DEVELOPMENT OF VISION-BASED RESPONSE OF AUTONOMOUS VEHICLES TOWARDS EMERGENCY VEHICLES USING INFRASTRUCTURE ENABLED AUTONOMY

A Thesis

by

ABHISHEK NAYAK

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Sivakumar Rathinam
Committee Members,	Swaminathan Goplswamy
	Susan Chrysler
Head of Department,	Andreas A. Polycarpou

December 2019

Major Subject: Mechanical Engineering

Copyright 2019 Abhishek Nayak

ABSTRACT

The effectiveness of law enforcement and public safety is directly dependent on the time taken by first responders to arrive at the scene of an emergency. The primary objective of this thesis is to develop techniques and actions of response for an autonomous vehicle in emergency scenarios. This work discusses the methods developed to identify Emergency Vehicles (EV) and use its localized information to develop response actions for autonomous vehicles in emergency scenarios using an Infrastructure-Enabled Autonomy (IEA) setup. IEA is a new paradigm in autonomous vehicles research that aims at distributed intelligence architecture by transferring the core functionalities of sensing and localization to a roadside infrastructure setup. In this work two independent frameworks were developed to identify Emergency vehicles in a video feed using computer vision techniques: (1) A one-stage framework where an object detection algorithm is trained on a custom dataset to detect EVs, (2) A two-stage framework where an object classification is independently implemented in series with an object detection pipeline to classify vehicles into EVs and non-EVs. The performance of many popular classification models were compared on a combination of multi-spectral feature vectors of an image to identify the ideal combination to be used for EV identification rule. Localized position co-ordinates of an EV are obtained by deploying the classification routine on IEA. This position information is used as an input in an autonomous vehicle and an ideal response action is developed.

DEDICATION

"To my mother, father, uncle, and guru,

Nagaveni, Ganapathi, Gokuladas and Prof. Sivakumar.

My strength, my joy, my inspiration"

ACKNOWLEDGMENTS

Firstly, I would like to thank Dr. Sivakumar Rathinam who believed in me since my first day at Texas A&M and has constantly motivated me to redefine my limits. To Dr. Swaminathan Gopalswamy who has encouraged me, and provided me with opportunities to experiment and work on new projects. This thesis is truly a testament to their belief in me. I would like to thank the Texas A&M Transportation Institute and Dr. Sue Chrysler who funded this project and made this work possible through her guidance and support.

Special thanks to my friends Radhika, Sai Krishna and Ronak for being my support system, and mentoring me throughout this journey. I extend my thanks to my fellow lab mate Kenny Chour with whom I spent time making and breaking stuff together, but more importantly learning new things. My hearty thanks to the IGSA-TAMU family, and all my friends who made my time in Aggieland a memorable one. Last but not at all least, my mother Nagaveni, my father Ganapathi and my uncle Gokuldas for believing in me, without whose support I would never have ventured abroad to pursue my higher education. To all its worth, this thesis is dedicated to all of them.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis committee consisting of Dr. Sivakumar Rathinam and Dr. Swaminathan Gopalaswamy of the Department of Mechanical Engineering and Dr. Susan Chrysler of the Department of Industrial Engineering.

The Emergency Vehicle (EV) detection techniques in Chapter 2 appeared as a technical paper [1] listed from the proceedings of 2019 SAE World Congress Experience. The work towards developing the concept of Infrastructure enable Autonomy in Chapter 3 was partly done by Kenny Chour and Tyler Marr of the Department of Mechanical Engineering and Deepika Ravipathi of the Department of Electrical & Computer Engineering under the guidance of Professor Swaminathan Gopalswamy and Professor Sivakumar Rathinam. The response action implementation and controller design in Chapter 4 were conducted in part by Sai Praneeth Devanuri of the Department of Civil Engineering and Mengke Liu of the Department of Mechanical Engineering and was published in an article listed in the 2019 European Control Conference (ECC).

All other work conducted for this thesis was completed by the student independently.

Funding Sources

Support for this research was provided in part by a grant from the U.S. Department of Transportation, University Transportation Centers Program to the Safety through Disruption University Transportation Center (451453-19C36).

NOMENCLATURE

IEA	Infrastructure enabled Autonomy
DSRC	Dedicated short range communication
MSSP	Muli Sensor Smart Pack
RSU	Road side Unit
RGB	Red, Green, Blue
HSV	Hue, Saturation, Value
SVM	Support Vector Machines
k-NN	k - Nearest Neighbours

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iii

ACKNOWLEDGMENTS iv

CONTRIBUTORS AND FUNDING SOURCES v

NOMENCLATURE vi

TABLE OF CONTENTS vii

LIST OF FIGURES ix

LIST OF TABLES xi

1. INTRODUCTION AND LITERATURE REVIEW 1

1.1 Introduction 1
1.2 Organization 1
1.3 Motivation 2
1.4 Related Work 3
1.4.1 Datasets 3
1.4.2 Object Detection 4
1.4.3 Object Tracking 5

2. VISION-BASED TECHNIQUES FOR IDENTIFYING EMERGENCY VEHICLES 7

2.1 RAVEV Dataset 7
2.2 FRAMEWORK - A 8
2.2.1 Limitation 11
2.3 FRAMEWORK - B 12
2.4 EV Classification in Framework B 13
2.4.1 Feature Descriptors 15
2.4.1.1 Image Pixel Array 15
2.4.1.2 HOG (Histogram of Oriented Gradients) 16
2.4.1.3 Color Histograms 16
2.4.1.4 Custom Feature Array 17
2.4.2 Classification Models 18
2.4.2.1 **scikit-learn** Classification Models 18

2.4.2.2 XGBoost Classifier 21
2.4.2.3 Neural-network Classifier 22

3. INFRASTRUCTURE-ENABLED AUTONOMY 25

3.1 IEA Simulation Architecture 27
3.2 IEA Experimental Setup 30
3.3 Vision Processing 31

4. EMERGENCY RESPONSE ACTION AND CONCLUSIONS 34

4.1 Desired Trajectory Generation 34
4.2 Lateral Controller for Autonomous Vehicle 36
4.2.1 Curve Fitting and Error Calculation 36
4.2.2 Vehicle Model and Controller Design 39
4.3 Response Action 41
4.4 Further Scope 43

REFERENCES 44

APPENDIX A. LIST OF PERFORMANCE METRICS 50

A.1 Performance Metrics 50

APPENDIX B. LOCALIZATION METHODS 51

B.1 Transformation-based Localization 51
B.2 Location-reference based Localization 52

APPENDIX C. LIST OF PARAMETERS USED BY THE CLASSIFICATION MODELS AND FUNCTIONS ... 54

C.1 *scikit_image* Package Hyper-parameters 54
C.1.1 HOG Feature Descriptor 54
C.2 *scikit_learn* Package Hyper-parameters 54
C.2.1 SVM 54
C.2.2 Random Forrest 54
C.2.3 Adaboost 54
C.2.4 Gradient Boosting 55
C.2.5 XGBoost 55
C.2.6 k-NN 55

LIST OF FIGURES

FIGURE Page

2.1 Flowchart representation of Framework A 9

2.2 Object detection outputs from YOLOv3 trained on RAVEV dataset 9

2.3 Flowchart representation of Framework B 14

2.4 Flowchart representation classification routine 15

2.5 Feature descriptors visualized for an EV 19

2.6 Feature descriptors visualized for a non-EV 20

2.7 Learning curves for neural network EV classifier training on RAVEV dataset 23

2.8 Architecture of neural network classifier 24

3.1 Architecture of an autonomous vehicle 25

3.2 Functional schematic of an IEA corridor 28

3.3 Full hardware setup of hybrid simulation 29

3.4 Detailed setup of MSSP computer processes 30

3.5 Detailed setup of vehicle computer processes 31

3.6 Experiment setup at RELLIS campus 32

3.7 View from MSSP 33

4.1 Flowchart of the emergency response action 35

4.2 Illustration of lateral error on the curve. Modified from (Liu, 2019) [2] 38

4.3 Illustration of lateral error on line 39

4.4 Illustration of the bicycle model 41

4.5 Illustration of lane change response by the emergency vehicle 42

B.1 Representation of vehicle localization error in IEA 52

B.2 Location-reference based Localization's mapped between image and world co-ordinates . . . 53

LIST OF TABLES

TABLE Page

1.1 Speed/accuracy tradeoff chart of mAP at 0.5 IOU metric on COCO test-dev. Table adopted from [3]. 5

2.1 Confusion matrix for binary EV classification corresponding to 13429 detections 11

2.2 Results of classification models to different feature vector input 21

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Introduction

Emergency response services are crucial and counted upon by people across the world in dangerous and potentially fatal road situations. Emergency responders provide urgent medical attention, manage road disputes, enforce the law or even repair unsafe road conditions. Effective response times can mean the difference between life and death in some of the worst driving emergencies for those involved in accidents and crashes.

In emergency response situations, safety is critical as response maneuvers by the human operator or even self-driving vehicles, must not present new scenarios resulting in traffic congestion or vehicle collisions. Studies [4] show that human operators still do not fully trust the automation control, and prefer to intervene in such scenarios to guarantee safety. However according to [5] a single autonomous vehicle can control the flow of at least 20 human-controlled vehicles around it, with substantial reductions in velocity deviations, excessive braking, and fuel consumption. Thus a study into developing emergency response capabilities for an autonomous vehicle is essential in reducing emergency response times. This thesis begins with the introduction of a need to develop Emergency vehicle response actions in Autonomous vehicles

1.2 Organization

In Chaper 1 we discuss the motivation and introduces some of the computer vision techniques used in this work. Chapter 2 focuses on Infrastructure enabled Autonomy (IEA) and discuss its advantages and challenges. A simulation-based feasibility study discussing the scalability and distribution of IEA technology is presented. The implementation of the experimental setup of IEA at RELLIS campus is discussed in detail. Chapter 3 discusses the different vision-based techniques that were investigated for developing methods to identify EV from a video scene. A comparative study involving the effectiveness of different combinations of detection and classification routines for different multi-spectral feature vectors used in identifying EVs is presented.Chapter 4 discusses the development of a response action for an autonomous vehicle using the EV identification algorithms implemented on IEA.

1.3 Motivation

The time taken by first responders such as police, fire, ambulance or any other public utility vehicles (commonly known as Emergency response time) is a key measure in evaluating the effectiveness of law enforcement and public safety in emergency scenarios. Average response times further vary depending on factors like time of day, traffic conditions, weather, etc. Deploying autonomous vehicles on roads is one of the avenues to reduce traffic congestion.

The Texas transportation code, as a part of the Texas Constitution and Statutes [6] states several laws and guidelines for emergency vehicles and operating guidelines for other vehicles in the presence of EV. Sec. 545.204. defines the scenarios for a streetcar being approached by authorized emergency vehicle "(a) On the immediate approach of an authorized emergency vehicle using audible and visual signals that meet the requirements of Sections 547.305 and 547.702, or of a police vehicle lawfully using only an audible signal, the operator of a streetcar shall immediately stop the streetcar clear of any intersection and remain there until the authorized emergency vehicle has passed, unless otherwise directed by a police officer."

An autonomous vehicle can effectively respond to EV only when it can accurately detect, track and map the EV in its surrounding environment and has a suitable control law to perform a safe response maneuver. Emergency vehicles in Texas are required to use audio and visual warning indicators to alert other vehicles of their presence and negotiate traffic as specified in Sections 547.305 and 547.702 of the Texas Transportation Code. There have been several published works [7], [8], and [9], which deal with exclusively using sound patterns to identify EVs. However, no concrete work has been published on real-time sensor fusion of sound and vision data, that can be used to localize the EV and control an autonomous vehicle. A patent [10] by Google discusses the different avenues they have explored for controlling their autonomous vehicles in emergency scenarios. However, audio-based techniques have limitations in terms of the accuracy of localization estimates it can provide. A fusion of vision-based EV detection methods with the audio data aids in enhancing the detection accuracies and provides for improved localization estimates [11]. In this thesis, the scope of the study is restricted to using computer vision-based techniques to develop methods for identifying EVs.

1.4 Related Work

Vehicle recognition for Advanced driver-assist system (ADAS) applications has been an actively researched topic in the field of transportation research. Several works have been carried out in identifying vehicles in a video feed [12]. Sivaram et al. [13] presents a comprehensive literature review of the work carried out on on-road vision-based vehicle detection, tracking, and behavior understanding. However, none of these works specifically discuss methods for detecting, tracking and localizing EVs.

In recent days, with the advent of machine learning and deep learning applications for computer vision, and the introduction of datasets with many hundred to thousands of labeled examples, remarkable advancements have been made in developing high-capacity object detection and tracking algorithms. We use some of these techniques to develop methods to identify EV and obtain a stream of localized position co-ordinates. Most object detection algorithms are capable of identifying multiple classes of objects and at the same time obtain excellent detection accuracy. However these results are highly dependent on the annotated datasets that are available to train them.

1.4.1 Datasets

Datasets and online challenges contribute significantly to developing image understanding methods. They play a key role in driving computer vision research as they pose specific unsolved problems, and provide standardized benchmarks for researchers to develop and compare new algorithms to the state-of-the-art methods. This allows for a quantitative evaluation of the capacities and limitations of newly developed techniques. Some of the most popular object detection data-sets include PASCAL VOC (20 object classes) [14], Microsoft COCO (80 object classes) [15], Imagenet (150,000 photographs with over 1000 classes) [16]. In the context of transportation and autonomous vehicles research, KITTI dataset by Geiger et al. [17], Cityscapes [18] and more recently BDD100K [19] have provided a platform for researchers to develop algorithms for specific tasks involved in developing self-driving vehicle technologies like object annotation, vehicle recognition, semantic scene segmentation, vehicle motion estimation, and aid in bridging the gap between simulation and real-world testing.

Even though several public datasets are available for autonomous driving, none of these datasets focus on aspects specifically required for identifying and localizing EV, neither do any of the object detection datasets list EV as a detection class. Realizing the lack of a dedicated dataset for EV identification tasks, a dataset was generates to train machine learning and deep learning algorithms, and develop methods required for identifying and localizing EVs.

1.4.2 Object Detection

Object detection involves using a bounding box to locate and classify the objects-of-interest in a given image, labeling it, and associating a confidence score to each of the detections. Robust and real-time object detection is a crucial requirement in applications pertaining to autonomous vehicle navigation and control. In a high traffic environment, it is essential that the autonomous vehicle is constantly updated with the map of surrounding objects for accurate path-planning and efficient navigation. Classical image processing techniques for vision-based object detection techniques have been extensively investigated in literature for application in vehicular detection [13] using monocular and stereo cameras. Since Krizhevsky et al. won the Imagenet challenge [20] in 2012 using deep convolutional neural networks, significant progress has been done in developing object detection methods using neural networks which perform significantly better than the classical methods.

Object detection algorithms can be broadly divided into two categories: 1) Region-proposal based object detection frameworks and 2) Regression based frameworks. Region-proposal based frameworks involve a two-step process of generating region proposals for all objects in an image first and then classify each of these proposals into specific object categories. Region-proposal based algorithms achieve higher levels of accuracy in object detection tasks but perform poorly in terms of time required for predictions. On the contrary, a regression-based framework works based on a one-step regression method where the location, classification, and bounding boxes for the objects are simultaneously proposed, reducing the prediction times required for final object detection. Some of the most popular region-proposal frameworks include R-CNN [21], Fast R-CNN [22], Faster R-CNN [23], R-FCN [24], FPN [25] and Mask R-CNN [26]. The popular regression-based frameworks include YOLO [27], SSD [28], YOLOv2 [29] and YOLOv3 [3]. Table 1.1 compares the speed/accuracy trade-off and performance of some of these algorithms when trained on COCO dataset [15].

Method	***mAP-50***	***time(ms)***
YOLOv2	21.6	25
Faster R-CNN	41.5	200
SSD321	45.4	61
DSSD321	46.1	85
R-FCN	51.9	85
SSD513	50.4	125
DSSD513	53.3	156
FPN FRCN	59.1	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	22
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Table 1.1: Speed/accuracy tradeoff chart of mAP at 0.5 IOU metric on COCO test-dev. Table adopted from [3].

1.4.3 Object Tracking

The goal of object tracking is to estimate the states of the target in the subsequent video frames given the initialized state (e.g., position and size) of a target object in one of the initial video frames.

There are three key steps involved in video analysis: detection of interesting moving objects, frame to frame tracking of these objects, and analysis of object tracks to recognize their behavior [30]. Extensive research has been conducted on developing tracking algorithms using the appearance features of the objects to be tracked [30]. Due to recent developments in object detection using machine learning and deep learning models, higher levels of detection accuracy were achievable which led to a renewed interest in the tracking method of tracking-by-detection. Tracking-by-detection involves a detection pipeline identifying a certain object-of-interest which are associated with each other over multiple frames using a tracker over time. In this perspective, object tracking can be considered as an extension of object detection, since consistent tracking essentially involves robustly connecting the detected objects between image frames. Hence tracking-by-detection results are directly dependent on detection errors of the classifier. An additional benefit of using this method is that the detection architecture itself can be replaced by newer and more potent algorithms with higher detection performance enabling better tracking performances.

Multiple Object Tracking (MOT) aims at locating multiple targets of interest, inferring their trajectories and maintaining their identities throughout the video sequence yielding their trajectories [31] which is a key aspect necessary for identifying and localizing vehicles in a video feed. In the context of MOT, for the method of tracking-by-detection, adding a tracker on top of an object detector slows down the overall tracking speeds due to the computational loads required by both detection and tracking processes. Hence it is impractical to perform multi-object detection in real-time. For automotive applications achieving lower tracking delays is a critical aspect for a tracker. Simple Online and Realtime Tracker (SORT) [32] developed by Bewley et al. using a rudimentary combination of familiar techniques such as the Kalman Filter and Hungarian algorithm for the tracking components, claims to updates at a rate of 260 Hz which is over 20x faster than the other algorithms. Since SORT has lower tracking delays and achieves performance comparable to other state-of-the-art MOT trackers, we choose to use SORT as the tracking algorithm in this work.

2. VISION-BASED TECHNIQUES FOR IDENTIFYING EMERGENCY VEHICLES [1]

The objective was to develop vision-based algorithms that will analyze the video input, identify and differentiate EV from normal vehicles, and output a continuous stream of pixel coordinates of tracked EV positions in the video. This chapter discusses in detail the different frameworks developed for identifying emergency vehicles in a video feed.

2.1 RAVEV Dataset

As discussed in the previous chapter, realizing the lack of a dedicated dataset for EV identification tasks, a dataset was developed to train the machine learning and deep learning algorithms for identifying and localizing EVs. A large set of videos containing sequences of emergency responders in action, responding to emergencies is collected on YouTube. Random images frames are extracted from these videos and annotated to locate all the objects-of-interest in the image. An open-source annotation tool YOLO_mark [33] is used to annotate the images and generate the label files in Darknet format, i.e. a .txt file with a line for every object ground truth in the image. Annotation corresponding to each object are formatted as:

$$< object - class >< x >< y >< width >< height >$$

where x, y, $width$, and $height$ are relative to the image's width and height.

The study is restricted to a binary classification of EV and non-EV and hence all the vehicles in these images are accordingly labeled into EV and non-EV. The reasoning behind limiting the evaluation to binary classification is attributed to the fact that our analysis focuses on evaluating the ability of algorithms in distinguishing between 2 similar looking object classes, rather than the algorithm's performance in multi-class object detection. Two separate datasets were developed for the detection and classification tasks, whose application will be discussed in detail in the following sections. For the detection dataset, 1070 images were annotated, listing all the EVs and non-EVs

[1]Reprinted from Nayak, Abhishek, Swaminathan Gopalswamy, and Sivakumar Rathinam. Vision-Based Techniques for Identifying Emergency Vehicles. No. 2019-01-0889. SAE Technical Paper, 2019.

in the image. For the classification dataset, the images inside the bounding boxes of the annotated image are extracted and grouped into two parts: (1) a train set containing 1000 images each of EV and non-EV, (2) a validation set containing 350 images each of EV and non-EV. The train set is used to train the learning algorithms, while the validation set served as the ground truth to evaluate the performance of the algorithm. We will refer to these as the RAVEV datasets in the following discussions.

2.2 FRAMEWORK - A

In Framework A, an object detection framework is selected based on its performance in terms of detection speeds and prediction accuracy. Detection speed is an important factor in deciding on what framework is to be used as autonomous vehicles require real-time detections capabilities to safely avert any possible collisions. The comparison of speed and accuracy trade-offs was made between several object detection algorithms as tabulated in Table 1.1. Based on the results listed in Table 1.1, YOLOv3 [3] performs significantly faster while achieving comparable mean Average Precision (mAP) scores as the other state-of-the-art object detectors. Hence we choose YOLOv3 as the object detection pipeline in framework A. YOLOv3 uses a variant of Darknet[34] architecture as the convolutional neural network (CNN) feature extraction layer. Darknet-53 has a 53 layer network trained on Imagenet. For the task of detection, 53 more layers are stacked onto it, forming a 106 layer fully convolutional underlying architecture for YOLOv3.

The YOLOv3 object detector is trained on the RAVEV dataset with a learning rate of 0.001, learning momentum of 0.9 in batches of 64 images to generate testing weights. The training weights generated by the detection model is subsequently used to detect EV in the input video feed to generate object region proposals during testing. The region proposals so generated along with the object class labels are delivered to a tracking-by-detection algorithm to yield trajectories of object motion in the video feed. The overview of framework A in flow-chart representation can be seen in Figure 2.1.

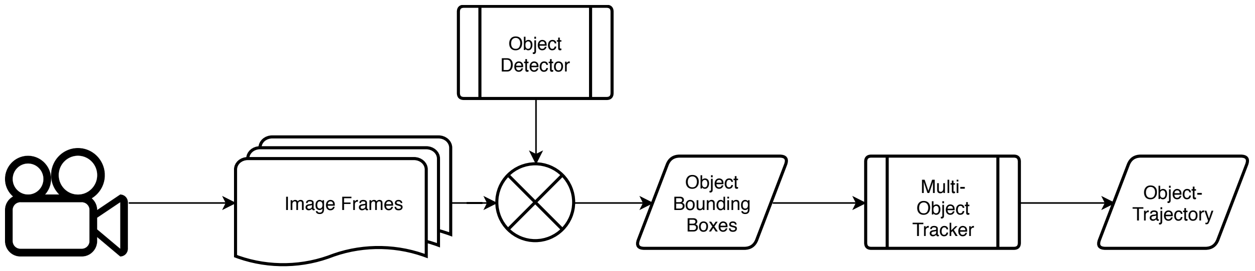


Figure 2.1: Flowchart representation of Framework A

We train the YOLOv3 detector on the RAVEV detection dataset with a learning rate of 0.001 and a learning momentum of 0.9 in batches of 64images. The trained model weights are saved and subsequently used for testing. During testing, an array of images from a video feed are input to the trained object detector. The region proposals generated for the objects-of-interest (EV and non-EV), along with the detection labels are delivered to an MOT tracker with real-time tracking capabilities.

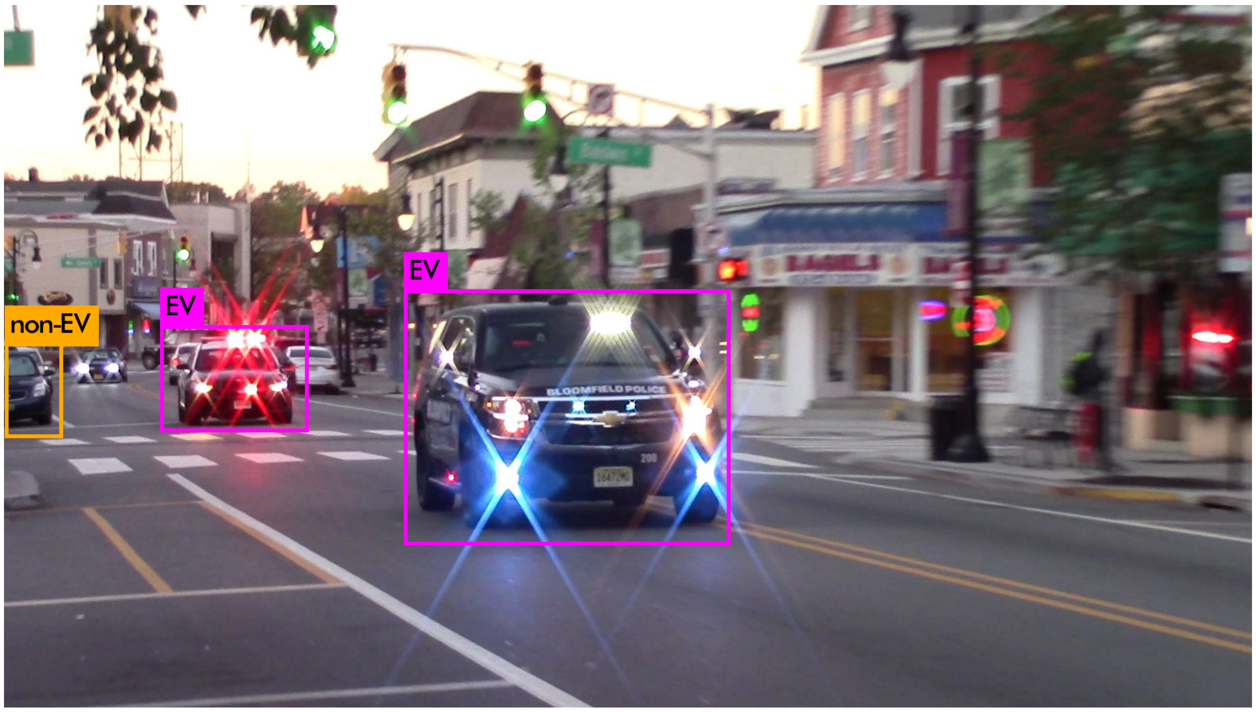


Figure 2.2: Object detection outputs from YOLOv3 trained on RAVEV dataset

As observed in Chapter 1, SORT [32] obtains has much lower tracking delays while obtaining comparable performance to other state-of-the-art object trackers. SORT (Simple Online and Real-time tracking algorithm) developed by Bewley et al. in 2016 uses a combination of Kalman Filter and Hungarian algorithm for tracking components as compared to image appearance features. SORT follows a tracking-by-detection framework for multiple object tracking. The tracker receives bounding boxes and labels of the detected objects of interest in each frame from the object detection algorithm. The inter-frame displacements of each object are approximated with a linear constant velocity model which is independent of other objects and camera motion.

The state of each target is modeled as:

$$x = \begin{bmatrix} u & v & s & r & \dot{u} & \dot{v} & \dot{s} \end{bmatrix}^T$$

where,

- $u \rightarrow$ horizontal pixel location of the centre of the target
- $v \rightarrow$ vertical pixel location of the centre of the target
- $s \rightarrow$ scale (area) of the target's bounding box
- $r \rightarrow$ aspect ratio of the target's bounding box

When a detected bounding box is obtained from the object detector, SORT updates its target states and computes the velocity components optimally via a Kalman filter framework [35]. If no detections are associated with the target, then its state is predicted without any error correction using the linear velocity model. Tracking between frames is achieved by the data association of bounding boxes between frames. The detections are assigned to existing targets by predicting its new location in the current frame. The assignment cost matrix was computed as the intersection-over-union (IOU) distance between each detection and all of the predicted bounding boxes from the existing targets. The assignment problem is solved optimally using the Hungarian algorithm [36]. Using these computationally inexpensive methods SORT tracker claims to update at a rate of

260 Hz which is over 20x faster while achieving accuracy level comparable to other state-of-the-art online trackers [32]. Hence, SORT is an ideal choice for our applications where it is critical to achieving lower tracking delays.

The focus of Framework A is to evaluate the performance of a trained object detection framework in detecting EV and its compatibility with the tracking-over-detection object tracking framework. Figure 2.2 shows the detections made by the YOLOv3 algorithms trained on the RAVEV dataset. The trained YOLOv3 model is tested on 4 different videos which contains about 13429 instances where EVs are present in the image frame. Table 2.1 shows the confusion matrix for predictions made by the YOLOv3 detector trained on the RAVEV dataset. From Table 2.1 we compute the performance metrics as defined in Appendix A.

Object Class		**Predicted**	
		non-EV	**EV**
Actual	**non-EV**	2610 (19%)	460 (4%)
	EV	1856 (14%)	8503 (63%)

Table 2.1: Confusion matrix for binary EV classification corresponding to 13429 detections

Accuracy Score = 0.634   Precision Score = 0.949

Recall Score = 0.821   F-Measure = 0.88

2.2.1 Limitation

Since Framework A is based on the tracking-by-detection framework, the tracking performance is highly dependent on the detection accuracy of the object detector throughout the video sequence where the objects need to be tracked. It is necessary to obtain a continuous stream of localization data to implement a robust way-point control on autonomous vehicles. To achieve this it is essential that the object detector accurately identifies all the object classes between continuous frames. SORT tracker works based on assigning an identity(ID) to detections belonging to the same object class in the consecutive image frames based on the vicinity of predicted pixel position coordinates in the previous frames(based on the IOU overlap score). Whenever there is a false detection in one of the intermediate frames, the tracker ID history is reset, and the subsequent detections are assigned to a new ID. The major distinguishing visual features between an EV and normal vehicles are the flashing lights on the EV. In the absence of flashing lights, both EV and non-EV appear similar and exhibit similar visual features making it harder for the object detector to distinguish between the two object classes. Thus the possibility of EV being wrongly identified as a normal vehicle in some of the intermediate frames is considerably higher when the visual features from the flashing lights of the EV are not pronounced. We can observe from Table 2.1 that there were 2316 (1856+460) false detections out of the total 13429 detections (about 17%) which can lead to tracker id resets. Thus we explore a new framework to address this limitation.

2.3 FRAMEWORK - B

The disadvantage of Framework A is that the limitations of the object detector leads to the EV region proposals being wrongly classified as non-EV in some of the intermediate frames which re-initializes the tracker IDs assigned to the object. The proposed method to overcome this limitation is to group both EV and non-EV into a single object class named 'Vehicle' during the detection step and track each instance belonging to the class 'Vehicle' by assigning them separate IDs. The region proposals of all the 'vehicle' objects in the image frame as proposed by the object detector are input to the MOT tracker. The tracker makes a frame-to-frame prediction and associates the stream of region proposals with individual object IDs. A classification layer is implemented after the tracking step to classify the vehicles into EV and non-EV. The images from inside the bounding boxes of a particular ID belonging to the class 'vehicle' are passed onto a feature extractor. The feature extractor converts these images into feature vector inputs as required by the classification models. The image classifier predicts the vehicle class based on a classification model trained on the RAVEV classification dataset and assigns the label of EV or non-EV to the stream of images of that particular ID.

The advantage of this approach is that false classifications in some of the intermediate frames by the image classifier are acceptable, contrary to Framework A. Since all the objects of interest are grouped under the class 'vehicle' before the tracking, object ID history is preserved throughout the tracking phase. Figure 2.3 shows the flowchart depicting the overall architecture of Framework B. This flowchart is different from Framework A in a manner that vehicles are tracked prior to their classification into EV and non-EV, whereas in Framework A tracking is done after vehicles are classified into EV and non-EV which as noted, is inefficient when there is a wrong classification. Even in Framework B, we use the combination of YOLOv3 as the object detector and SORT as the object tracker. However, there is no restriction on using a detection algorithm trained on the RAVEV dataset for object detection in Framework B. Any of the popular datasets that are used commonly used in transportation research can be used to train the object detector. During the classification step, the object classifier is restricted to only classes of objects corresponding to vehicles like car, truck, van and other similar classes. For our analysis, we choose the KITTI dataset for training the YOLOv3 object detector. The KITTI object detection and object orientation estimation benchmark consist of 7481 training images and 7518 test images, comprising a total of 80,256 labeled objects belonging to classes of objects normally encountered during daily driving like car, van, truck, pedestrian, etc. Only the object detections corresponding to the class 'vehicle' are classified into EV and non-EV using different EV classification models trained on the RAVEV classification dataset. YOLOv3 produced scores of 84.300% accuracy with a run-time of 0.04s under moderate difficulty conditions of KITTI dataset[17].

2.4 EV Classification in Framework B

The EV classification pipeline can be divided into 2 stages: 1) Extraction of feature descriptors or the feature vector from the input image, 2) Classification of the feature vectors into EV and non-EV using a classification model. The study was restricted to a binary classification of vehicles

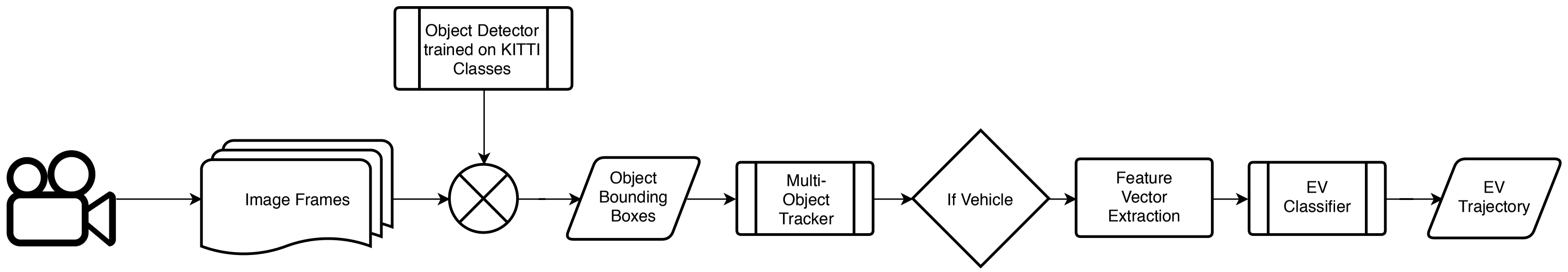


Figure 2.3: Flowchart representation of Framework B

into EV and non-EV. The different feature vectors and classification models explored in this thesis for EV classification is discussed below. Figure 2.4 illustrates the emergency vehicle classification routine.

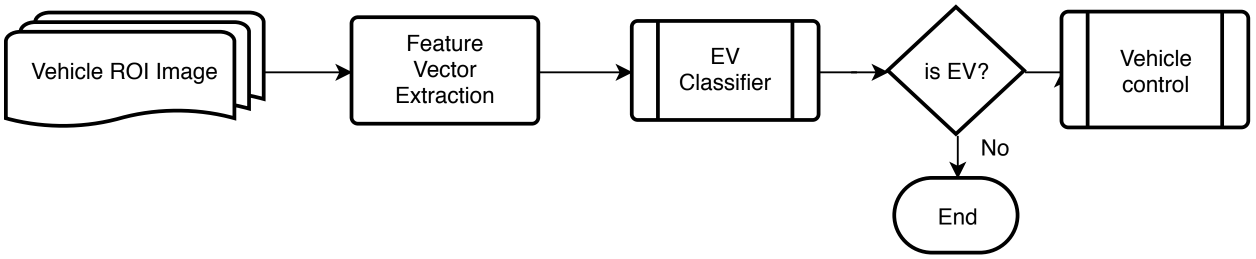


Figure 2.4: Flowchart representation classification routine

2.4.1 Feature Descriptors

The objective of step 1 in the classification routine is to identify the feature vectors which perform best towards classification between EV and non-EV. A feature descriptor is a form of representation of an image that simplifies it by extracting useful information and throwing away extraneous information. These unique attributes extracted from the images will be used by the classification models to classify them into EV and non-EV. We explore 3 commonly used feature descriptors for image classification applications, and also define a new custom-feature vector consisting of features specifically selected for identifying EV.

2.4.1.1 Image Pixel Array

Image pixel array as a feature vector refers to using the raw image of the object as input for classification. Here the pixel information is extracted from the array of bounding boxes identified under the class of 'Vehicles' during the detection step. The extracted image reshaped into a 128×128 size image to maintain parity of dimensions between different region proposals. The reshaped image is then flattened by re-sizing it into a $(128^2 \times 1)$ feature vector which will be used as input to the classification model. Part (a) of Figure 2.5 and 2.6 shows the images extracted respectively for an EV and non-EV.

2.4.1.2 HOG (Histogram of Oriented Gradients)

HOG is a feature descriptor commonly used in computer vision for object detection tasks [37]. HOG features have also been extensively used for vehicle detection tasks [38]. The HOG descriptor essentially counts occurrences of gradient orientation in localized portions of an image - detection window, or region of interest (ROI). The gradient orientation of the pixel intensities is extracted from the image inside the bounding boxes obtained from the detection algorithm. These orientations are then discretized and binned into a histogram of orientations with the channels evenly spread over 0 to 360 degrees. The histogram is then flattened into a linear array and used as the feature vector for classification. In our analysis, we discretize the gradients into 8 discrete orientations.

Part (b) of Figure 2.5 and 2.6 plots the HOG features extracted from each of their respective images in part (a). 2D image of a vehicle can be approximated to be a combination of rectangular sections. Thus consisting of straight-line edges on its boundaries. The gradient of pixel intensities across these edges thus orients mainly along 4 directions (2 along each edge) approximately. However, in an EV, the presence of bright spots due to the flashing lights introduce several gradient orientations across pixels on the boundary edges of the light spots. Thus HOG array of an EV is expected to have a more distributed orientation of pixel intensity gradients. This is reflected in the HOG feature image of EV as compared to that of non-EV where the gradient directions are observed to be restricted to only a few orientations.

2.4.1.3 Color Histograms

Color Histogram of an image is a plot of the range of pixel intensities vs the number of pixels at that intensity in each of the Red-Green-Blue (RGB) color space. Color Histograms are widely used in image classification tasks as they have been found to perform better on classification models that perform poorly on high dimensional feature vectors [39]. For our application, a 3D color histogram was extracted from an image by dividing the RGB pixel values (range: $[0 \rightarrow 255]$) into 32 bins

[40]. The number of pixels belonging to each of those bins was plotted against the color intensity bins as shown in Part (c) of Figure 2.5 and 2.6.

$cv2.calcHist()$ function from OpenCV [40] was used for this implementation to obtain the 3D histogram. The 3D (RGB) histogram array so obtained was flattened into a linear vector of size $(32^3 \times 1)$ and used as input to the classification pipeline. It was observed that reducing the number of bins deteriorates the accuracy levels of classification models, whereas an increase in the number of bins above 32 has no significant improvement. Hence we maintain 32 bins for our evaluation. As observed in Figure 2.5(c) the levels of blue and red pixel count in the color histogram plot of an EV are overall higher as compared to the histogram plot on a non-EV. This feature can be attributed to the red and blue color regions in the image of an EV due to the presence of red and blue flashing lights which is absent in a non-EV. This overall trend is expected to generate better results in classification between EV and non-EV on a classification model using Color Histograms as the feature vector. Figure 2.5 and Figure 2.6 illustrates the 3 different feature vectors that are discussed above.

2.4.1.4 Custom Feature Array

The distinguishing visual features between an EV and non-EV are the flashing neon lights mounted on the EV. We propose features from the EV image that captures information of these lights and serves as the feature vector for classification. Assuming that the flashing lights on the EV are a combination of red and blue, the following 8×1 feature vector was identified to be used for classification.

$$F^T = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 \end{bmatrix}$$

where:

- $f_1 \rightarrow HSV_{blue}{}^2 Area$ (Total area in the image corresponding to blue in the HSV Color space,
- $f_2 \rightarrow HSV_{red} Area$ (Total area in the image corresponding to red in the HSV Color space)

[2]HSV: hue, saturation, value

- $f_3 \rightarrow$ Max contour blue (Area of the largest blue contour)
- $f_4 \rightarrow$ Max contour red (Area of the largest red contour)
- $f_5 \rightarrow x_{blue}$ centroid (X-coordinate of f3 center position as a fraction to image width)
- $f_6 \rightarrow y_{blue}$ centroid (Y-coordinate of f3 center position as a fraction to image height)
- $f_7 \rightarrow x_{red}$ centroid (X-coordinate of f4 center position as a fraction to image width)
- $f_8 \rightarrow y_{red}$ centroid (Y-coordinate of f4 center position as a fraction to image height)

The above features are extracted from all the images in the RAVEV train dataset and used as an 8×1 feature vector (F) to train the classification models.

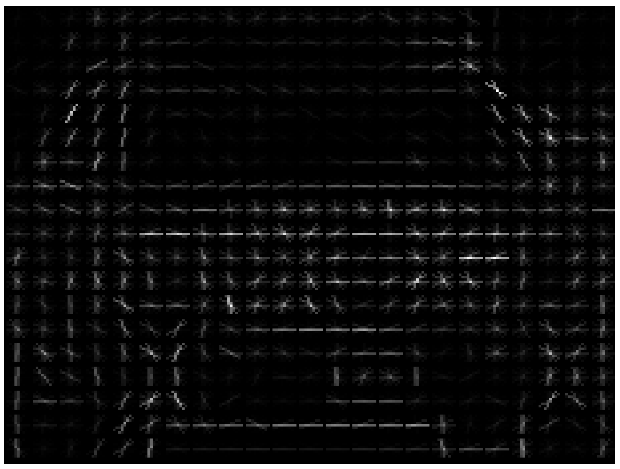
2.4.2 Classification Models

The feature vectors obtained by processing the images of vehicles are used as input to the classification models trained on the RAVEV dataset. We compare the performances of some of the most commonly used classifiers like SVM, K-Nearest neighbors, Adaboost, etc. for binary classification of EV from different feature vectors. We also develop a neural network-based EV classifier and test out the performance of the XGBoost package for EV classification tasks. The stream of objects classified as EV can be further processed for tasks like EV localization, vehicle motion planning, etc.

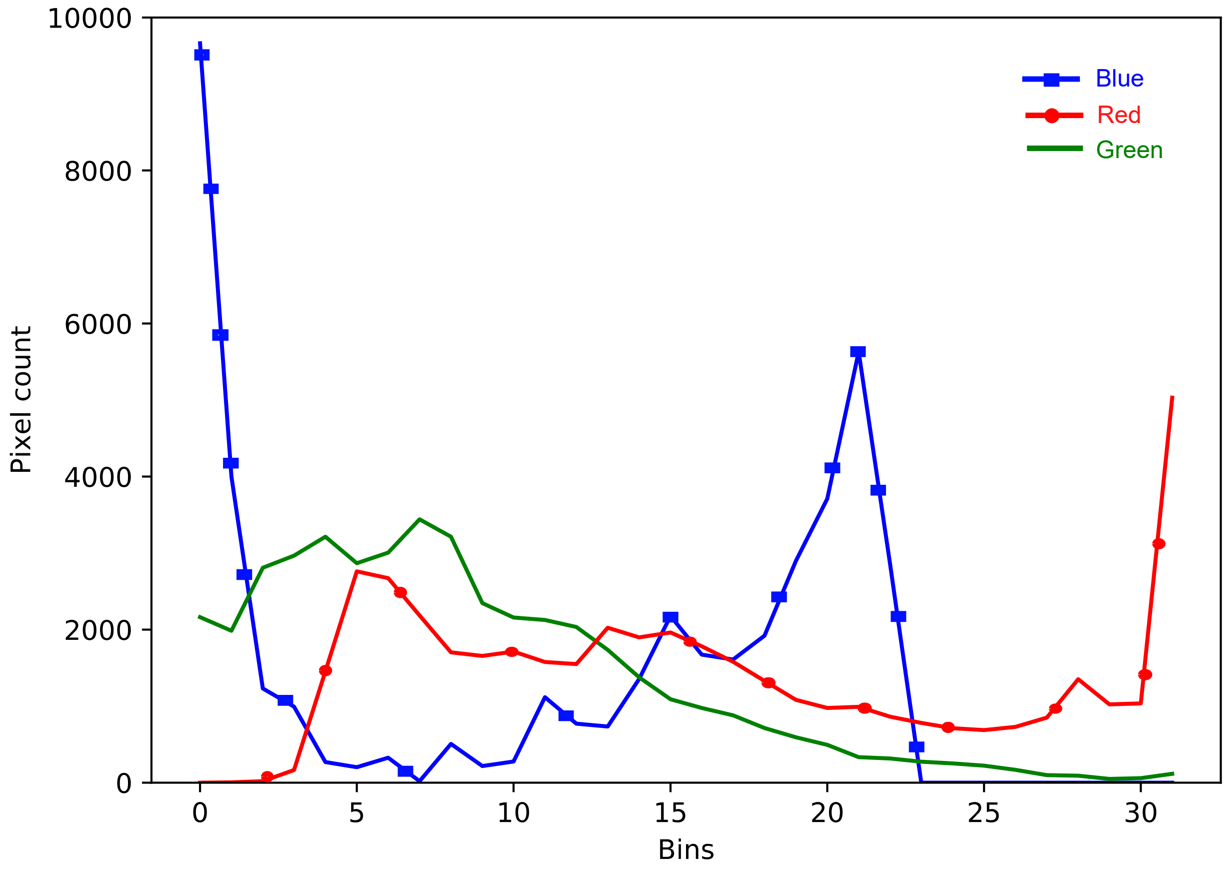
2.4.2.1 ***scikit-learn*** *Classification Models*

scikit-learn [41] is a free library for Python consists of a wide range of machine learning algorithm implementations for solving supervised and unsupervised problems. It contains various classification, regression and clustering algorithms designed to operate with the Python numerical and scientific libraries NumPy and SciPy. Classification models from *scikit-learn* were selected and trained on the RAVEV train set. The trained model was then tested out on the RAVEV validation set and the classification performance results were listed in Table 2.2. The object classification models from scikit-learn evaluated by us during this study include hyperplane based methods like

(a) EV image

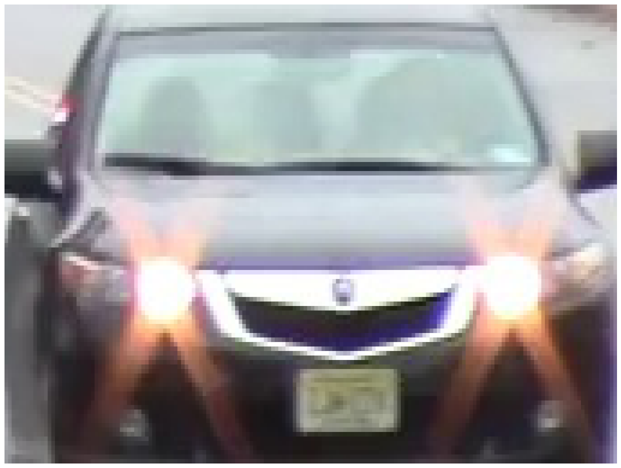


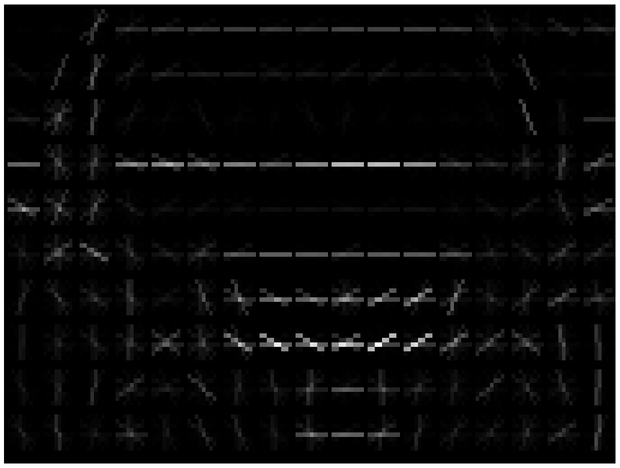
(b) HOG descriptor

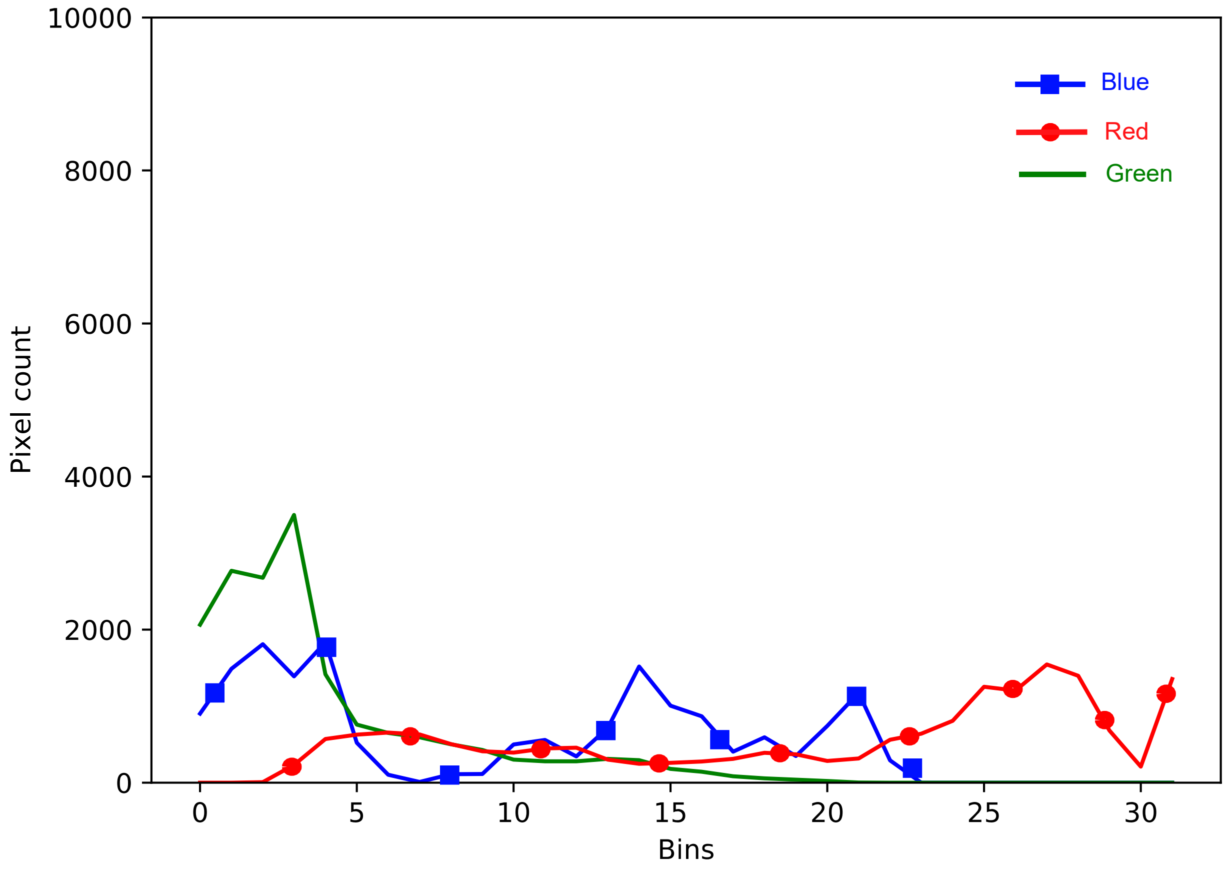


(c) Color histogram

Figure 2.5: Feature descriptors visualized for an EV


(a) non-EV image


(b) HOG descriptor



(c) Color histogram

Figure 2.6: Feature descriptors visualized for a non-EV

Support Vector Machines (SVM)[42], instance-based learning algorithms like k-Nearest Neighbours (k-NN)[43], ensemble learning methods like Random Forrests[44] and Gradient Boosting, and boosting based learning algorithms like Adaboost [45]. The hyper-parameters used for training the classification models are specified in the APPENDIX section.

2.4.2.2 XGBoost Classifier

XGBoost [46] is an optimized python implementation of gradient boosted decision tree algorithms, designed for high efficiency and performance. XGBoost is a popular tool among online-challenge competitors and data scientists as it performs extremely well on most regression and classification tasks. Hence XGBoost is evaluated for its performance towards EV classification in this work.

Feature	**Score**	**SVM**	**Adaboost**	**Rand. Forrest**	**Gradient Boosting**	**XGBoost**	**k-NN(k=3)**
HOG	*Accuracy*	58.02	99.51	95.06	95.31	97.53	76.79
	Precision	53.3	99.48	96.77	93.10	97.42	67.48
	Recall	100	99.48	92.78	97.42	97.42	99.48
Color Histogram	*Accuracy*	47.90	99.51	98.52	97.28	99.75	92.84
	Precision	47.90	99.48	98.96	96.92	100	89.10
	Recall	100	99.48	97.94	97.42	99.48	96.91
Pixel Array	*Accuracy*	85.19	96.54	95.56	94.32	98.27	94.32
	Precision	100	96.88	97.83	93.85	99.47	100
	Recall	69.07	95.88	92.78	94.33	96.91	88.14
Custom Array	*Accuracy*	83.98	93.20	92.62	92.23	94.82	89.71
	Precision	95.50	86.36	85.71	85.55	90.78	82.74
	Recall	53.00	93.25	92.02	90.80	93.5	85.28

Table 2.2: Results of classification models to different feature vector input

As observed in Table 2.2, the overall performance of Adaboost and XGBoost perform better than other classification models on the RAVEV dataset. This can be attributed to the fact that unlike SVM, k-NN and decision trees, boosting algorithms combine relatively weak learners to create a highly accurate prediction rule during training. They also do not suffer from the curse of dimensionality [47] when subjected to higher-order feature vectors, as is the case with SVM. Thus, SVM classifiers are expected to perform poorly on HOG and color histogram feature vectors, which is reflected in Table 2.2. XGBoost, due to its optimized implementation of gradient boosting, performs better than the scikit-learn implementation of gradient boosting for EV classification on the RAVEV dataset. Color Histogram and Image Pixel feature descriptors perform better with XGBoost, while HOG descriptors work better with Adaboost. Though the custom feature vector showed better results on SVM, its performance on other classification models is significantly lower. Among all the combinations, XGBoost classifier with Color Histogram feature array as input generated the best EV classification.

2.4.2.3 Neural-network Classifier

Neural networks have time and again proven to produce best-in-class results in computer vision tasks. Keras API [48] was used to construct a binary neural network classifier for EV classification. Feature descriptors are not specifically defined as an input here, as the convolutional neural network generates its own set of feature maps. For generating the feature maps, a 3-layer stack was used, each made of a 2D convolution layer with Rectified Linear Unit (ReLU) activation followed by a max-pooling layer. On top, a fully connected ReLU activation layer was implemented. A single unit sigmoid activation function is used as the final layer for binary classification of EV and non-EV. The neural network architecture is depicted in Figure 2.8.

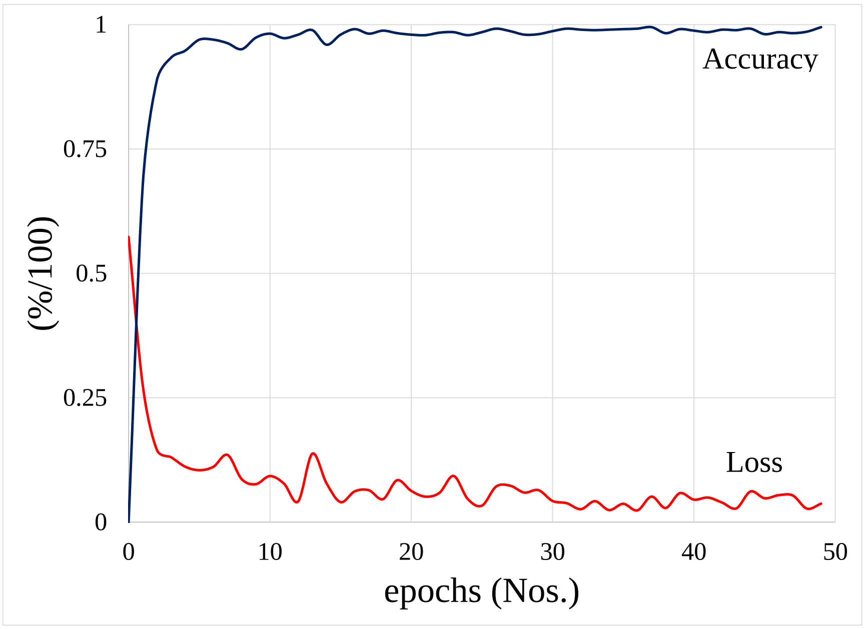


Figure 2.7: Learning curves for neural network EV classifier training on RAVEV dataset

Plots of Accuracy and Loss against training epochs of the neural network during training on the RAVEV dataset are shown in figure 2.7. The training accuracy was observed to be about 98.88% and Loss was 4.07% over the last 20 training epochs. Neural network based classifiers are significantly faster as no image pre-processing is required for classification. Also neural-network classifiers are easily to implement in conjunction with the object detection algorithms. Considering the benefits of real-time classification and ease-of-implementation we choose to use this classifier for our analysis.

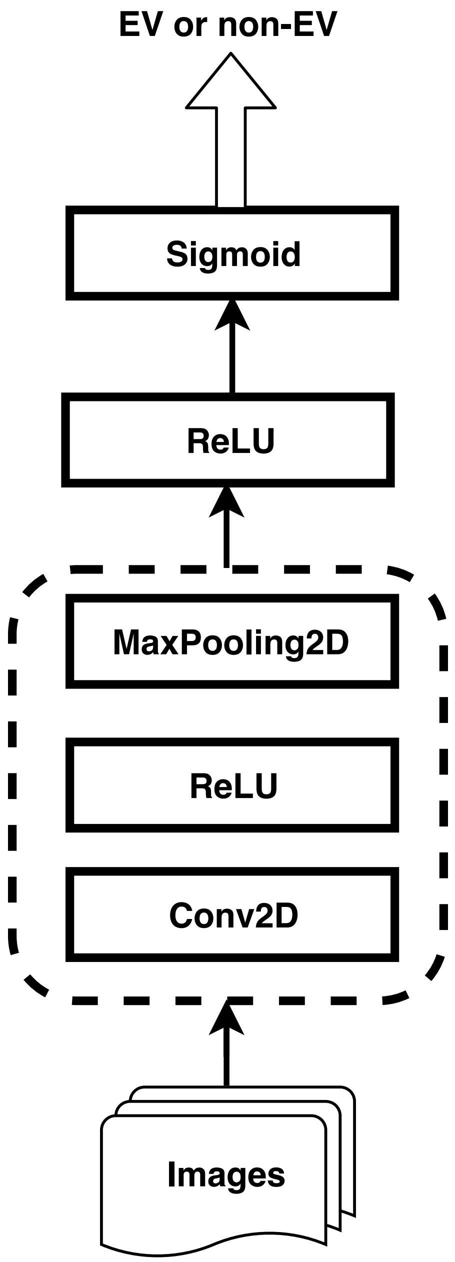


Figure 2.8: Architecture of neural network classifier

3. INFRASTRUCTURE-ENABLED AUTONOMY

The current work focuses on developing vision-based methods for identifying Emergency vehicles during daytime and developing response actions in Infrastructure Enabled Autonomy (IEA) enabled corridors. This work can further be extended to develop an emergency response in non-IEA type scenarios when the vehicle is completely autonomous and all the processing of the car takes place within the autonomous vehicles. The accuracy and performance of the proposed method during night time and off-nominal weather conditions have not been tested in this work.

A vehicle can drive autonomously if it can achieve the following aspects, 1) Sense surrounding environment using the available sensors, termed as Direct Perception [49], 2) Localize itself with respect to the environment, 3) Make driving decisions according to received perceptions and situation awareness, 4) Control the actuators. The generalized autonomous vehicle architecture is visualized the block Figure 3.1

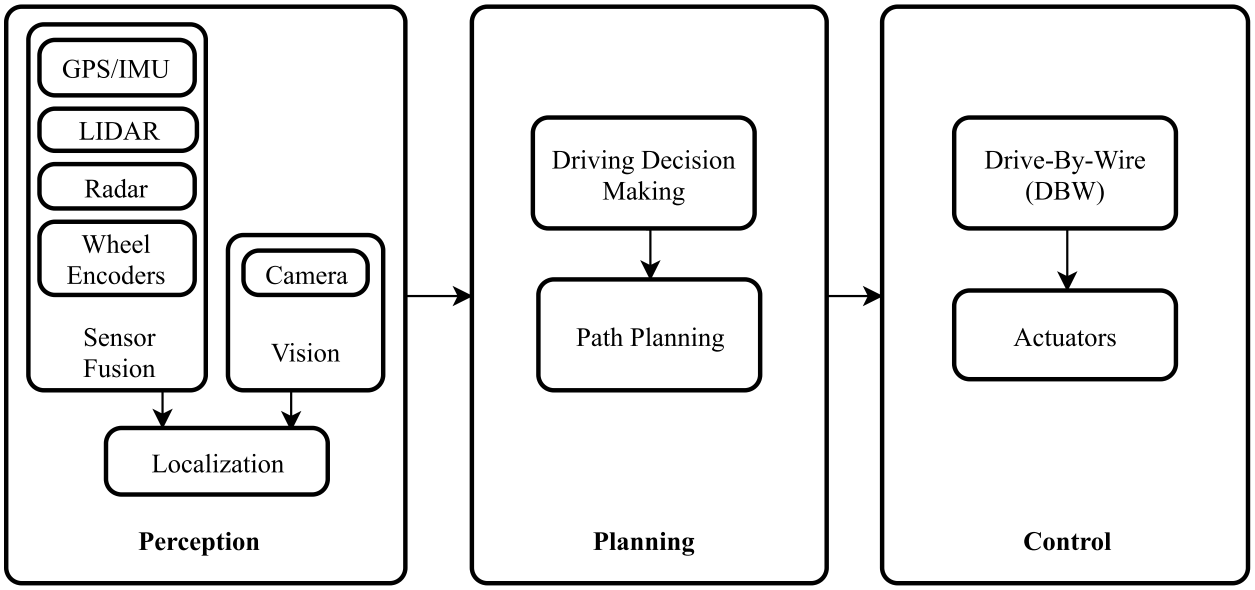


Figure 3.1: Architecture of an autonomous vehicle

Despite great efforts and research, the penetration of autonomous vehicles into roads has been very slow. This is mainly because the current prototypes demand the automobile manufacturers to bear the responsibility and liability in providing perception and situation awareness capabilities to the vehicle. As described in [50], driving functionality can be decomposed into three parts:

1. Situation Awareness (SA) synthesis using one or more sensors to develop a contextual and self-awareness for the vehicle and driver

2. Driving Decision Making (DDM), which defines the desired motion based on the self-awareness information

3. Drive-by-Wire (DBW), which generates control signals for the actual motion of the vehicle.

The penetration of autonomous vehicles is proposed to move faster if these responsibilities can be redistributed among automobile manufacturers, third party ADAS systems in the car and infrastructure [51].

Infrastructure Enabled Autonomy (IEA) is a new paradigm of transportation for autonomous vehicle driving. In this model, OEMs shall take responsibility for core functionalities relating to DBW and vehicle-level-sensing. The situational awareness capabilities will be the responsibility of infrastructure operators who generate this data through sensors embedded in the infrastructure. Driving Decision Making (DDM) can be set up to be provided by yet another third party that combines the situational awareness information coming from the infrastructure operators through a technology-agnostic communication protocol, and use standardized Application Programming Interfaces (APIs) to interface with the DBW functionalities in the cars to drive them autonomously. Through this distributed setup IEA provides a solution of shared liabilities by transferring the primary responsibility of localization from vehicle to infrastructure [51] which in-turn enables of greater situational awareness of the area under the purview of IEA. In other words, sensors embedded on the infrastructure (smart sensor pack) can provide improved localization and SA information to drive the traffic in that roadways as compared to what would be possible in a scenario with sensors installed on the autonomous vehicle.

IEA architecture is deployed on specific sections of roads or special traffic corridors by installing Road-Side Units (RSU) on either side of the road. These RSUs are fitted with multi-sensor smart packs (MSSP) containing sensors required for localizing vehicles. These MSSPs constantly monitor the vehicles in the section of the roads under the purview of IEA and aid in generating the situational awareness which can be transmitted to the vehicles subscribing to this information. MSSP includes several sensors that carry-out specific individual tasks and as a whole aid in generating the localization information. For example, cameras installed on the RSUs as a part of the MSSP are used to monitor traffic by identifying and locating all the objects of interest in the traffic corridor. MSSPs are installed with special SmartConnect devices, whose function is to establish wireless connectivity between MSSPs and the vehicles subscribing to its information and thus enabling transmission of information necessary for its localization. The SmartConnect devices are communication medium agnostic and modular so that they can be easily substituted by newer technologies.

Driving Decision Making (DDM) is locally implemented in the vehicle which uses the SA information received from the MSSP and used for DBW functionalities. Figure 3.2 depicts a typical functional schematic of a IEA enabled traffic corridor. The liabilities between different infra-solution providers can be re-distributed which in turn is expected to accelerate the deployment of self-driving cars on roads. However, there remain many challenges in the realization of IEA in terms of scale, distribution, cost and complexity that need to be addressed.

3.1 IEA Simulation Architecture

To address the challenges involved in developing the concept of IEA, a simulation environment imitating IEA needs to be developed to evaluate the architecture before the real-time testing on the vehicle. Developing such a simulation environment that incorporates all the functional components of IEA is challenging due to its complexity and scale. As the number of RSUs increase, the computational requirements of simulations increases significantly. Besides, the communication protocols involved are computationally intensive to simulate and are highly dependent on the hardware characteristics. Thus a hybrid and distributed Hardware-in-the-Loop (HIL) simulation architecture was

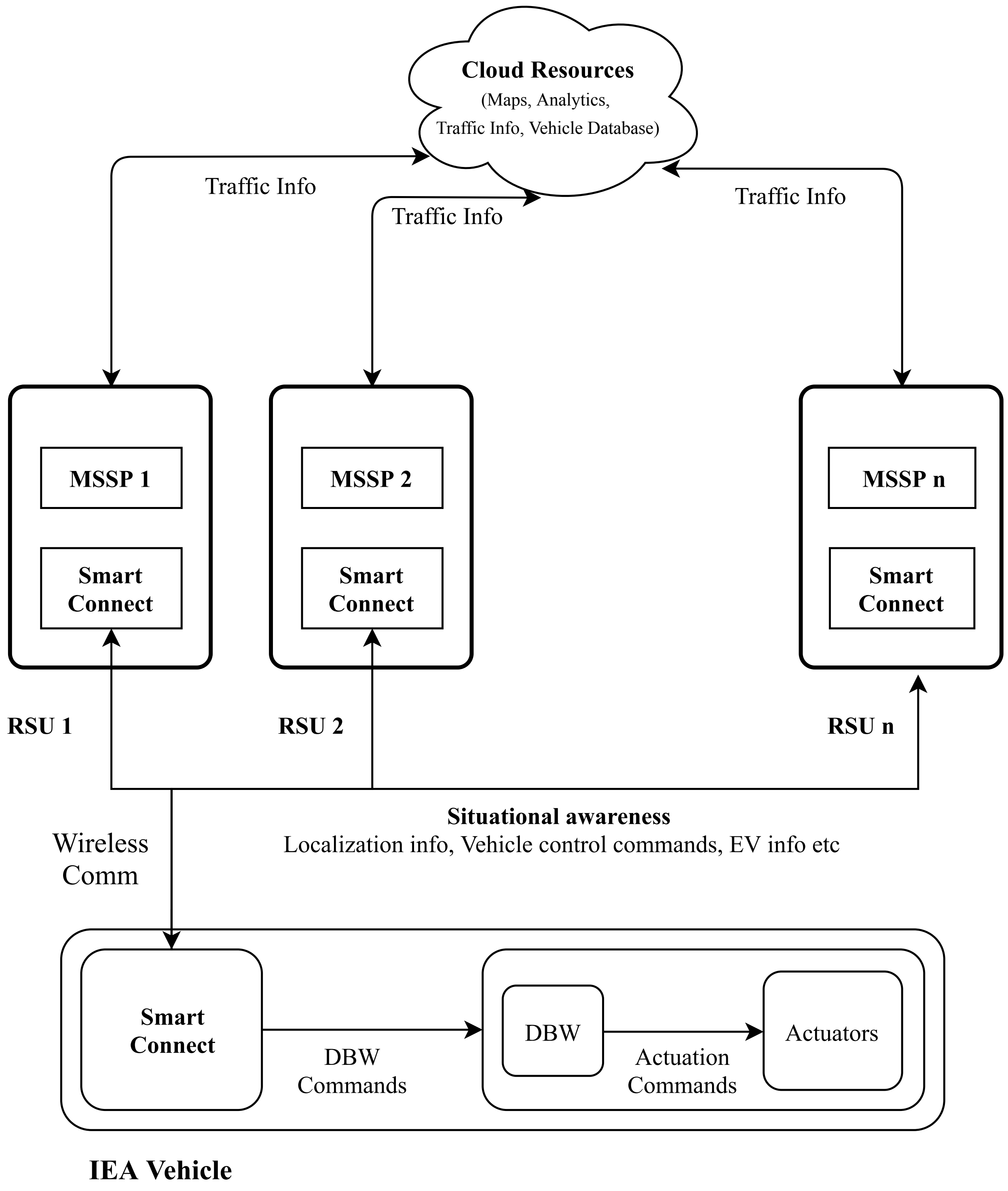


Figure 3.2: Functional schematic of an IEA corridor

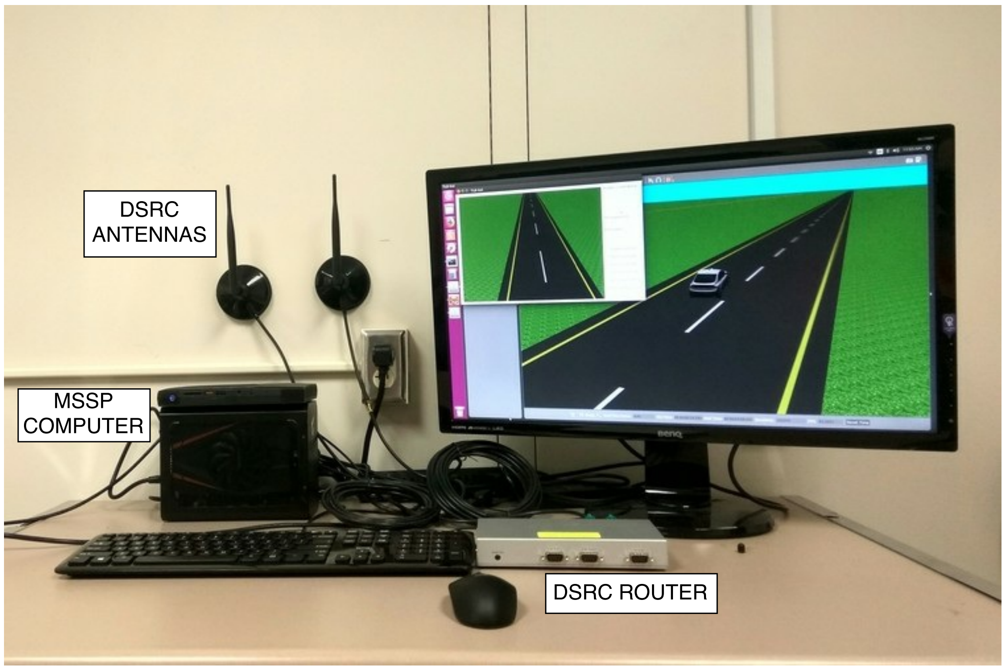


Figure 3.3: Full hardware setup of hybrid simulation

developed and is explained briefly below. A simulation environment was deveoped using Robot Operating System (ROS) and Gazebo simulator [52] which makes it possible to rapidly test algorithms, design robots, perform regression testing, and train AI system using realistic scenarios. The simulation setup is shown in Figure 3.3.

The simulation setup consists of multiple MSSP computers, each representing an RSU. A real-world like environment was developed on Gazebo to simulate an equivalent span of road covered by MSSP units. One computer is dedicated per MSSP system with an accompanying camera that simulates the camera streaming and represents the MSSPs mounted on Road-side units. Each of these computers is connected to a DSRC unit to establish communication between the MSSP computer and the vehicle computer. Dedicated short-range communication (DSRC) is the technology used to establish a vehicle to vehicle communication. It is a dedicated spectrum of 75 MHz in 5.9 GHz frequency band used for Intelligent Transportation Systems application which was established in 1998 after Congress passed the Transportation Equity Act for the 21st Century ("TEA-21") [53], in consultation with the Department of Transportation (DOT) after a petition by ITS America in

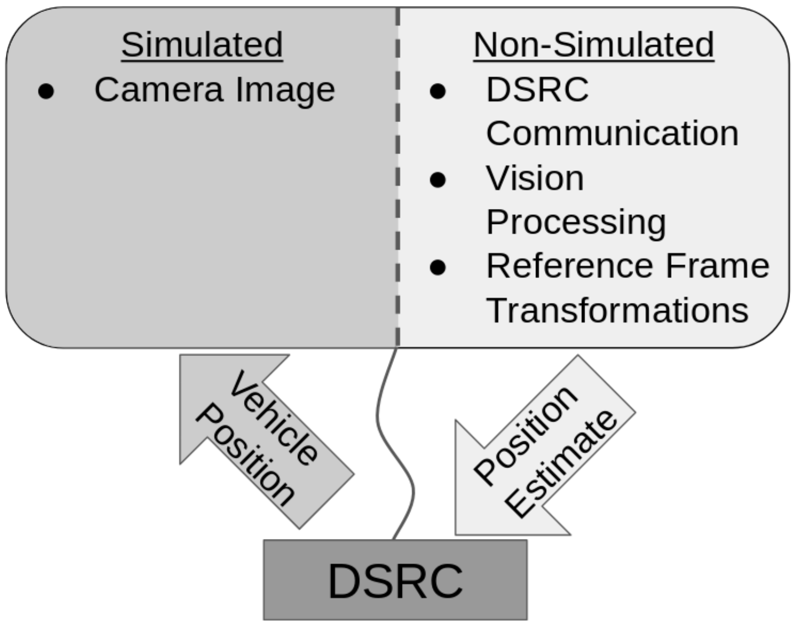


Figure 3.4: Detailed setup of MSSP computer processes

1997 . In this simulation setup, DSRC devices are used as SmartConnect devices to establish a connection between MSSPs and vehicle computers. As mentioned earlier DSRC can also be replaced by other viable communication technologies like 5G, Li-Fi, etc. A dedicated computer acts as the vehicle's on-board computer and is used to simulate the vehicle DBW functionalities. The schematic of process distribution between an MSSP computer and a vehicle computer can be seen in Figure 3.4 and Figure 3.5. An elaborate description of the simulation setup and the results obtained can be found in [50]

3.2 IEA Experimental Setup

In order to test IEA, four RSUs were setup about 50mts apart on a stretch of road at the RELLIS Campus of Texas A&M University. The RSUs were mounted with MSSPs which consist of a monocular vision camera, DSRC communicator and a computer to perform all the necessary computations. The camera was mounted at a height of around 35ft with a field of view of nearly 100 meters. The cameras used on the MSSPs were the Blackfly PoE GigE Color Camera (BFLY-PGE-20E4C-CS 1/1.8") which is an ethernet-based camera with 2MP resolution. As discussed

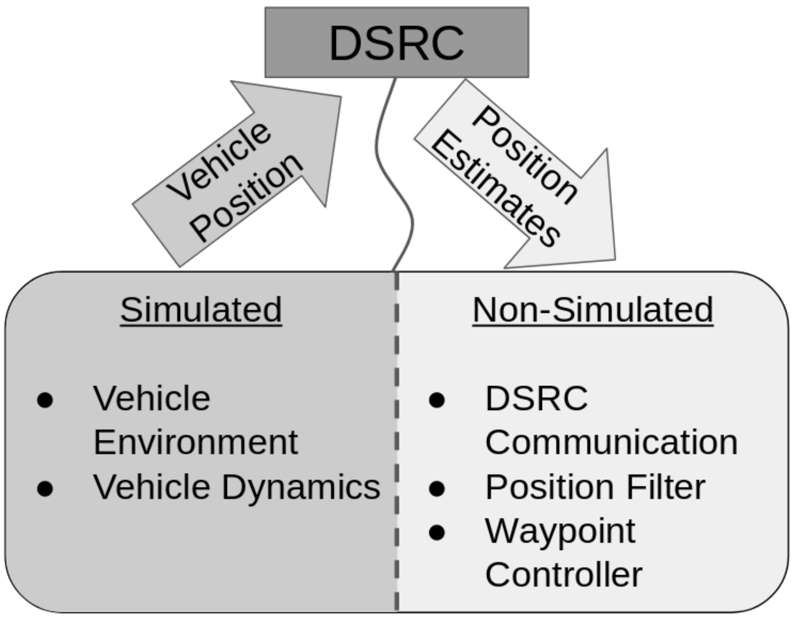


Figure 3.5: Detailed setup of vehicle computer processes

previously DSRC units are used as the SmartConnect devices to establish a connection between MSSP and vehicle computer. A high accuracy differential GPS was used for position calibration as well as for generating a ground truth reference to compare with the localization estimates. Real-time kinematic (RTK) positioning is a satellite navigation technique used to enhance the precision of position data derived from satellite-based positioning systems. PIKSI Multi GNSS Module was used as the high accuracy differential GPS unit for RTK measurements and positioning. The test vehicle with DBW capabilities is set up with a computer, DSRC communicator unit and low-resolution GPS/IMU sensors. An image of the experimental setup with the test vehicle and an RSU installed with MSSPs on a runway at RELLIS campus can be seen in Figure 3.6.

3.3 Vision Processing

The goal of vision processing in IEA is to detect and track the car in the IEA corridor and publish the image coordinates of a bounding box that tightly encloses the car as a ROS topic message. Vehicle detection is accomplished using the YOLOv3 algorithm [3] which has the best Speed/Accuracy trade-off performance ideal for IEA type applications. YOLOv3 processes the

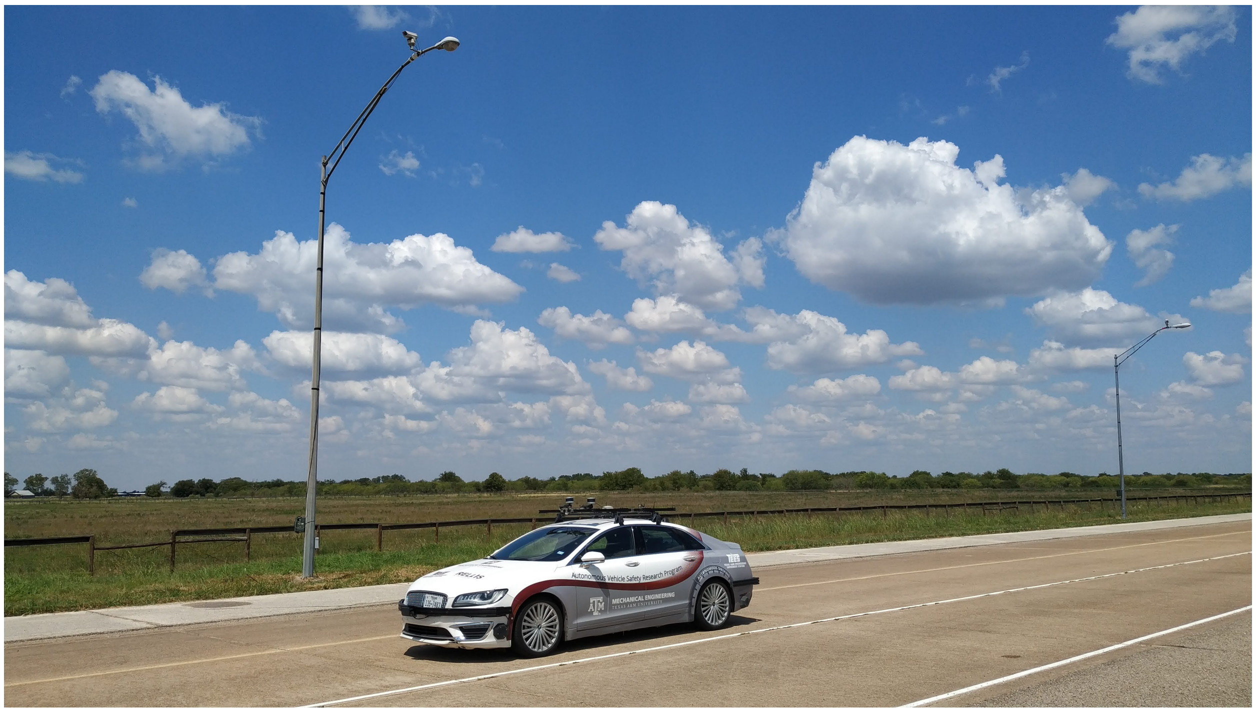


Figure 3.6: Experiment setup at RELLIS campus

images identifies the presence of a vehicle and draws a bounding box around the detected vehicle in the frame of the camera. The bounding box details are input to SORT which performs tracking to generate a stream of image position localization in the 2D image frame. Fig. 3.7 shows the view from one of the cameras along with the car detected in the frame.

The vehicle requires location data in-terms of Latitude-Longitude to perform way-point following navigation from the start point to the desired end-point. Image-to-world localization methods are used to achieve the conversion between 2D image pixel co-ordinates to 3D world co-ordinates of Latitude-longitude. Two different localization methods evaluated in this thesis work are elaborated in Appendix B. A novel location-reference based localization technique was used to obtain localization estimates in this work.

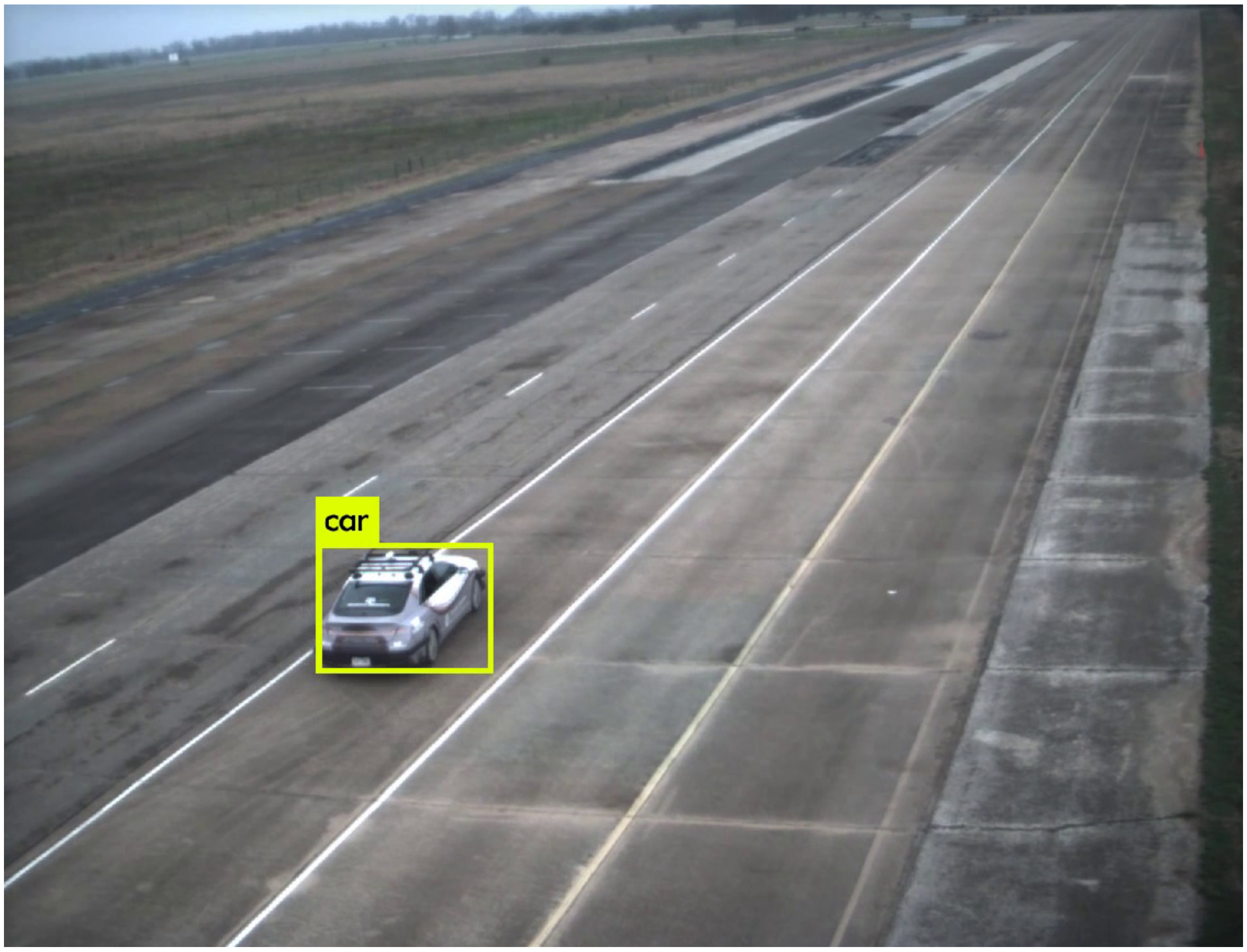


Figure 3.7: View from MSSP

4. EMERGENCY RESPONSE ACTION AND CONCLUSIONS

4.1 Desired Trajectory Generation

Vehicle navigation or path planning involves finding a geometric path from an initial point to a desired final point on global co-ordinates so that each configuration and state on the calculated path is feasible. Performing local planning and safe trajectory is central to the lateral control task for autonomous vehicles. If there are no turn maneuvers involved and the vehicle maintains the desired trajectory in the center of the lane, this driving behaviour is commonly referred to as lane-keeping. In emergency scenarios when an emergency vehicle is in the vicinity of the autonomous vehicle, an emergency lane change trajectory needs to be planned in real-time by the autonomous vehicle as a response action in order to evade any obstacle and make way for an emergency vehicle that has been detected.

The trajectory generation needs to involve real-time planning of a vehicle's maneuvers from one state to the next, satisfying the car's kinematic limits based on its dynamics. This thesis proposes a 1-lane change maneuver from the reference path for the controller. Here the lane change maneuver is not generated by a real-time planner on the autonomous vehicle, but a pre-recorded lane change maneuver is engaged to make the vehicle change lanes.The reference lane path co-ordinates are pre-recorded using Piksi Multi, a multi-constellation RTK GNSS receiver that provides centimeter-level accurate positioning. The MSSP actively scans the video feed to detect the presence of an emergency vehicle utilizing the Framework B classification routine as described in Chapter 2. On sensing the presence of an emergency vehicle in the IEA corridor, the MSSP triggers a lane change mode. The trigger is transmitted to the autonomous vehicle over the Smart-connect device. On receiving the lane change trigger, the new reference path to be followed is engaged on the autonomous vehicle and is safely maneuvered to make way for the emergency vehicle using a lateral controller. Figure 4.1 shows the overall flowchart of the respone action during emergencies.

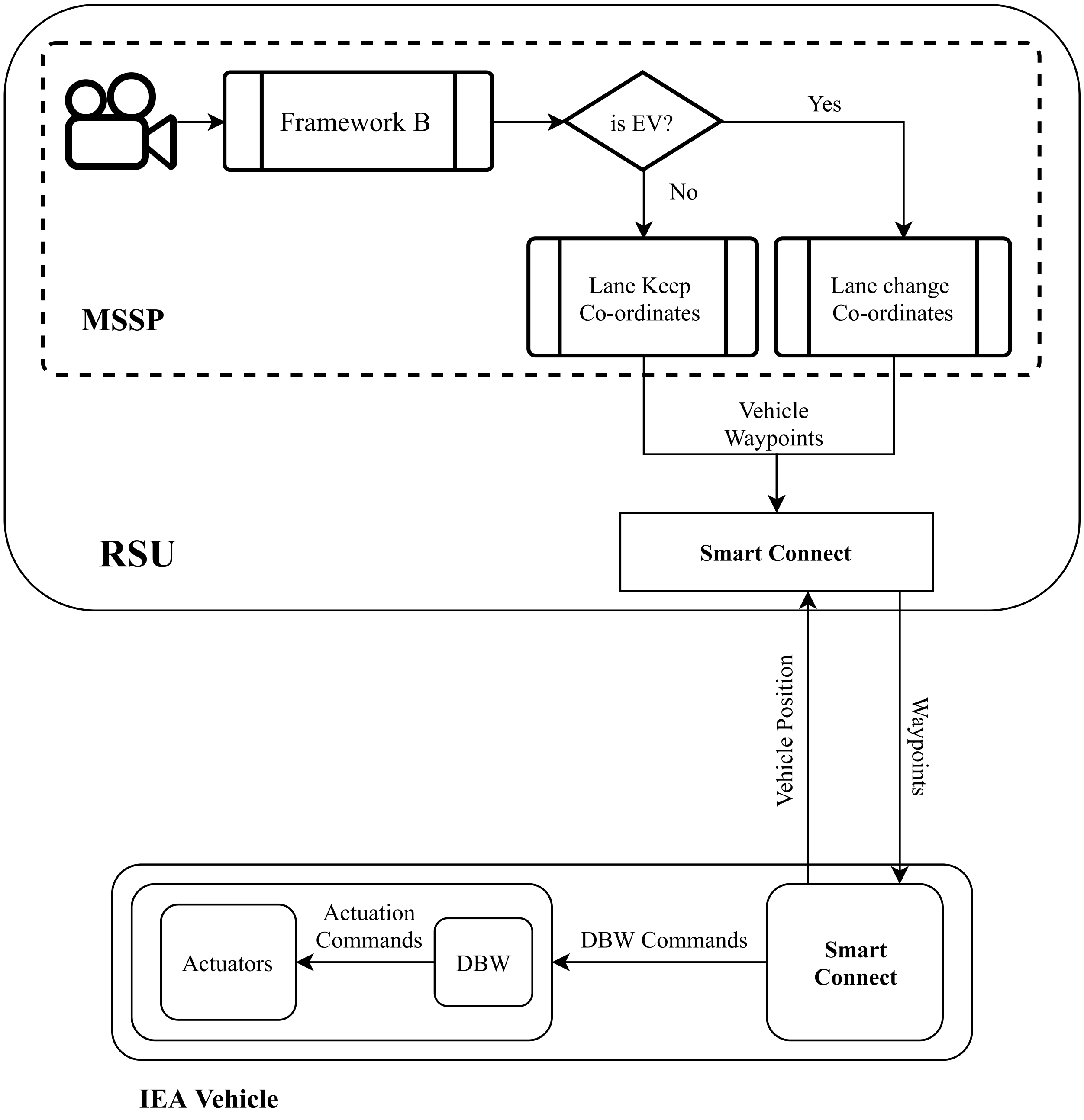


Figure 4.1: Flowchart of the emergency response action

4.2 Lateral Controller for Autonomous Vehicle

We choose a lateral controller as proposed in [2] to accomplish the lane change maneuver. The controller involves 2 parts: 1) Feed-forward controller and 2) Feedback controller. The feed-forward controller estimates the steering command based on vehicle speed (v_x), and the reference path radius of curvature (R). Under conditions of zero initial errors (lateral error(e_lat), yaw error ($\tilde{\theta}$), and yaw rate error ($\dot{\tilde{\theta}}$)), no model uncertainties and assuming zero disturbances, the vehicle should be able to follow the curve generated by the controller without any deviations. However, these conditions are too ideal to be true in real-life applications. Thus a feedback controller is required to compensate for the errors between the vehicle current state and the desired states. The state errors that are considered in this application are the lateral error (e_lat), yaw error ($\tilde{\theta}$), and yaw rate error ($\dot{\tilde{\theta}}$).

4.2.1 Curve Fitting and Error Calculation

When the emergency vehicle enters the IEA corridor the MSSP detects its presence and triggers the lane change. The vehicle engages the lane change maneuver and follows the pre-recorded data. To accomplish this task, the radius of curvature of the path that the vehicle has to track is calculated using a curve-fitting algorithm. The error signals used in the feed-forward and feedback controller design are also separately calculated.

Let (x_1, y_1), (x_2, y_2),...., (x_N, y_N) be the N most recent localization data as collected by the GPS unit. All the localization data received will be separated into two categories: a line segment or a curve segment. The data is classified into a straight line segment if the deviation of (x_i, y_i) from the line joining (x_1, y_1) to (x_N, y_N) is within a predefined threshold. Otherwise, if the deviation is outside the limits we find the center (x_c, y_c), and compute radius R of the 'least square fit' circular arc through these points using [54].

Then we determine the feedback signals: e_lat, $\tilde{\theta}$ and $\dot{\tilde{\theta}}$. The essential task of this algorithm is to fit the reference data into a circular arc. Based on the circular arc fit, the lateral deviation from the center of the road to the vehicle's position is calculated. The least-square fitting algorithm employed is specified below:

- Given (x_1, y_1), (x_2, y_2), ..., (x_N, y_N) the neared N data samples
- Find a 'least squares fit' of a circular arc, i.e., find the center (x_c, y_c) and radius R so that the error
$$J = \sum_{i=1}^{N} (R^2 - (x_i - x_c)^2 - (y_i - y_c)^2) \tag{4.1}$$
is minimized.

Setting the optimization constraints

$$\frac{\partial J}{\partial R} = 0, \quad \frac{\partial J}{\partial x_c} = 0, \quad \frac{\partial J}{\partial y_c} = 0 \tag{4.2}$$

and solving. Using this form of least square error reduces the error to zero when the point is exactly on the circular arc. Also, this form is much easier to perform partial differential respect to the center (x_c, y_c) and radius R. If the partial differentiation is too complicated to solve, there may some delay between each computation loop. Since autonomous vehicle control needs real-time implementations the curve fitting needs to happen in real-time. hence we choose the least square fitting method as discussed above.

As per [2] the lateral error when the data is fitted into the curve segment is defined as

$$e_{lat} = R - \sqrt{(X_v - X_c)^2 + (Y_v - Y_c)^2} \tag{4.3}$$

The lateral error on the curve segment is illustrated in Figure 4.2.

The lateral error when the data is fitted into a line segment is defined as

$$e_{lat} = \frac{(y_v - mx_v - c)}{\sqrt{(1 + m^2)}} \tag{4.4}$$

where,

$$y = mx + c \tag{4.5}$$

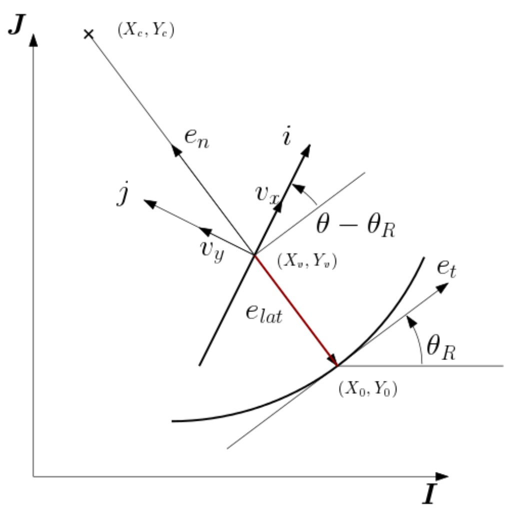


Figure 4.2: Illustration of lateral error on the curve. Modified from (Liu, 2019) [2]

is the straight-line function from given data. The lateral error on a straight line segment is illustrated in Figure 4.3.

The yaw error and yaw rate error can be computed as

$$\tilde{\theta} = \theta - \theta_R \tag{4.6}$$

$$\dot{\tilde{\theta}} = \dot{\theta} - \frac{v_x}{R} \tag{4.7}$$

Once all the error signals and radius of curvature for the path are computed, these parameters serve as input to the feed-forward and feed-back controller for the autonomous vehicle.

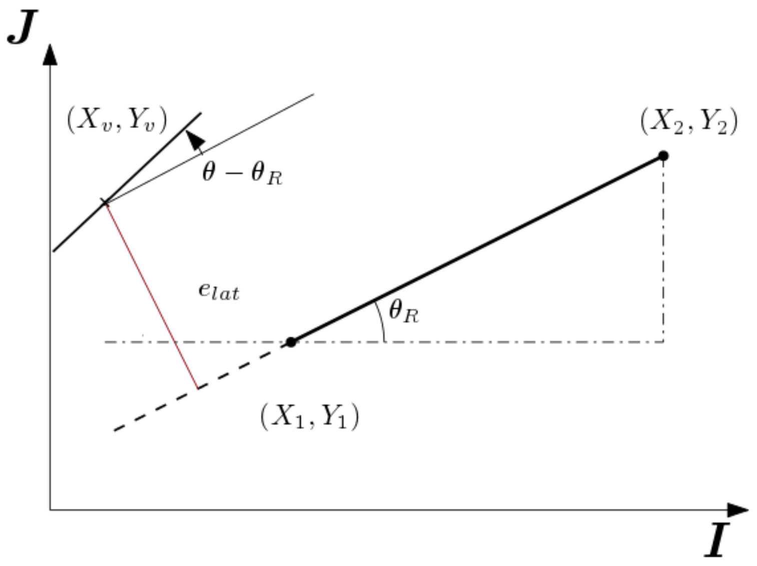


Figure 4.3: Illustration of lateral error on line

4.2.2 Vehicle Model and Controller Design

For developing the vehicle lateral controller, the vehicle is modeled as a kinematic bicycle model. Figure 4.4 illustrates the Bicycle model representation the vehicle. The non-linear terms in the vehicle model are linearized by making the following assumptions in developing the kinematic vehicle model:

- The turn radius of the vehicle (R) is far larger than the wheelbase (L).
- The left and right steer angles are assumed to be approximately the same.
- The sideslip angle of front wheels α_f is equal, so is the rear wheels side slip angle α_r.
- Both the side slip angles are assumed to be small:

$$\alpha_f \approx \delta_f - \Big(\frac{(v_y + a\frac{d\theta}{dt})}{v_x}\Big) \tag{4.8}$$

$$\alpha_r \approx -\Big(\frac{(v_y - b\frac{d\theta}{dt})}{v_x}\Big) \tag{4.9}$$

- Linear model is assumed for cornering forces:

$$F_r = C_r \alpha_r F_f = C_f \alpha_f \tag{4.10}$$

As per [2] the equation of motion can be simplified as

$$m(\frac{dv_y}{dt} + v_x\dot{\theta}) = C_f\delta_f - \frac{(C_f + C_r)}{v_x}v_y - \frac{(aC_f - bC_r)}{v_x}\dot{\theta} \tag{4.11}$$

$$I\ddot{\theta} = aC_f\delta_f - \frac{(aC_f - bC_r)}{v_x}v_y - \frac{(a^2C_f + b^2C_r)}{v_x}\dot{\theta} \tag{4.12}$$

where,

- $m \rightarrow$ Vehicle Mass
- $I \rightarrow$ Vehicle Inertia
- $a \rightarrow$ Distance of the center of mass to vehicle front tire axis
- $b \rightarrow$ Distance of the center of mass to vehicle rear tire axis
- α_f, $\alpha_r \rightarrow$ Side slip angles of the front and rear tires respectively
- C_f, $C_r \rightarrow$ Cornering stiffness of front and rear tires respectively

There are two parts to the lateral controller used: a feed-forward part and a feedback part. The feed-forward control input (δ_{ff}) provides the steering input of the vehicle based on vehicle speed and previewed path's curvature. At lower speeds, with no initial error and no disturbances/model uncertainties, the vehicle should track the circular arc without any error. However, assuming no errors is too ideal of a condition to assume, and errors inevitably must be accounted for in the design. The feedback part compensates for the errors and disturbances in the model. The feedback controller provides the feedback control input (δ_{fb}), based on lateral error (e_{lat}), yaw error (θ), and yaw rate error ($\dot{\theta}$). The summation of feed-back and feed-forward signal serves as the final

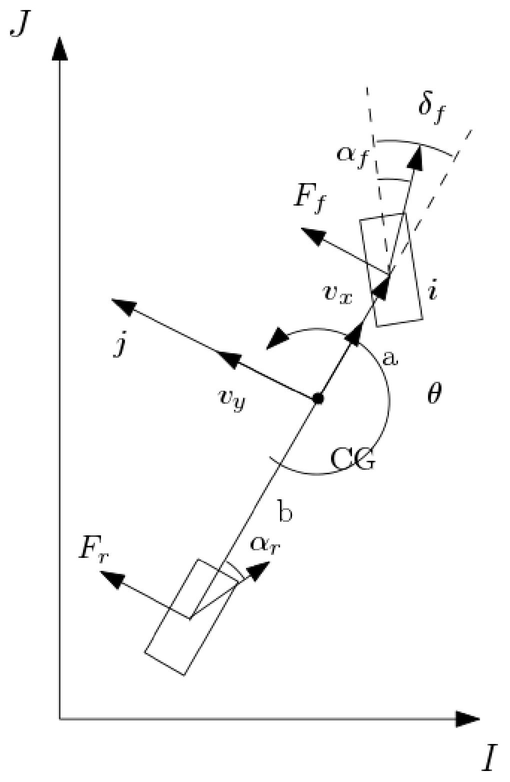


Figure 4.4: Illustration of the bicycle model

control input to the vehicle. The feed-forward, feedback and final control command, δ_f,as per [2] are defined as:

$$\delta_{ff} = \frac{L}{R} + \frac{m}{L}\left(\frac{b}{C_f} - \frac{a}{C_r}\right)\frac{v_x^2}{R} \tag{4.13}$$

$$\delta_{fb} = -k_e e_{lat} - k_\theta \tilde{\theta} = k_\omega \dot{\tilde{\theta}} \tag{4.14}$$

$$\delta_f = \delta_{ff} + \delta_{fb} \tag{4.15}$$

4.3 Response Action

The sensing and control algorithms were implemented and tested on the IEA corridors installed at the RELLIS campus of Texas A&M University. The results from a single test of lane changing and parking when an emergency vehicle has been detected are discussed below.

The test vehicle equipped with DBW functionalities was autonomously driven at 30mph (13.4 ms^{-1}) using the MSSP based localization along a straight line. An emergency vehicle is manually

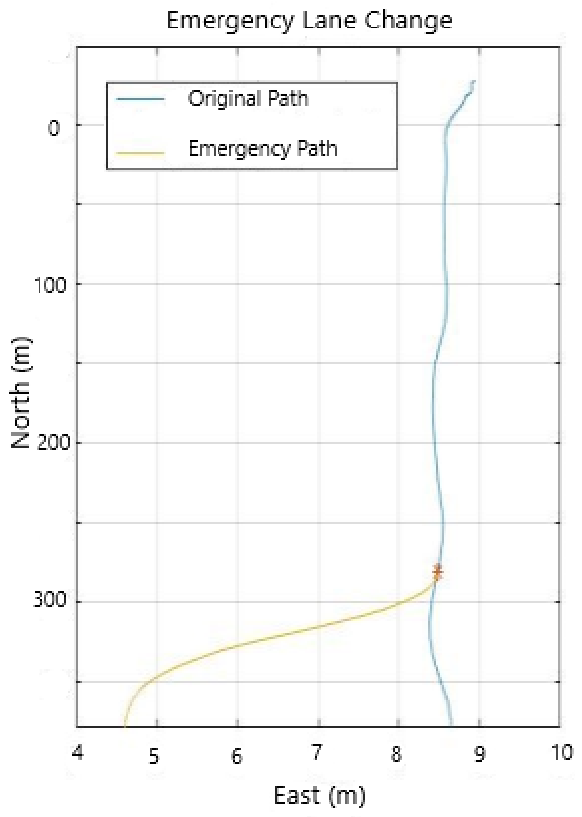


Figure 4.5: Illustration of lane change response by the emergency vehicle

driven behind the autonomous vehicle at a safe distance of about 50m without turning on the emergency lights. After a brief period, the lights on the emergency vehicle are turned on. The EV identification algorithms on MSSP which monitors the video feed, on identifying the EV, triggers the emergency maneuver and sends the lane change path co-ordinates over to the autonomous vehicle through the SmartConnect device. The controller on receiving the new path co-ordinates engages the emergency response mode, slows down and follows the new lane changing the path. The autonomous vehicle is made to stop one lane over to the right of the previous lane path. All the tests are performed by neglecting the delay involved in data processing and communication lags between the sensing and control algorithms. The original path and emergency response path are shown in Figure 4.5.

4.4 Further Scope

Developing emergency response capabilities during the night and off-nominal weather conditions are challenging and as essential as response during normal weather conditions. Hence research into this is essential for a complete EV identification system. Since the shape of the vehicle is hardly visible in the night, the predominant features that can be used for vision-based identification are the flashing lights on the emergency vehicle. Thus an extension of this work can also aim at developing algorithms to identify EVs during nighttime by capturing these flashing light features. This work has mainly focused on developing emergency vehicle identification using vision-based techniques. Emergency vehicles also output sound features that can act as a rich source of information. In crowded traffic scenarios, sound features are captured much before when the emergency vehicles are in the line of sight. Thus sound-based features when fused with vision-based techniques can also lead to more robust emergency vehicle detections.

REFERENCES

[1] A. Nayak, S. Gopalswamy, and S. Rathinam, "Vision-based techniques for identifying emergency vehicles," tech. rep., SAE Technical Paper, 2019.

[2] M. Liu, S. Rathinam, and S. Darbha, "Lateral control of an autonomous car with limited preview information," in *2019 18th European Control Conference (ECC)*, pp. 3192–3197, June 2019.

[3] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

[4] C. Lv, D. Cao, Y. Zhao, D. J. Auger, M. Sullman, H. Wang, L. M. Dutka, L. Skrypchuk, and A. Mouzakitis, "Analysis of autopilot disengagements occurring during autonomous vehicle testing," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 58–68, 2018.

[5] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, 2018.

[6] "Texas Constitution and Statutes." `https://statutes.capitol.texas.gov/?link=TN`.

[7] B. Fazenda, H. Atmoko, F. Gu, L. Guan, and A. Ball, "Acoustic based safety emergency vehicle detection for intelligent transport systems," in *ICCAS-SICE, 2009*, pp. 4250–4255, IEEE, 2009.

[8] F. Meucci, L. Pierucci, E. Del Re, L. Lastrucci, and P. Desii, "A real-time siren detector to improve safety of guide in traffic environment," in *Signal Processing Conference, 2008 16th European*, pp. 1–5, IEEE, 2008.

[9] O. Karpis, "System for vehicles classification and emergency vehicles detection," *IFAC Proceedings Volumes*, vol. 45, no. 7, pp. 186–190, 2012.

[10] Y. Tian, W.-Y. Lo, and D. I. F. Ferguson, “Real-time active emergency vehicle detection,” Sept. 1 2016. US Patent App. 14/471,640.

[11] P. Arabi and S. Zaky, “Integrated vision and sound localization,” in *Proceedings of the third international conference on information fusion*, vol. 2, pp. THB3–21, IEEE, 2000.

[12] Z. Sun, G. Bebis, and R. Miller, “On-road vehicle detection: A review,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 5, pp. 694–711, 2006.

[13] S. Sivaraman and M. M. Trivedi, “Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[14] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.

[17] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? the kitti vision benchmark suite,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3213–3223, 2016.

[19] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, “BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling,” *arXiv preprint arXiv:1805.04687*, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[21] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Computer Vision and Pattern Recognition*, 2014.

[22] R. Girshick, “Fast R-CNN,” in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[23] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 91–99, Curran Associates, Inc., 2015.

[24] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object detection via region-based fully convolutional networks,” in *Advances in neural information processing systems*, pp. 379–387, 2016.

[25] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, “Feature pyramid networks for object detection,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[26] K. He, G. Gkioxari, P. Dollar, and R. Girshick, “Mask R-CNN,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2018.

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[28] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[29] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *arXiv preprint*, 2017.

[30] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[31] C. Huang, Y. Li, and R. Nevatia, "Multiple target tracking by learning-based hierarchical association of detection responses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, pp. 898–910, April 2013.

[32] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, 2016.

[33] "Yolo_mark: GUI for marking bounded boxes of objects in images for training neural network Yolo v3 and v2." `https://github.com/AlexeyAB/Yolo_mark`.

[34] J. Redmon, "Darknet: Open Source Neural Networks in C." `http://pjreddie.com/darknet/`, 2013–2016.

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

[36] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 12, pp. 83–97, 1955.

[37] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[38] P. E. Rybski, D. Huber, D. D. Morris, and R. Hoffman, "Visual classification of coarse vehicle orientation using histogram of oriented gradients features," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, pp. 921–928, IEEE, 2010.

[39] O. Chapelle, P. Haffner, and V. N. Vapnik, "Support vector machines for histogram-based image classification," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1055–1064, 1999.

[40] G. Bradski and A. Kaehler, "OpenCV," *Dr. Dobbś Journal of Software Tools*, vol. 3, 2000.

[41] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[42] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[43] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[44] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[45] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[46] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM sigkdd International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, ACM, 2016.

[47] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *International Work-Conference on Artificial Neural Networks*, pp. 758–770, Springer, 2005.

[48] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015.

[49] R. R. Murphy, *Introduction to AI robotics*. The MIT Press, 2000.

[50] A. Nayak, K. Chour, T. Marr, D. Ravipati, S. Dey, A. Gautam, S. Gopalswamy, and S. Rathinam, “A distributed hybrid hardware-in-the-loop simulation framework for infrastructure enabled autonomy,” *arXiv preprint arXiv:1802.01787*, 2018.

[51] S. Gopalswamy and S. Rathinam, “Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 986–992, IEEE, 2018.

[52] “GAZEBO: Robot simulation made easy.” `http://gazebosim.org/`.

[53] “TEA-21, Moving Americans into the 21st century.” `https://www.fhwa.dot.gov/tea21/`.

[54] S. Ashok and G. Samuel, “Least square curve fitting technique for processing time sampled high speed spindle data,” *International Journal of Manufacturing Research*, vol. 6, no. 3, pp. 256–276, 2011.

[55] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision.* Cambridge University Press, 2003.

APPENDIX A

LIST OF PERFORMANCE METRICS

A.1 Performance Metrics

$$\text{Accuracy Score} = \frac{TP}{TP + TN + FP + FN} = \frac{\text{True Positives}}{\text{Total number of Detections}} \tag{A.1}$$

$$\text{Precision Score} = \frac{TP}{TP + FP} = \frac{\text{True Positives}}{\text{Total Positive Detections}} \tag{A.2}$$

$$\text{Recall Score} = \frac{TP}{TP + FN} = \frac{\text{True Positives}}{\text{Total Actual Positives}} \tag{A.3}$$

$$\text{F-Measure} = \frac{2 \times P \times R}{P + R} \tag{A.4}$$

APPENDIX B

LOCALIZATION METHODS

B.1 Transformation-based Localization

This approach uilizes the intrinsic and extrinsic matrix calibration parameters of a the camera and lens setup to perform localization. The center of the vehicle bounding box as output by the vision processing is used as input for transforming the vision estimates to global position coordinates. Let $[x, y, 1]$ be the homogeneous image coordinates of a pixel and $[X, Y, Z, 1]$ be corresponding vehicle position on the road relative to the MSSP in homogeneous coordinates. The mapping between the two is given by,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{B.1}$$

where K is the intrinsic camera matrix, R is the derived rotation matrix from camera's orientation with respect to X, Y, Z axes of the world and T is the translation vector derived from camera's known orientation and position. Using the back projection [55], world location of the car is obtained and is used by the controller for navigation towards the destination.

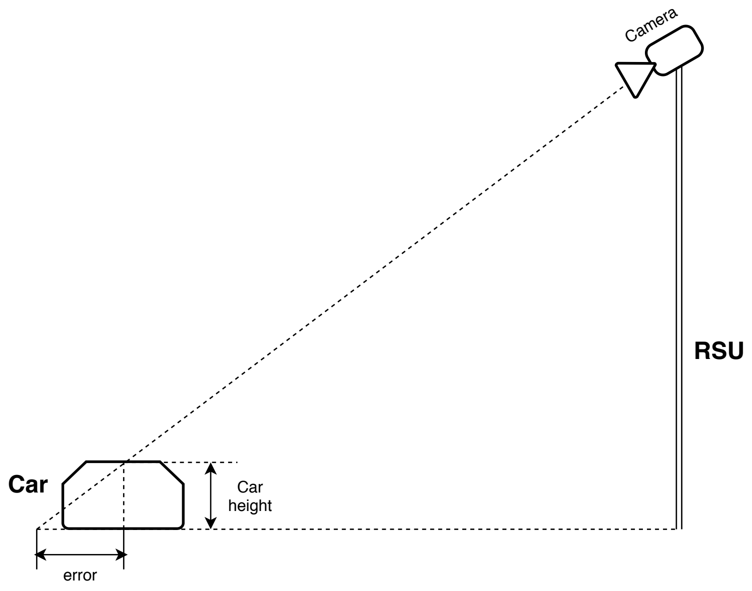
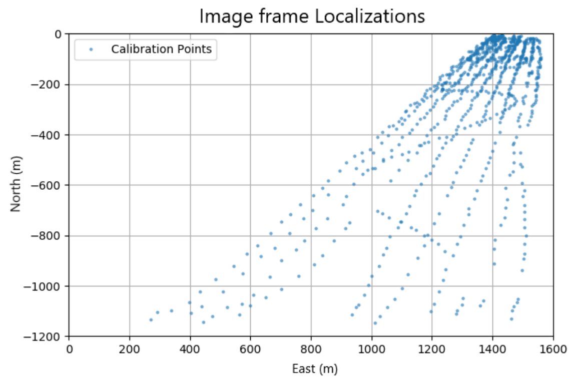


Figure B.1: Representation of vehicle localization error in IEA

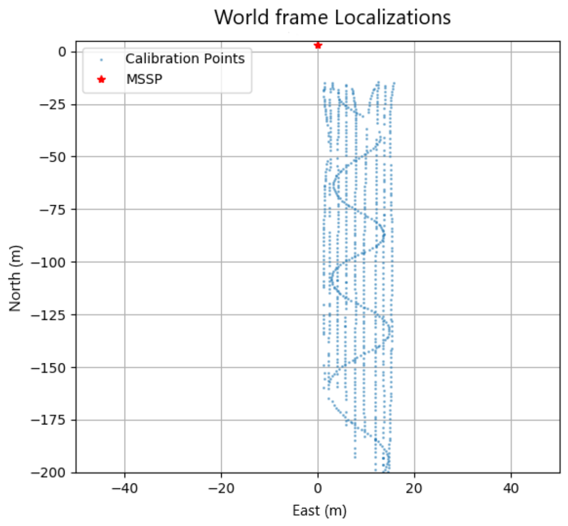
This approach is not ideal for localizing objects with a small height as compared to the height of the camera because the line of sight may not match with the true position of the vehicle. Figure B.1 shows how the true position of the vehicle would always be closer to the MSSP than the estimated position. An alternative approach is followed to compensate for this error. The approach is to calibrate each camera with real position data of the vehicle and use this calibration data as a reference to estimate the true position of the vehicle.

B.2 Location-reference based Localization

Vision processing generates a bounding box around the detected vehicle. This bounding box information can be used to extract the exact pixel coordinates in image frame. Assuming the camera is relatively stationary, the camera is assumed to return approximately the same position coordinates every time the vehicle returns to the same position on the road at a future time. Under these conditions, a one-to-one map is generated between the pixel coordinates of the vehicle and to world coordinates using Real-time kinematic (RTK) positioning measurements obtained using the Piksi Multi GNSS module. Using the mapped coordinates a real-world position estimate can be obtained for any given pixel coordinates through multivariate interpolation. An illustration of the localization mapping from image frame to world frame can be seen in B.2. The position estimate accuracy depends on the number of measurements obtained, the accuracy of the measurements and the density of the calibration points.



(a) Localization points in image coordinates



(b) Localization points in world coordinates

Figure B.2: Location-reference based Localization's mapped between image and world coordinates

APPENDIX C

LIST OF PARAMETERS USED BY THE CLASSIFICATION MODELS AND FUNCTIONS

C.1 *scikit_image* Package Hyper-parameters

C.1.1 HOG Feature Descriptor

- Orientation $\rightarrow 8$ (Number of orientation bins),
- pixels_per_cell $\rightarrow (10, 10)$ (Size (in pixels) of a cell)
- transform_sqrt $\rightarrow$ True (Number of cells in each block)
- block_norm $\rightarrow$ L1 (Block normalization method: Normalization using L1-norm)

C.2 *scikit_learn* Package Hyper-parameters

C.2.1 SVM

- kernal $\rightarrow poly$,
- max_iter $\rightarrow 50000$,

C.2.2 Random Forrest

- n_estimators $\rightarrow 10$,
- max_depth $\rightarrow none$,
- min_samples_split $\rightarrow 2$,
- random_state $\rightarrow 0$,

C.2.3 Adaboost

- n_estimators $\rightarrow 100$,

C.2.4 Gradient Boosting

- n_estimators $\rightarrow 100$,
- learning_rate $\rightarrow 1.0$,
- max_depth $\rightarrow 1$
- random_state $\rightarrow 0$

C.2.5 XGBoost

- default XGBClassifier()

C.2.6 k-NN

- n_neighbours $\rightarrow 3$,
- weights $\rightarrow uniform$,
- algorithm $\rightarrow auto$
- leaf_size $\rightarrow 30$
- p $\rightarrow 2$
- metric $\rightarrow minkowski$
- metric_params $\rightarrow None$
- n_jobs $\rightarrow 1$