu8 sensors were connected to the Treerunner board using the the same CAN bus, rfilter1 and rfilter2 (see Algorithm 2) are filters that were implemented to filter CAN messages based on CAN ID and to differentiate data from short range and long range Vu8 sensors. More details about how the filters work can be found in Section 2.7.

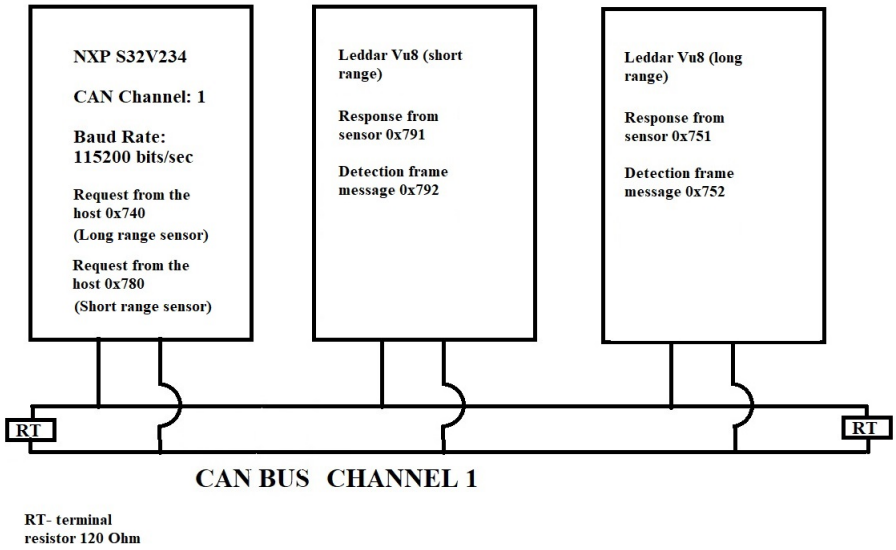

Figure 5.11: Leddar Vu8 Sensors Connected to Treerunner.

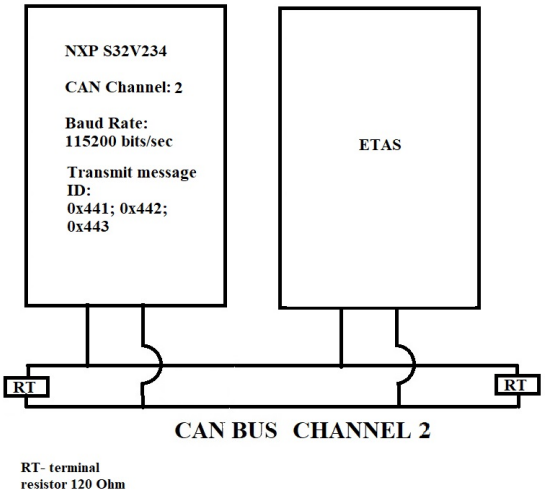

Figure 5.12: Treerunner Connected to ETAS to Transmit the Detection Data to the Vehicle CAN Bus.

To check the data acquisition and processing from the Leddar Vu8 sensors using CAN, a test procedure was performed as described in Section 5.1.2. The retro reflective board was used as a target. A request message from Treerunner was transmitted to the Vu8 sensors and the response was read. The response from the sensor was processed

to obtain the distance to the retro reflective target that was detected by the sensor, and the obtained distance was compared with the actual distance.

Each time an input image is captured, a request message is sent from Treerunner to Vu8 through CAN and the response is read. Both the image frame and sensor data are processed in parallel. The camera and Vu8 sensors were mounted along the center axis of the vehicle. This was useful to calibrate the camera and get a correspondence of the horizontal field of view of the camera and the FoVs of the Vu8 sensors.

To fuse the processed data including the Vu8 sensor data together with the vehicle detection results using the camera, the horizontal FOV of the camera was split uniformly into three bins, each representing left, host and right lanes. The coordinates of each detected vehicle's bounding box were utilized to identify the lane in which the considered bounding box was located, and the processed sensor data corresponding to those lanes was mapped to the detection. A sample annotated frame is shown in Figure 5.13, in which the target is present in the host lane and the target's distance information is displayed near the top of the image frame.

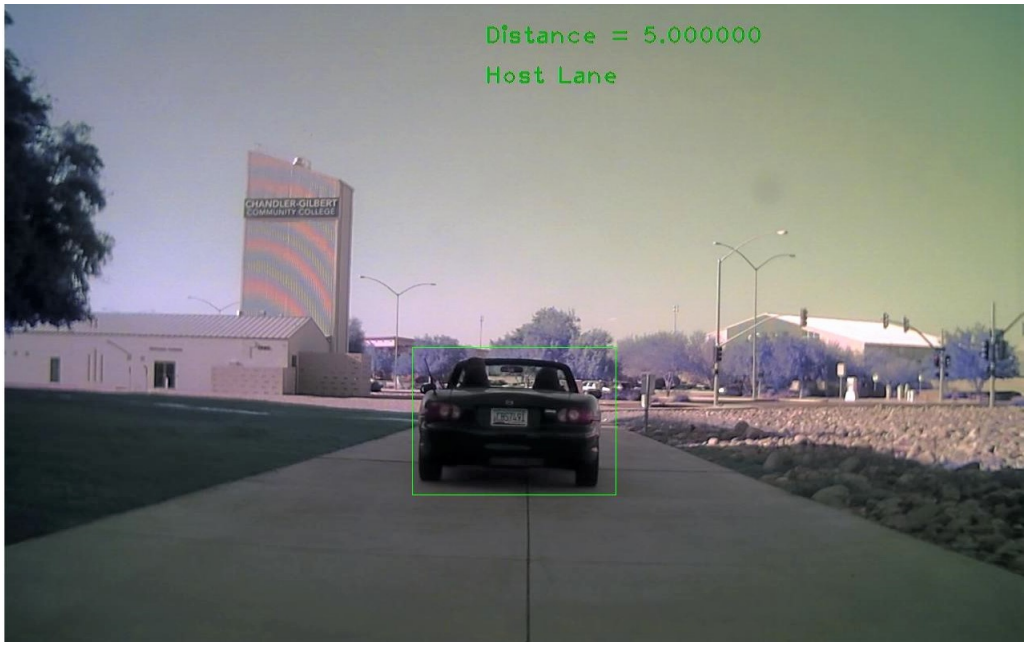

Figure 5.13: Sample Annotated Frame Showing Bounding Box Obtained From Camera Data and Distance Estimation in Meters, Computed From Leddar Vu8 Sensor Data.

The ETAS ES910, which receives the CAN message from the NXP S32V234 board, processes the data to identify the distance between the ego vehicle and the target in the host lane. This is done to generate a pulsated signal that controls the driver feedback LED and buzzer. To simplify the design, only the target present in the host lane is considered to avoid potential rear end collision. The blinking frequency of the LED varies based on the distance between the target and ego vehicle. The LED starts blinking slowly with a farther distance and the blinking frequency is increased as the distance decreases. The same frequency pattern is applied to the buzzer. Figure 5.14 shows the prototype version of the LED indicator used as feedback to the driver.

Figure 5.14: Driver Feedback LED Prototype. In This Setup, a Single LED Is Used to Test the Functionality.

Chapter 6

CONCLUSION

This thesis implements a driver assistance system using real-time sensor data acquisition and processing. This chapter summarizes the contributions of this thesis and proposes several directions for future research.

6.1 Contribution

In this thesis, a driver assistance system was developed that detects vehicles and estimates the distance between detected vehicles and ego vehicle to provide feedback to the driver. The contributions of the thesis can be summarized as follows:

- Hardware selection for range estimation, by analyzing the hardware capabilities, logistics and, mechanical and interface compatibility.
- Hardware integration by strategically identifying suitable locations and working with mechanical engineers to develop enclosure for the proper fixture.
- Implementation of data acquisition and processing from range estimation sensor through Controller Area Network, to estimate the lateral and longitudinal distance between the target and ego vehicle.
- Adaptation of Aggregated Channel Feature object detection, to detect vehicles.
- Implementation of a front collision system based on range measurement data obtained from range estimation sensors, irrespective of the type of obstacle.

6.2 Future Research Direction

Possible enhancements and future directions for the proposed framework are as follows:

- Implementation of blind-spot detection and side distance warning will enhance the safety of the vehicle. This implementation can be achieved using short-range ultrasonic sensors or short-range RADAR. Along with blind-spot detection, the implementation of a camera-based side view on the right side will enhance safety while performing a right turn, as the right-side view is maximally obstructed in left-hand drive.

- Use of data processor with higher computation capacity for faster data processing to ramp up the processing rate up to 40 frames per second.

- Establishing effective communication between ADAS and vehicle to ensure the ADAS is aware of details such as acceleration, current velocity, steering angle, fuel consumption, and other factors that could affect the performance of ADAS.

- Implementation of haptic feedback such as steering vibration, along with visual and audio feedback, can be provided. This implementation ensures to capture the attention of the driver even if he or she misses the other types of feedback.

- Implementation of an ADAS based electronic braking system to improve the fuel efficiency of the vehicle. This could be automated based on the ADAS detections, or it could be controlled by the user, based on the ADAS feedback to the driver. In a user-controlled scenario, a button on the steering wheel, or an extra shaft next to the hand brake lever, can be used by the driver to brake the vehicle based on the ADAS feedback. This would improve not only the safety but also the fuel efficiency and power regeneration in hybrid vehicles.

REFERENCES

[1] Hedges & Company. US vehicle registration statistics. https://hedgescompany.com/automotive-market-research-statistics/auto-mailing-lists-and-marketing/, 2018. Online; accessed: 30-November-2018.

[2] National Highway Traffic Safety Administration.(NHTSA). 2017 fatal motor vehicle crashes: Overview. https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812603, 2018. Online; accessed: 30-November-2018.

[3] National Highway Traffic Safety Administration.(NHTSA). What Is Distracted Driving?. https://www.nhtsa.gov/risky-driving/distracted-driving., 2018. Online; accessed: 30-November-2018.

[4] Advanced Vehicle Technology Competition. EcoCAR 3. https://avtcseries.org/competitions/ecocar3-2/, 2019. [Online; accessed 08-September-2019].

[5] Wikipedia contributors. Collision avoidance system — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Collision_avoidance_system&oldid=850164806, 2018. Online; accessed: 30-November-2018.

[6] Karim Nice. How Anti-Lock Brakes Work. https://auto.howstuffworks.com/auto-parts/brakes/brake-types/anti-lock-brake.htm, Jan 2000. Online; accessed 30-November-2018.

[7] Ross D Olney, Robert W Schumacher, Francine H Landau, and R Wragg. Collision warning system technology. Technical report, Delco Electronics, 1995.

[8] The Editors of Publications International Ltd. Traction control explained. https://auto.howstuffworks.com/28000-traction-control-explained.htm, September 2005. Online; accessed: 30-November-2018.

[9] Karim Nice. How Cruise Control Systems Work. https://auto.howstuffworks.com/cruise-control.htm, January 2001. Online; accessed: 30-November-2018.

[10] Automoblog.net. A Brief History of The High-Tech Safety Features in your car. https://www.automoblog.net/2019/04/20/brief-history-high-tech-safety-features/, 2019. Online; accessed 05-May-2019.

[11] Vijay Gaikwad and Shashikant Lokhande. Lane departure identification for advanced driver assistance. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):910–918, 2015.

[12] Richard O Duda and Peter E Hart. Use of the Hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1): 11–15, 1972. Online: https://www.cse.unr.edu/~bebis/CS474/Handouts/HoughTransformPaper.pdf, accessed:15-Sept-2017.

[13] Yanjun Fan and Weigong Zhang. Traffic sign detection and classification for Advanced Driver Assistant Systems. In *IEEE 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pages 1335–1339, 2015.

[14] Wouter F Schmidt, Martin A Kraaijveld, Robert PW Duin, et al. Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, pages 1–1. IEEE COMPUTER SOCIETY PRESS, 1992.

[15] Nobuyuki Uchida, Takashi Tagawa, and Kenji Sato. Development of an augmented reality vehicle for driver performance evaluation. *IEEE Intelligent Transportation Systems Magazine*, 9(1):35–41, 2017.

[16] Yi Gao, Chunyu Lin, Yao Zhao, Xin Wang, Shikui Wei, and Qi Huang. 3-D Surround View for Advanced Driver Assistance Systems. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):320–328, 2018.

[17] Rudolf Kingslake. *A history of the photographic lens*, volume ISBN: 9780080508177. Elsevier, 1 edition, 1989.

[18] Wikipedia contributors. Tesla Autopilot — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Tesla_Autopilot&oldid=887162031`, 11 March 2019. [Online; accessed 30-November-2018].

[19] Wikipedia contributors. SAE International — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=SAE_International&oldid=864912717`, 20 October 2018. [Online; accessed 30-November-2018].

[20] SAE On-Road Automated Vehicle Standards Committee. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems (j3016). J3016, 2014. Online: `https://www.sae.org/standards/content/j3016_201401/`; accessed 30-November-2018.

[21] Laurent Busin, Nicolas Vandenbroucke, and Ludovic Macaire. Color spaces and image segmentation. *Advances in Imaging and Electron Physics*, 151:65–168, 12 2008. doi: 10.1016/S1076-5670(07)00402-8.

[22] International Telecommunication Union. Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. 2011. Online: `https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf`; accessed 30-November-2018.

[23] International Telecommunication Union. Parameter values for the HDTV standards for production and international programme exchange. 2015. Online: `https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.709-6-201506-I!!PDF-E.pdf`; accessed 30-November-2018.

[24] Paul Viola and Michael J Jones. Robust real-time face detection. *Springer International Journal of Computer Vision*, 57(2):137–154, 2004.

[25] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Academic Press, Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[26] Wikipedia contributors. Jaccard index — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/Jaccard_index, 14 March 2019. [Online; accessed 30-November-2018].

[27] CAN in Automation. History of CAN technology. https://www.can-cia.org/can-knowledge/can/can-history/. Online; accessed: 30-November-2018.

[28] Linux Kernel Documentation. SocketCAN. https://www.kernel.org/doc/Documentation/networking/can.txt. Online; accessed: 30-November-2018.

[29] Dave Lewis, National Semiconductor Corporation. SerDes architectures and applications. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.4371&rep=rep1&type=pdf, 2004. Online; accessed: 26-July-2019.

[30] Michael S. Brown. Understanding the In-Camera Image Processing Pipeline for Computer Vision. In *IEEE Computer Vision and Pattern Recognition - Tutorial*, 2016. https://www.eecs.yorku.ca/~mbrown/Brown_Tutorial.pdf; accessed 10-August-2019.

[31] Ron Kimmel. Demosaicing: image reconstruction from color CCD samples. *IEEE Transactions on Image Processing*, 8(9):1221 – 1228, 1999.

[32] Lanlan Chang and Yap-Peng Tan. Hybrid color filter array demosaicking for effective artifact suppression. *Journal of Electronic Imaging*, 15(1), 2006.

[33] Henrique S Malvar, Li-wei He, and Ross Cutler. High-quality linear interpolation for demosaicing of Bayer-patterned color images. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 485–488, 2004.

[34] Argonne National Laboratory. EcoCAR 3 Year 4 Event Rules Rev-D, 2018.

[35] Velodyne Inc. *HDL–64E S3 High Definition LiDAR Sensor Users Manual and Programming Guide*. Velodyne Inc., 63–HDL64ES3 REV G. edition, 2013. https://velodynelidar.com/lidar/products/manual/HDL-64E%20S3%20manual.pdf; accessed 10-September-2017.

[36] Velodyne Inc. *Velodyne LiDAR HDL-32E*. Velodyne Inc., 97-0038 REV-M. edition, 2013. http://www.mapix.com/wp-content/uploads/2018/07/97-0038_Rev-M_-HDL-32E_Datasheet_Web.pdf; accessed 10-September-2017.

[37] Velodyne Inc. *Velodyne LiDAR Puck*. Velodyne Inc., 63-9229 REV-J. edition, 2013. https://autonomoustuff.com/wp-content/uploads/2019/05/Puck__Datasheet_whitelabel.pdf; accessed 10-September-2017.

[38] Hokuyo Automatic Co. Ltd. Hokuyo UXM-30LX-EW. https://www.hokuyo-aut.jp/search/single.php?serial=171. accessed: 19-April-2018.

[39] Y Kamioka and A Yamamoto. *Scanning Laser Range Finder Smart-URG mini UST-10LX Specification.* Hokuyo Automatic Co., Ltd., 2016. accessed: 19-April-2018.

[40] Continental Automotive. Advanced Radar Sensor–ARS441. https://www.continental-automotive.com/en-gl/Passenger-Cars/Chassis-Safety/Advanced-Driver-Assistance-Systems/Radars/Long-Range-Radar/Advanced-Radar-Sensor-ARS441. Online, accessed: 19-April-2018.

[41] LeddarTech. *Leddar Sensor Evaluation Kit.* LeddarTech, . https://leddartech.com/app/uploads/dlm_uploads/2016/02/Datasheet-Leddar-Sensor-Evaluation-Kit.pdf; Online, accessed: 19-April-2018.

[42] LeddarTech. *LEDDAR ONE single-segment LIDAR sensor module.* LeddarTech, . https://leddartech.com/app/uploads/dlm_uploads/2018/04/Spec-Sheets-LeddarOne-ENG-12avril2018-web.pdf; Online, accessed: 19-April-2018.

[43] LeddarTech. *LEDDAR VU 8-segment solid-state LiDAR sensor module.* LeddarTech, . https://leddartech.com/app/uploads/dlm_uploads/2018/04/Spec-Sheets-LeddarOne-ENG-12avril2018-web.pdf; Online, accessed: 19-April-2018.

[44] Omni Vision. Ov10635 HD HDR Product Brief. https://www.ovt.com/sensors/OV10635. Accessed: 19-April-2018.

[45] Argonne National Laboratory. EcoCAR 3 Non-Year Specific Rules Rev-L, 2017.

[46] Maxim Integrated. MAX96705. https://www.maximintegrated.com/en/products/interface/high-speed-signaling/MAX96705.html. [Online; accessed 26-February-2019].

[47] everything RF. What are FAKRA connectors? https://www.everythingrf.com/community/what-is-fakra-connector. Accessed: 19-April-2018.

[48] Wikipedia contributors. SMB connector — Wikipedia, the free encyclopedia, 2018. URL https://en.wikipedia.org/w/index.php?title=SMB_connector&oldid=838931952. [Online; accessed 26-July-2019].

[49] NXP Semiconductors. S32v234: Data sheet: Technical data. https://www.nxp.com/docs/en/data-sheet/S32V234.pdf. Online; accessed 30-November-2018.

[50] Piotr Dollár, Ron Appel, Serge Belongie, and Pietro Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1532–1545, 2014.

[51] Jonas Gårding and Tony Lindeberg. Direct computation of shape cues using scale-adapted spatial derivative operators. *Springer International Journal of Computer Vision*, 17:163–191, 1996.

[52] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge J. Belongie. Integral channel features. In *British Machine Vision Conference*, pages 91.1–91.11, 2009.

[53] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian conference on computer vision*, pages 290–306. Springer, 2014.

[54] LeddarTech. *Leddar Vu8 User Manual*. Available online: `https://www.robotshop.com/media/files/pdf2/leddarvu_and_configurator_user_guide.pdf`.

[55] Piotr Dollár. Piotr's Computer Vision Matlab Toolbox (PMT). `https://github.com/pdollar/toolbox`.

[56] Wikipedia contributors. IP Code — Wikipedia, the free encyclopedia, 2019. URL `https://en.wikipedia.org/w/index.php?title=IP_Code&oldid=923727297`. [Online; accessed 4-August-2019].

[57] Thomas Porter and Tom Duff. Compositing digital images. In *ACM Siggraph Computer Graphics*, volume 18, pages 253–259, 1984.

[58] groupRectangles. `https://docs.opencv.org/3.4/d5/d54/group__objdetect.html#ga3dba897ade8aa8227edda66508e16ab9`. Accessed: 19-April-2018.

APPENDIX A

LEDDAR VU8 PARAMETERS

Table A.1: Leddar Vu8 Parameters and Corresponding Effects [43].

Parameter	Description	Effect
Accumulation	Number of accumulations	Higher values enhance the range and reduce the measurement rate and noise.
Oversampling	Number of oversampling cycles	Higher values enhance the accuracy/precision/resolution and reduce the measurement rate.
Points	Number of base sample points	Determines the maximum detection range.
Threshold Offset	Modification to the amplitude threshold	Higher values decrease the sensitivity and reduce the range.
Smoothing	Object smoothing algorithm	Smoothes the Leddar Vu8 module measurements. The behavior of the smoothing algorithm can be adjusted by a value ranging from -16 to 16. Higher values enhance the module precision but reduce the module reactivity. The smoothing algorithm can be deactivated by clearing the Enable check box. Higher positive values improve precision for slow moving object detection & lower negative values for fast moving objects.
Light Source Control	Light source power control options	Selects between manual and automatic power control. In automatic mode, light source power is adjusted according to incoming detection amplitudes. The current LED power level is visible in the Device State window.
Change Delay	Minimum delay between power changes	Smaller numbers speed up the response time of the light source power adjustment.
Object demerging	Discrimination of objects close to each other	Eases the discrimination of multiple objects in the same segment. Object demerging is only available for measurement rates under 5.0 Hz. The number of merged pulses that can be processed in each frame is also limited. A status field is available in the device state window indicating if the sensor processes all merged pulses.
Crosstalk Removal	Inter-segment interference noise removal	Crosstalk is a phenomenon inherent to all multiple segments time-of-flight sensors. It causes a degradation of the distance measurement accuracy of an object when one or more objects with significantly higher reflectivity are detected in other segments at a similar distance. This option enables an algorithm to compensate the degradation due to crosstalk but increases computational load of the Leddar Vu module microcontroller.

APPENDIX B

LEDDAR VU8 CAN MESSAGE DETAILS

Table B.1: CAN Bus Request Message Byte 0 and Byte 1 Functionality [43].

Function Request (Byte 0)	Function Request Description	Function Arguments (Byte 1)
1	Stop sending detections continuously	
2	Send detection once	Bit field of operation mode Bit-0: 0 = Return detection in single message mode 1 = Return detection in multiple message mode
3	Start sending detections continuously (that is, the module will send a new set of detections each time they are ready without waiting for a request).	Bit field of operation mode Bit-0: 0 = Return detection in single message mode 1 = Return detection in multiple message mode
4	Get input data (read only)	Table B.2
5	Get holding data	Table B.3
7	Set base address	Table B.4
8	Read module data	Table B.5

Table B.2: BYTE 1 Values for “Get Input Data” Request [43].

Input Data Type (Byte1)	Input Data Description
0	Number of segments
1	Device identification and option
2 and 3	Firmware version
4 and 5	Bootloader version
6	FPGA version
7 through 12	Serial number
13 through 18	Device name
19 through 24	Hardware part number
25 through 30	Software part number

Table B.3: BYTE 1 Values for “Get Holding Data” Request [43].

Holding Data Type (Byte 1)	Holding Data Description
0	Acquisition configuration
1	Smoothing and detection threshold
2	Light source power management
3	Distance resolution and acquisition options
4	CAN port configuration 1
5	CAN port configuration 2
6	CAN port configuration 3
7	Reserved
8	Segment enabled

In Tables B.1, B.2, B.3, the first column refers to the value of the 0^{th} byte, 1^{st} byte when 0^{th} byte value is 4, and 1^{st} byte when 0^{th} byte value is 5, respectively.

Table B.4: Setting Base Address Request [43].

Data Description	Argument	Argument Description
Base address	Bytes 4 through 7	Base address to access (from 0x00000000 to 0x00FFFFFF)

Table B.5: Reading Module Data Request [43].

Data Description	Argument	Argument Description
Read module data	Byte 1	Data length (1, 2, or 4)
Read module data	Bytes 2 and 3	Offset from 0x0000 to 0xFFFF (final address to access is the result of the base address plus this offset).

APPENDIX C

SOFTWARE TOOLS

Piotr's Computer Vision Toolbox

Piotr's Computer Vision Toolbox, implemented by Piotr Dollár, is a MATLAB toolbox in which Aggregate Channel Features (ACF) object detection code is implemented. This toolbox contains code implementation for all the stages of the algorithm such as computing of channels, feature pyramid, implementation of boosting trees, detector training and color plane conversion and extraction is implemented in MATLAB code. It requires MATLAB Image Processing Toolbox to be installed. More information about the toolbox can be found in [55].

Leddar Configurator

Leddar Configurator is a user interface designed by Leddar Tech to alter the configurations of Vu8 sensors. It can also be used to log data, view raw detections (shown in Figure C.1) and view graphical representation of data from the Vu8 sensors. This is achieved by connecting the sensor to the computer, on which Leddar Configurator is installed, using a serial port. The configurator was extensively used for the following tasks:

1. Check the correctness of the sensor's detection.
2. Configure properties such as
   (a) CAN baud rate.
   (b) CAN request and response message ID (explained in Section 4.2).
   (c) Measurement units (meters or centimeters).
   (d) Acquisition settings such as accumulation, oversampling and points.

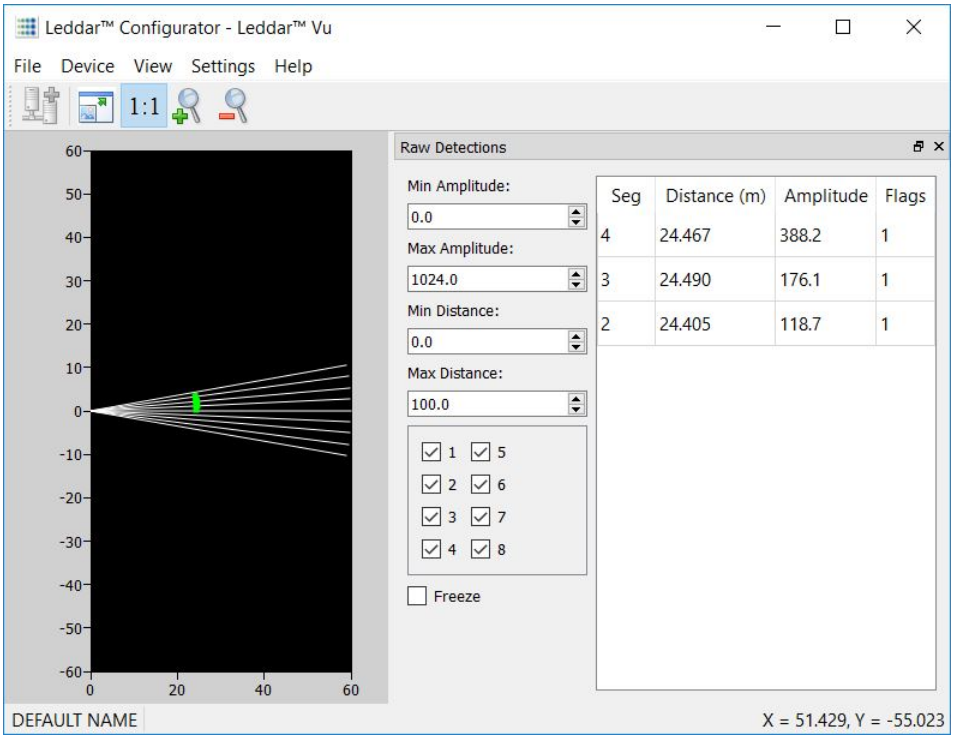

Figure C.1: Leddar Configurator [43].