Response of Autonomous Vehicles to Emergency Response Vehicles (RAVEV)

May 2020 | Final Report

SAFE-D
SAFETY THROUGH DISRUPTION

Texas A&M Transportation Institute

San Diego State University
Leadership Starts Here

VT | Transportation Institute
Virginia Tech

Disclaimer

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated in the interest of information exchange. The report is funded, partially or entirely, by a grant from the U.S. Department of Transportation's University Transportation Centers Program. However, the U.S. Government assumes no liability for the contents or use thereof.

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. 03-051	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Response of Autonomous Vehicles to Emergency Response Vehicles (RAVEV)		5. Report Date June 2020 6. Performing Organization Code:
7. Author(s) Abhishek Nayak, Sivakumar Rathinam, Swaminathan Gopalswamy		8. Performing Organization Report No. Report 03-051
9. Performing Organization Name and Address: Safe-D National UTC Texas A & M University, College Station		10. Work Unit No. 11. Contract or Grant No. 69A3551747115/Project 03-051
12. Sponsoring Agency Name and Address Office of the Secretary of Transportation (OST) U.S. Department of Transportation (US DOT) State of Texas		13. Type of Report and Period Final Research Report 14. Sponsoring Agency Code
15. Supplementary Notes This project was funded by the Safety through Disruption (Safe-D) National University Transportation Center, a grant from the U.S. Department of Transportation – Office of the Assistant Secretary for Research and Technology, University Transportation Centers Program, and, in part, with general revenue funds from the State of Texas.		
16. Abstract The objective of this project was to explore how an autonomous vehicle identifies and safely responds to emergency vehicles using visual and other onboard sensors. Emergency vehicles can include police, fire, hospital and other responders' vehicles. An autonomous vehicle in the presence of an emergency vehicle must have the ability to accurately sense its surroundings in real-time and be able to safely yield to the emergency vehicle. This project used machine learning algorithms to identify the presence of emergency vehicles, mainly through onboard vision, and then maneuver an in-path non-emergency autonomous vehicle to a stop on the curbside. Two image processing frameworks were tested to identify the best combination of vision-based detection algorithms, and a novel lateral control algorithm was developed for maneuvering the autonomous vehicle.		
17. Key Words Emergency vehicles, Autonomous Vehicles, Vision	18. Distribution Statement No restrictions. This document is available to the public through the Safe-D National UTC website, as well as the following repositories: VTechWorks, The National Transportation Library, The Transportation Library, Volpe National Transportation Systems Center, Federal Highway Administration Research Library, and the National Technical Reports Library.	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 21 / 22. Price $0

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

Abstract

The objective of this project was to explore how an autonomous vehicle identifies and safely responds to emergency vehicles using visual and other onboard sensors. Emergency vehicles can include police, fire, hospital and other responders' vehicles. An autonomous vehicle in the presence of an emergency vehicle must have the ability to accurately sense its surroundings in real-time and be able to safely yield to the emergency vehicle. This project used machine learning algorithms to identify the presence of emergency vehicles, mainly through onboard vision, and then maneuver an in-path non-emergency autonomous vehicle to a stop on the curbside. Two image processing frameworks were tested to identify the best combination of vision-based detection algorithms, and a novel lateral control algorithm was developed for maneuvering the autonomous vehicle.

Acknowledgements

This project was funded by the Safety through Disruption (Safe-D) National University Transportation Center, a grant from the U.S. Department of Transportation – Office of the Assistant Secretary for Research and Technology, University Transportation Centers Program.

We acknowledge valuable comments by Prof. Dezhen Song, Texas A&M University, College Station, who served as a subject matter expert for this proposal.

Table of Contents

List of Figures

List of Tables

Introduction

The effectiveness of law enforcement and public safety efforts is directly dependent on first responders' (e.g., police, fire, ambulance) response time in emergency scenarios. The Texas Transportation Code states several laws and guidelines applicable to emergency vehicles (EVs) and operating guidelines for other vehicles operating in their presence. Sec. 545.156 of this code defines the scenarios for a vehicle being approached by an authorized emergency vehicle as follows [1]:

a) On the immediate approach of an authorized emergency vehicle using audible and visual signals that meet the requirements of Sections 547.305 (Restrictions on Use of Lights) and 547.702 (Additional Equipment Requirements for Authorized Emergency Vehicles), or of a police vehicle lawfully using only an audible or visual signal, an operator, unless otherwise directed by a police officer, shall:

    1) yield the right-of-way;
    2) immediately drive to a position parallel to and as close as possible to the right-hand edge or curb of the roadway clear of any intersection; and
    3) stop and remain standing until the authorized emergency vehicle has passed.

b) This section does not exempt the operator of an authorized emergency vehicle from the duty to drive with due regard for the safety of all persons using the highway.

Human operators generally do not follow these instructions precisely either due to lack of knowledge, willful negligence, or the alarm created by an EV's presence. Lack of knowledge, in particular, is likely due to different states having different legislation pertaining to traffic rules when navigating around EVs.

In these situations, public safety is critical, as response maneuvers by the human operator or an autonomous vehicle must not present new scenarios that may result in traffic flow disruption or vehicle collisions. In autonomous vehicle applications, studies [2] show that human operators still do not fully trust automation and prefer to intervene in such scenarios to guarantee safety. An autonomous vehicle can safely respond to an EV only when it can accurately detect, track, and map the EV in its surrounding environment. Emergency vehicles in Texas are required to use visual and audio warning indicators to alert other vehicles of their presence and negotiate traffic as specified in Sections 547.305 [3] and 547.702 [4] of the Texas Transportation Code. There are several published articles [5] [6] [7] which exclusively deal with identifying EVs based on sound signals. In this project, we investigate how an autonomous vehicle can sense and safely yield to an EV based on visual data. Prior to our work, we did not find any article in the area of vision-based detection of emergency vehicles.

In this work, we restrict our scope to exploring different vision-based techniques for identifying EVs and control algorithms for safely parking the autonomous vehicle after an EV is identified. Potential applications of this research include developing emergency response capabilities in autonomous vehicles, developing advanced driver-assistance systems with features like parking assist in the presence of an EV, and deployment of EV sensing capabilities in smart-infrastructure enabled autonomous systems [8] [9] [10] . Specifically, the contributions of our work are as follows:

1. We implemented two frameworks for identifying, classifying and tracking EVs in real-time using datasets collected at the Texas A&M Rellis campus.
2. Several well-known classifiers were implemented to identify the best algorithm that would provide a good tradeoff between classification performance and computation time.
3. As soon as an EV is identified, the autonomous vehicle maneuvers to park itself on the curbside. The novel aspects of the control algorithm developed for this parking action are as follows: (a) a fast “least-squares" technique is used to fit a trajectory consisting of circular arc and straight line segments to form a reference trajectory for the autonomous vehicle to follow from its current position, (b) the reference trajectory is used in the computation of the feed-forward control of the autonomous vehicle, and (c) a feedback controller is developed (based on a fixed structure controller concept) to account for any real-time errors that occur when the autonomous vehicle deviates from the reference trajectory. The developed control technique is generic and can be used in other platooning and emergency maneuvering scenarios.

Methods

Response of Autonomous Vehicles to Emergency Response Vehicles (RAVEV) Dataset Collection

A large set of videos containing emergency responders in action was collected at Rellis to test and benchmark the performance of the vision algorithms used in this project. The original plan was to collect these videos in collaboration with members from Texas A&M Engineering Extension Service (TEEX). However, as scheduling operations ahead of time with the TEEX members was challenging, the research team purchased emergency lights and installed them on personal cars to perform data collection. Image frames were extracted from these videos and annotated to locate all objects of interest in the images. The dataset was separated into two sets. The first set (Dataset I) was used for detection, and a second set (Dataset II) was used for the classification tasks. For the detection dataset, 1,070 images were annotated, listing all the EVs and non-EVs in each image. For the classification dataset, the images inside the bounding boxes of the annotated images were extracted and grouped into two parts: (1) a training set (Dataset II.1) with 1,000 images, each of

which contained an EV and a non-EV, (2) a validation set (Dataset II.2) with 350 images, each of which contained an EV and a non-EV. The training set was used for the learning algorithms while the validation set served as the ground truth to evaluate the performance of the classification algorithms.

Vision Based Identification of EVs

Two independent EV identification frameworks were investigated: (1) In Framework 1, the EVs and non-EVs in each image were directly identified using a machine learning algorithm and tracked across frames; (2) In Framework 2, all the moving objects were first identified using a neural network, and then classified into EVs and non-EVs using an object detector.

Framework 1

In Framework 1, an object detection framework was selected and trained using the RAVEV Dataset I containing images with both EVs and non-EVs. The training weights generated by the detection model were subsequently used to detect EVs in the input video feed and generate object region proposals. The region proposals generated, along with the object class labels, served as input to a detection-based tracking algorithm, which yielded tracked trajectories of the object in the video. The overview of framework 1 is illustrated in Figure 1.

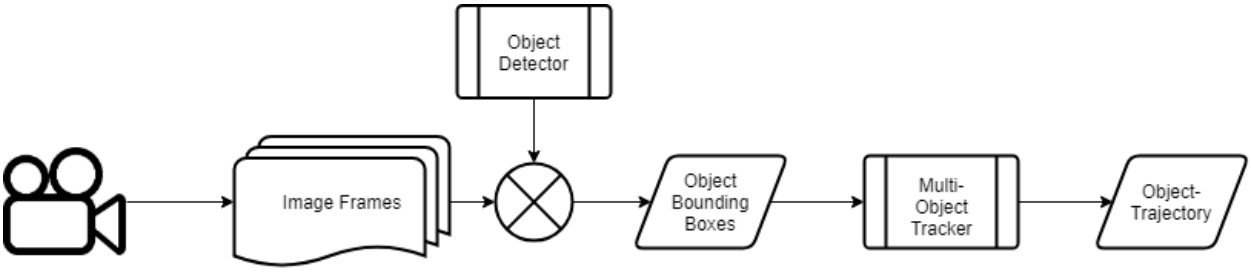

Figure 1. Flowchart representation of the sequence of steps for EV detection in Framework 1.

For object detection, we chose the You Look Only Once (YOLO) [11] algorithm due to its superior performance and speed compared to other state-of-the-art object detectors. The YOLOv3 object detector was trained on the RAVEV Dataset I, with a learning rate of 0.001 and learning momentum of 0.9, in batches of 64 images to generate testing weights. During testing, the video feed was passed on as input to the trained YOLOv3 object detector. The region proposals generated for objects of interest (EVs and non-EVs) in each frame, along with the detection labels, were sent to an object tracker to obtain continuously tracked object coordinates in the video feed. We used the simple online real-time tracker (SORT) algorithm developed by Bewley et al. [12] for tracking. This algorithm uses a combination of Kalman Filtering and the Hungarian algorithm for estimation and association of object bounding boxes between frames, as compared to the use of appearance features. Using these computationally inexpensive methods, the tracker was reported to update at a rate of 260 Hz, which is over 20 times faster than other state-of-the-art online trackers while achieving similar accuracy levels.

Limitations of Using Framework 1

Since Framework 1 is based on the tracking-by-detection framework, the tracking performance was dependent on the detection accuracy of the object detector throughout the video sequence containing the object to be tracked. It was essential for the object detector to accurately identify the object class throughout the video sequence, as this was required for the tracker to associate the detections in the subsequent image frames with the same object ID. Whenever there was a false detection or lack of detection in one of the intermediate frames, the tracer history was reset, and the subsequent detections were attributed to a new object ID. The distinguishing feature between an EV and a regular vehicle is the EV's flashing lights. In the absence of flashing lights, both object classes look similar and thus have similar visual features. Accordingly, there existed a possibility of an EV being wrongly identified as a regular vehicle in some of the intermediate frames where the visual features from the flashing lights were not very pronounced.

Framework 2

Framework 2 aimed to address the limitations of Framework 1. In Framework 1, an EV's proposed regions were being classified as non-EVs in some of the intermediate frames, which re-initialized the object ID in the tracker. This was prevented by considering both EVs and non-EVs as a single object class of vehicle during the object detection step. The region proposals generated for all the vehicles in the image frame served as an input to the SORT tracking algorithm. Once the tracker made a frame-by-frame prediction and associated the frames with an object ED, all the tracked objects were passed on to the image classification pipeline to be marked as an EV or a non-EV. The part of the image inside the bounding boxes proposed by the object detector was processed to extract feature vectors that served as input to the EV classifier.

The advantage of this approach was that we could afford some false detections in the intermediate frames. Since all the objects of interest were grouped under the vehicle class, object IDs were preserved throughout the tracking stage. Figure 2 shows the flowchart of the overall architecture of Framework 2.

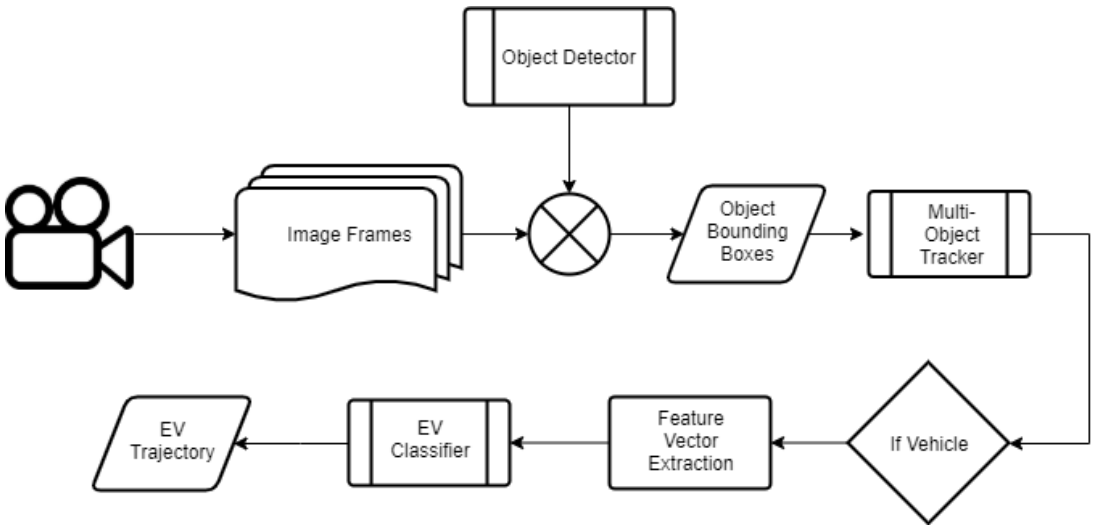

Figure 2. Flowchart representation of the sequence of steps for EV detection in Framework 2.

Framework 2 also used a combination of the YOLOv3 object detector and the SORT object tracker. Since the object detection and the object classification steps were clearly delineated in framework 2, the YOLOv3 could be trained on any of the popular datasets to identify a vehicle's object class. We used the YOLOv3 trained on RAVEV Dataset II.1 for object detection. YOLOv3 exhibited an accuracy score of 86% and a precision score of 95% on the RAVEV Dataset II.1. The models and the feature vectors used to further classify the vehicles will be discussed in the following sections.

EV Classification

A classification algorithm can be implemented in two steps. In the first step, the key features that characterize the given region of the image are extracted and stored as a feature vector. In the second step, a classification algorithm uses the feature vector to identify whether the given region corresponds to an EV or a non-EV.

Feature Descriptors and Feature Arrays

A feature vector consists of unique attributes extracted from the images of objects that will be used by the models to classify them into different classes. Some of the features we used in our feature arrays included the image pixel array, histogram of oriented gradients (HOG) descriptor, and color histogram. The array of bounding boxes associated with a single object ID was obtained from the object tracker and its corresponding image information was extracted from the video frame. The obtained image was processed to extract features or feature descriptors. This processed image was converted into a feature vector, which was then used as an input to the image classifier.

We explored three of the commonly used feature vectors for image classification (numbers 1–3 below) and developed one custom-feature vector (number 4 below) consisting of specific features to identify EVs.

1. **Image Pixel Array:** Image pixel array as a feature vector refers to using a raw image of the object as an input for classification. The image inside the bounding box is extracted using the data received from the object tracker and reshaped into a 128*128 image (using the reshape function in OpenCV). This is done to maintain parity of dimensions between different images. This reshaped image is then converted into a ({128*128} x 1) feature vector, which is used as input to the classification model.
2. **HOG (histogram of oriented gradients):** HOG is a feature descriptor commonly used for object detection tasks in computer vision research [13]. HOG features have also found extensive applications for vehicle detection tasks [14] in transportation research. HOG features are extracted by computing f gradient orientation of the image's pixel intensities. These orientations are discretized and binned into a histogram. The histogram is then converted into a linear array and used as the feature vector for classification. In our analysis, we discretized the gradients into nine orientations.
3. **Color Histograms:** A color histogram of an image is a plot of the range of pixel intensities vs the number of pixels. Color histograms are widely used in image

classification tasks, as they have been found to perform better on classification models, which perform poorly on high dimensional feature vectors [15]. For our application, a 3D color histogram was extracted from an image by dividing the RGB pixel values (range: [0, 255]) into 32 bins and plotting the number of pixels belonging to each of those bins. We used the *cv2.calcHist()* function from OpenCV [16] for our implementations. The histogram array containing the value of pixel quantities that belongs to the bin of each of the three colors was flattened into a linear vector of size ((32*32*32)x1) and supplied to the classification pipeline. It was observed that reducing the number of bins deteriorated the accuracy levels of the classification model, whereas an increase in the number of bins showed no significant improvement.

4. **Custom Array:** The distinguishing factor between a non-EV and EV in terms of visual features are the lights mounted on the EV. We identified features from the EV image that would capture details of these lights. Assuming that the lights on the EV are a combination of red and blue, the following eight features were used in the input feature vector for classification:

   F1 – HSV_Blue (Total area corresponding to blue in the HSV Color space)
   F2 – HSV_Red (Total area corresponding to red in the HSV Color space)
   F3 – Max Contour Blue (Area of the largest blue contour)
   F4 – Max Contour Red (Area of the largest red contour)
   F5 – Centroid Blue X (X-coordinate of the max blue contour as a ratio to image width)
   F6 – Centroid Blue Y (Y-coordinate of the max blue contour as a ratio to image height)
   F7 – Centroid Red X (X-coordinate of the max red contour as a ratio to image width)
   F8 – Centroid Red Y (Y-coordinate of the max red contour as a ratio to image height)

   The above features were extracted from all the images in the classification dataset and used as a 8*1 feature vector to train the classification models.

Classification Models

The feature vectors obtained by processing the vehicle images were used as input to the classification models trained on the RAVEV Dataset II.1. We compared the performances of some of the most commonly used classifiers, such as SVM, K-Nearest neighbors, and Adaboost, on each of these feature vectors. We also developed a three-stage neural-network classifier and tested the performance of the XGBoost package for EV classification tasks. We list the classification results below. Based on the stream of the tracked objects classified as EVs, we were able to process the data for tasks of object localization, motion planning, etc.

1. ***scikit-learn* Classification models:** The scikit-learn package for Python [17] consists of a wide range of machine learning algorithm implementations for solving supervised and unsupervised problems. We trained some of the object classification models on the RAVEV Dataset II.1. The trained model was then tested on the RAVEV Dataset II.2. The

object classification models from scikit-learn that we evaluated during this study included support vector machines (SVM) [18], K-nearest neighbors (KNN) [19], random forests [20], Adaboost [21] and gradient boosting [22].

2. **XGBoost**: XGBoost is an optimized python implementation of gradient boosted decision tree algorithms designed for high efficiency and performance. XGBoost performs extremely well on most regression and classification tasks.

3. **Neural Network Classifier**: Neural networks have time and again proven to produce the best-in-class results for computer vision applications. We used the Keras [23] library in python to construct a neural network-based binary classifier for classifying EVs and non-EVs. For generating the feature maps, we used a three-layer stack made of 2D convolution layers with ReLU activation followed by max-pooling layers. On top, we used a fully connected ReLU activation layer. We used a single unit sigmoid activation function as the final layer for binary classification into EV or non-EV categories. The neural network architecture is depicted in Figure 3. Plots of accuracy vs epoch and loss vs epoch for the neural network during training on the RAVEV dataset are shown below in Figure 4 and Figure 5. We see that the neural network reaches an accuracy level of about 97% and loss of about 5% after 50 training epochs.

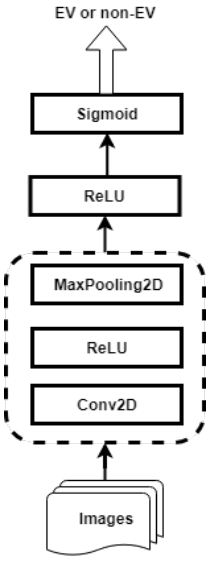

Figure 3. Neural network architecture of EV classifier.

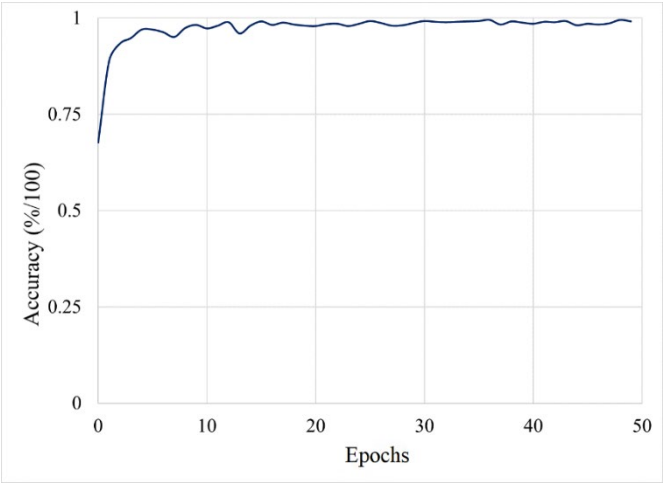

Figure 4. Accuracy vs epochs plot during training the neural network classifier.

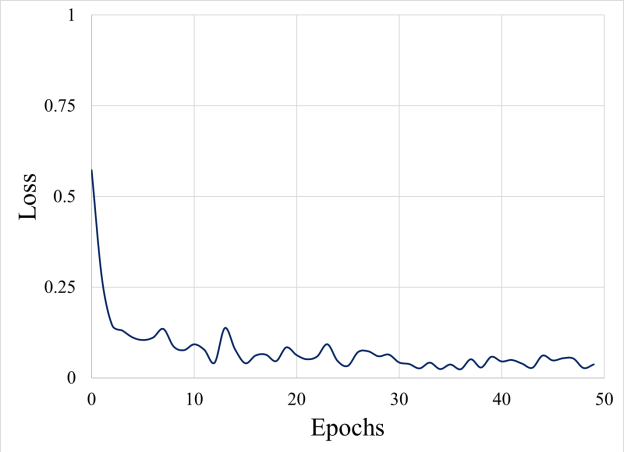

Figure 5. Loss vs epochs plot during training the neural network classifier.

Control Algorithm

In this section, we present the control algorithm that we developed to maneuver the vehicle from its current position to a safe location on the curbside. This algorithm is generic and can also be used in platooning and other emergency scenarios. The main structure of the controller is presented in Figure 6. The input to this controller is a set of reference data for the vehicle to follow. This data could be generated a-priori or be communicated to the autonomous vehicle in real-time. If the data is generated a-priori, its coordinates are transformed relative to the current position of the autonomous vehicle. Prior to discussing the details of the control algorithm, we refer the reader to the Appendix (Nomenclature) for an explanation of all the symbols used in this section.

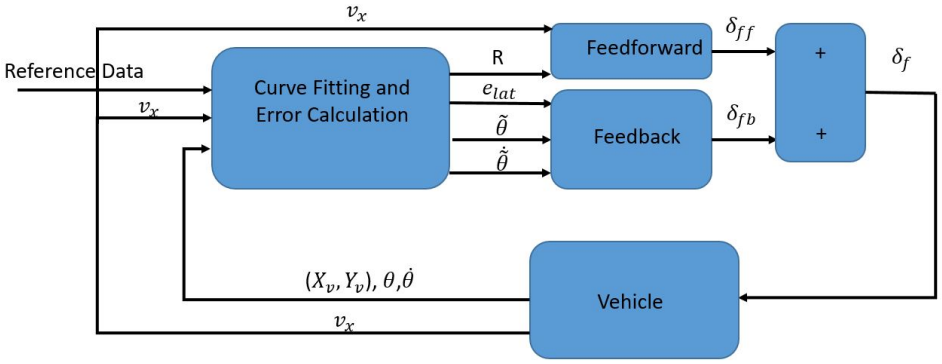

Figure 6. Structure of the controller.

This controller consists of three main parts: curve fitting/error computation block, feedforward block, and feedback block. The curve fitting block uses a collection of arcs and straight lines to accurately fit the given data with a reference trajectory for the vehicle to track. This block also computes the lateral and longitudinal errors, which quantifies the deviation of the autonomous vehicle's position from the reference trajectory. The feedforward block estimates the steering command based on vehicle speed, v_x, and reference path radius of curvature, R. To account for any other disturbance or model uncertainties and initial position errors, we also include a feedback

block to the control, which compensates for the lateral error, e_{lat}, yaw error $\tilde{\theta}$, and yaw rate error, $\dot{\tilde{\theta}}$. In the ensuing discussion, we briefly cover the key aspects of each of the blocks.

Curve Fitting and Error Calculation

A curve fitting algorithm is first used to find the radius of path curvature that the vehicle must track and the error signals used in the feedforward and feedback controller design. The flowchart for this procedure is illustrated in Figure 7.

All the data are separated into two types: a line segment or a curve segment. The data fits a straight line if the deviation of (x_i, y_i) from the line joining (x_1, y_1) and (x_N, y_N) is within a threshold. Otherwise, we find the center (x_c, y_c), radius R of the "least mean square (LMS) fit" circular arc through these points. Using this information, we determine the feedback signals: e_{lat}, $\tilde{\theta}$, and $\dot{\tilde{\theta}}$. An outline of the formulation and the method used to find the circular arc is as follows: given (x_1, y_1), (x_2, y_2) , …, (x_N, y_N), the problem is to 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 is minimized.

$$J = \sum_{i=1}^{N} (R^2 - (x_i - x_c)^2 - (y_i - y_c)^2)$$

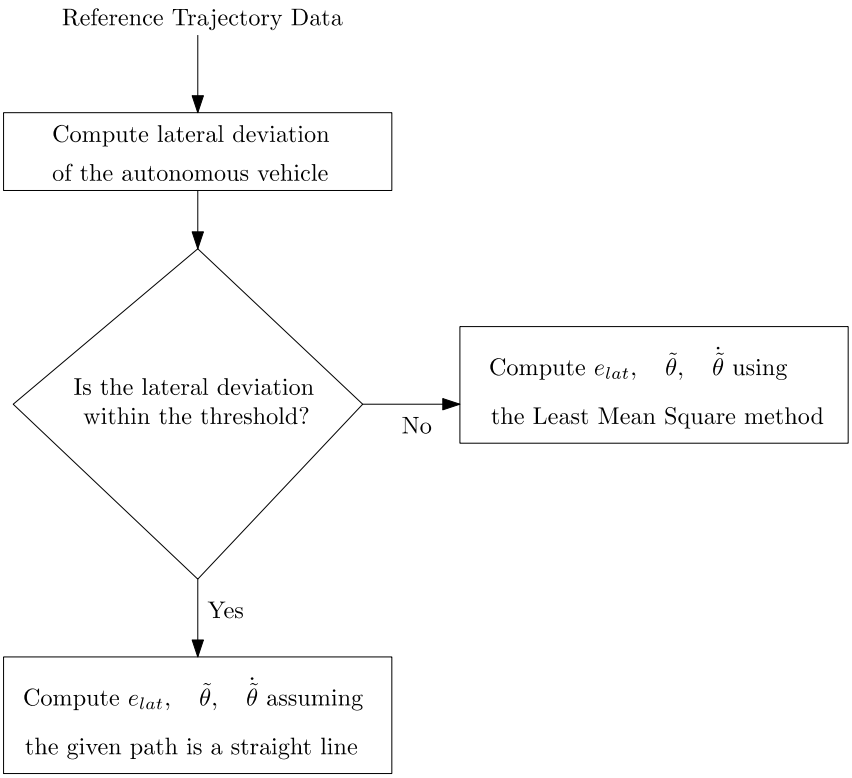

Figure 7. Curve fitting and error calculation.

The reasons for choosing the least square form of the error are as follows: first, when the points in the data are exactly on the circular arc, the error is zero. Second, this form is relatively easier to optimize, as the partial differentials with respect to the center (x_c, y_c) and radius R are easy to compute. As the curve fitting is implemented in real time, this formulation enabled us to compute the parameters quickly.

An illustration of the errors is shown in Figure 8 and Figure 9. If the given data is an arc, the lateral error is $e_{lat} = R - \sqrt{(X_v - X_c)^2 + (Y_v - Y_c)^2}$. If the given data fits a straight line better, the lateral error is $e_{lat} = \frac{y_v - mx_v - c}{\sqrt{1+m^2}}$, where y = mx + c is the equation of the straight-line. The yaw error and yaw rate error can be represented as $\tilde{\theta} = \theta - \theta_R$, $\dot{\tilde{\theta}} = \dot{\theta} - \frac{v_x}{R}$.

Once the error signals and radius of curvature for path are derived, we design the feedforward and feedback controller for the autonomous vehicle. Prior to that, we describe the vehicle model and the assumptions we used to derive the equations.

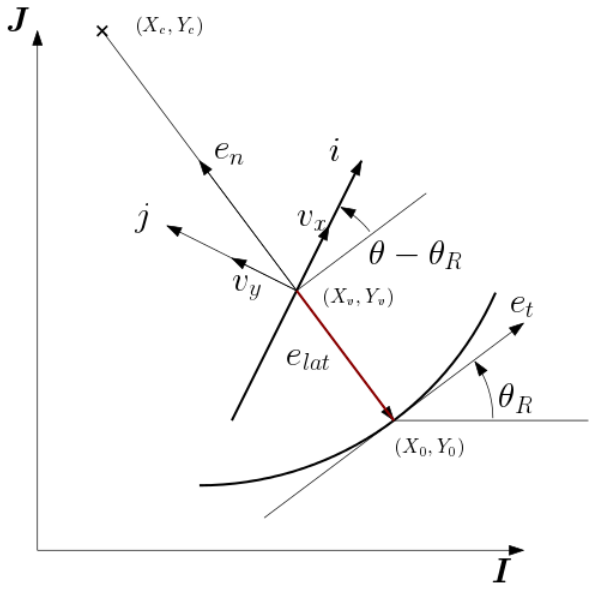

Figure 8. Illustration of lateral error for an arc.

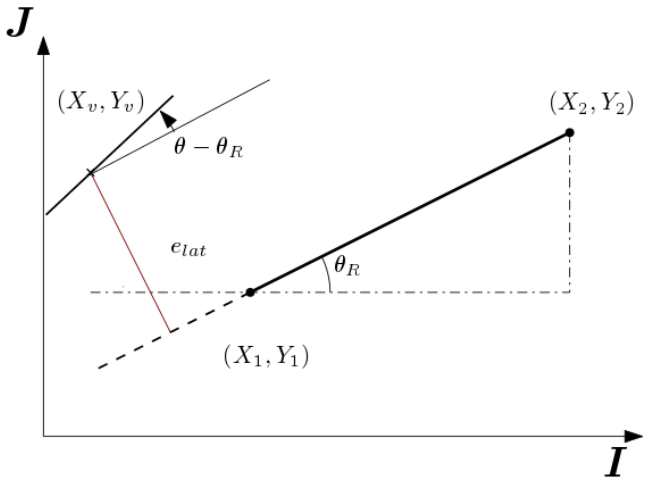

Figure 9. Illustration of lateral error for a straight line.

Feedforward and Feedback Controllers

Vehicle model

In this work, the autonomous vehicle is modeled as a bicycle (refer to Figure 10). The following assumptions must be followed for using this vehicle model:

- The radius of turn, R, is far larger than wheelbase L.
- The left and right steer angle must be approximately the same.

- The side slip angle of front wheels α_f is equal, as is the slip angle of rear wheel side slip angle α_r.
- Side slip angles are small: $\alpha_f \approx \delta_f - \left(\frac{v_y + a\frac{d\theta}{dt}}{v_x}\right)$ $\alpha_r \approx -\left(\frac{v_y - b\frac{d\theta}{dt}}{v_x}\right)$.
- Linear Model for Cornering Forces: $F_r = C_r\alpha_r$ $F_f = C_f\alpha_f$.

The equations of motion of the vehicle using the bicycle model can be described as:

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

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

where m, I, a and b are the vehicle mass, inertia and distance from the center of mass to vehicle front and rear tire, α_f and α_r are side slip angles of tires & C_f and C_r are the cornering stiffness of vehicle.

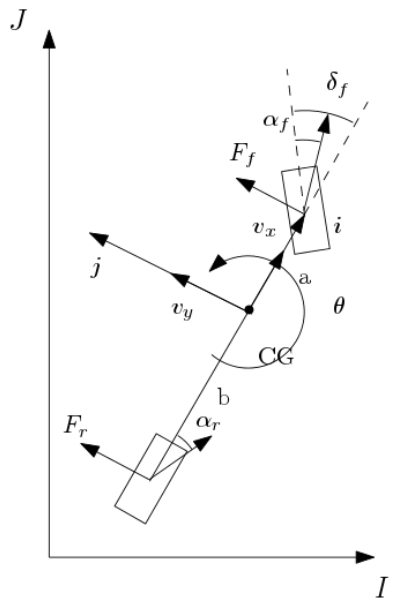

Figure 10. Bicycle model used solve dynamics of the autonomous vehicle.

Controller Design

The feedforward part provides a control command, δ_{ff}, based on the vehicle speed and previewed path's curvature. At a constant speed, no initial error and no disturbances/model uncertainties, the vehicle should track the circular arc without any error. However, errors are common and inevitable due to modeling assumptions and practical uncertainties. To handle these errors, a feedback controller is also required.

The feedback controller contributes control command, δ_{fb} , based on the lateral error, e_{lat}, yaw error, $\tilde{\theta}$, and yaw rate error, $\dot{\tilde{\theta}}$. The summation of feedback and feedforward is the final control input to the vehicle. The feedforward, feedback and final control command, δ_f, can be described as:

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

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

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

Results

Results of Vision-Based Identification of EVs

Performance of Framework 1

Figure 11 shows an image with EV detections made by YOLOv3 using the trained weights. Table 1 contains the confusion matrix for the predictions made by the trained YOLOv3 detector on a sequence of 13,429 detections from video sequences.

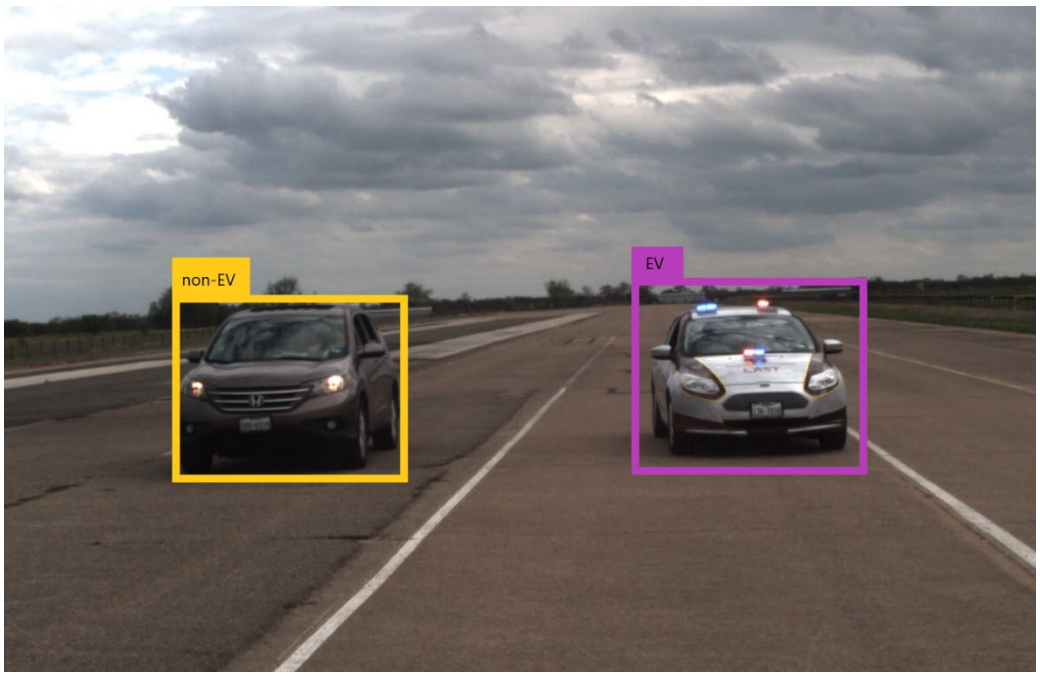

Figure 11. Detection outputs from YOLOv3 implemented on the RAVEV Dataset I.

Table 1. Confusion Matrix for 2 Class Classification Corresponding to 13,429 Detections in 4 Video Sequences

Object Class		Predicted	
		non-EV	EV
Actual	non-EV	2610	460
	EV	1856	8503

Performance of Framework 2

We first quantify the classification results of Framework 2 in terms of accuracy, precision and recall scores to help us select the best classifier. By definition,

$$Accuracy\ Score = \frac{TP + TN}{TP + FP + TN + FN} = \frac{Total\ correct\ prediction}{Total\ Predictions}$$

$$Precision\ Score = \frac{TP}{TP + FP} = \frac{Total\ Positive}{Total\ Predicted\ Positive}$$

$$Recall\ Score = \frac{TP}{TP + FN} = \frac{Total\ Positive}{Total\ Actual\ Positive}$$

where,

TP = True Positive
FP = False Positive
TN = True Negative
FN = False Negative

Accuracy, precision and recall values of each of the classification models against the different feature vectors are listed in Table 2.

Table 2. Classification Results (in %) of Different Classification Models Against the Feature Vectors

Feature	Score	SVM	Adaboost	Random Forrest	Gradient Boosting	XGBoost	KNN-3
HOG	*Accuracy*	58.02	99.51	95.06	95.31	97.53	76.79
HOG	*Precision*	53.3	99.48	96.77	93.10	97.42	67.48
HOG	*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	92.48
Color Histogram	*Precision*	47.90	99.48	98.96	96.92	100	89.10
Color Histogram	*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
Pixel Array	*Precision*	100	96.88	97.83	93.85	99.47	100
Pixel Array	*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
Custom Array	*Precision*	95.50	86.36	85.71	85.55	90.78	82.74
Custom Array	*Recall*	53.00	93.25	92.02	90.80	93.5	85.28

Overall, XGBoost and Adaboost performed better than the other classification models on the tested data. This can be attributed to the fact that, during training, boosting algorithms, unlike neural networks and SVMs, combine many relatively weak learners to create a highly accurate prediction rule, reducing the curse of dimensionality and potentially improving execution time. SVM inherently performs poorly on high dimensional vectors and we can see a similar result with HOG and color histogram feature vectors. XGBoost performs better than the scikit-learn implementation of gradient boosting when trained on RAVEV dataset for EV classification. XGBoost performs best with color histogram/pixel array, whereas HOG descriptor works best on Adaboost. The custom array that the research team defined has a worse performance rate compared to the other feature vectors.

In addition to the above classifiers, we also tested the neural network classifier directly on the pixel array, since finding the HOG and color histogram features required computation times in the order of a second. The results from the neural network classifier provided an accuracy of 98.57%, precision of 96.89%, and recall of 97.35%. In addition, the neural network classifier in combination with the pixel array ran approximately three times faster than the XGBoost classifier in combination with the color histogram or other feature vectors. Therefore, for real-time implementation, we used the neural network classifier on the pixel array to differentiate between an EV and a non-EV. Once an EV was identified by the vision software onboard the autonomous vehicle, the control algorithms were invoked to safely park that vehicle.

Performance of the Control Algorithms

The combined sensing and control algorithms were tested multiple times at the Texas A&M Rellis Campus. The results from a commonly repeated test of lane changing and parking when an EV was detected are discussed below. More details on the algorithm and its implementation at multiple speeds can be found in [24].

The test vehicle was autonomously driven in a straight line at 30 mph (13.4 m/s). An EV was driven behind this autonomous vehicle without triggering the emergency lights. After a brief period, the lights in the EV were turned on. On detecting the EV using the vision algorithm presented in the previous section, the controller slowed the test vehicle and followed a lane changing path. The autonomous vehicle was finally stopped at one lane right of the original path. The original path and lane change path are shown in Figure 12.

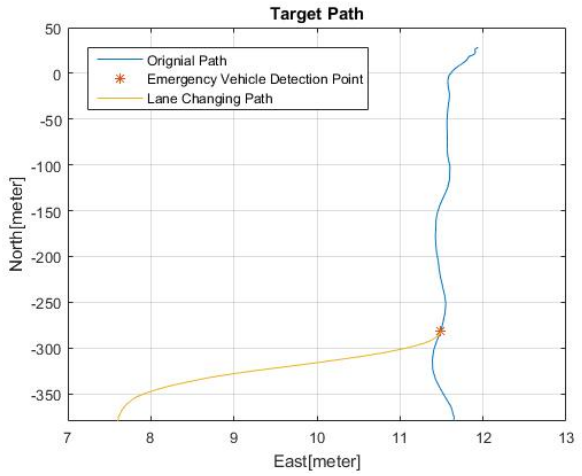

Figure 12. The original and lane changing path with detection point.

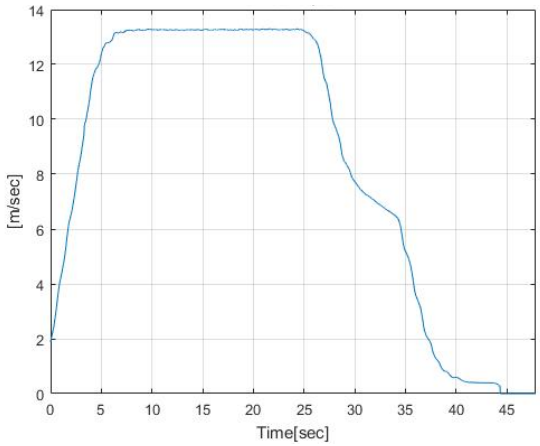

Figure 13. Vehicle speed variations during the lane change maneuver.

During the 26th second of the test, the autonomous vehicle detected the EV. The autonomous vehicle's speed then decreased with time and the vehicle moved one lane to the right (to the safety zone). Slowly, the vehicle finished the lane changing maneuver and came to a complete stop. Vehicle speed variation with time is displayed in Figure 13. The vehicle's lateral error along the lane changing path is shown in Figure 14. The maximum error occurred when the vehicle was undergoing the lane changing maneuver, at around 0.55 meters. The lateral error then decreased with time to around 10 centimeters. The vehicle yaw error and yaw rate error are illustrated in Figure 15 and Figure 16, which are all bounded in -0.05~+0.01 rad and -0.08~+0.06 rad/sec. This implies that the vehicle was not only following the lane changing path but was also within reasonable margins of error with respect to its heading angles.

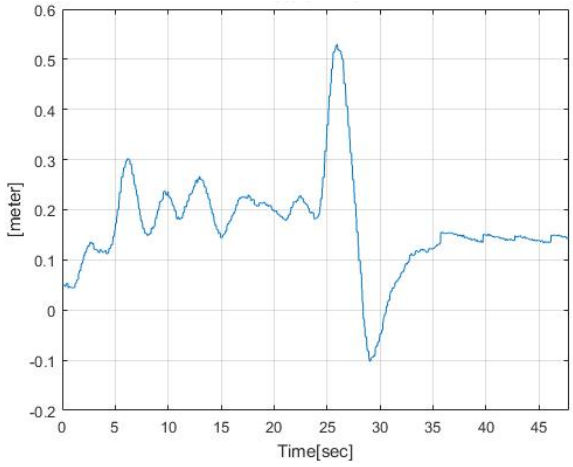

Figure 14. Lateral error measurements during the lane change maneuver.

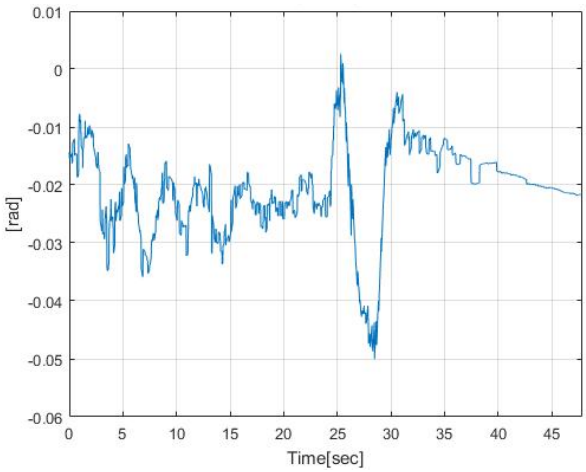

Figure 15. Yaw error measurements during the lane change maneuver.

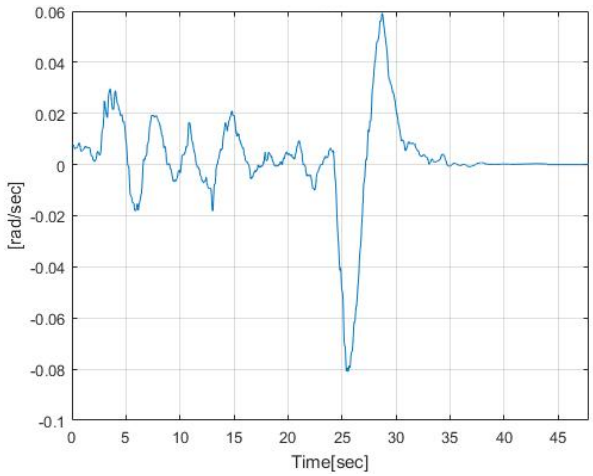

Figure 16. Yaw rate error measurements during the lane change maneuver.

Conclusions and Recommendations

With respect to control algorithms, our proposed approach is generic and can be extended to other applications where vehicles may have to follow trajectories and also avoid obstacles. The main contribution of our methods to the body of transportation research is in the formulation of the problem and in the development of a systematic procedure for identifying the gains of the controllers [24]. This approach is currently being extended to platooning applications.

The vision framework that worked the best for identifying EVs involved two steps. In the first step, we identified all vehicles and tracked them using YOLOv3 and SORT algorithms. In the second step, a classifier was used to identify whether each tracked vehicle was an EV or a non-EV. We also found that using a neural network-based classifier worked well for real-time implementations in our autonomous test vehicle.

The work presented in this report can be extended in several directions to include more realistic scenarios encountered in practice. For example, the parking operation of the autonomous vehicle could also use information from LiDAR and other sensors to avoid obstacles in real-time. There may also be scenarios where the brake lights from an EV may be similar in color (red) to the EV's emergency lights. One way to address this issue is to also use the location of the emergency lights on an EV as an input during classification. For nighttime conditions, we observed that the developed algorithms did not perform well primarily because identifying the vehicles and tracking them was challenging in and of itself. Though we also did some preliminary work on identifying EVs based on sound data, we observed that the sound signals had more noise in urban settings and led to many false positives/negatives. A better alternative would be to fuse the communicated data (typically broadcasted by the EVs) with vision data.

Additional Products

The Education and Workforce Development (EWD) and Technology Transfer (T2) products created as part of this project can be downloaded from the Safe-D website here. The final project dataset is located on the Safe-D Collection of the VTTI Dataverse.

Education and Workforce Development Products

Graduate student Abhishek Nayak's M.S. thesis on vision algorithms for identifying emergency vehicles was completely supported by this project. Nayak received his M.S. degree in Fall of 2019. A significant part of doctoral student Mengke Liu's Ph.D. dissertation of on the development of control algorithms for parking the car was funded by this Safe-D project. The following other graduate students also worked part-time or for a short span conducting the demonstrations during the project: Kenny Chour, Nishat Mehta, Saipraneeth Devunuri, Mani Deep Ankem, Aditya Gujrar.

It is expected that the data collected during this project will be used in a course module for a Math course taught at Texas A&M University in Fall 2020. This module will be made available to the public via the project page on the Safe-D website.

Technology Transfer Products

The following paper was published by graduate student Abhishek Nayak as part of his M.S. thesis. His thesis work was completely supported by this project.

Nayak, Abhishek, Swaminathan Gopalswamy, and Sivakumar Rathinam. *Vision-Based Techniques for Identifying Emergency Vehicles*. No. 2019-01-0889. SAE Technical Paper, 2019. https://www.sae.org/publications/technical-papers/content/2019-01-0889/

The following paper was published by graduate student Mengke Liu as part of his Ph.D. dissertation. His dissertation work was partly supported by this project.

Liu, Mengke, Sivakumar Rathinam and Swaroop Darbha. *Lateral Control of an Autonomous Car with Limited Preview Information*. European Control Conference, 2019. https://controls.papercept.net/conferences/conferences/ECC19/program/ECC19_ContentListWeb_4.html#fra7_06

The following poster was also presented at multiple conferences throughout the course of the project.

Poster presentations:

Nayak, A., Rathinam, S., Gopalswamy, S, and Chrysler, S.T. (2019). *RAVEV - Response of Autonomous Vehicles to Emergency Vehicles.* Poster presented at the CSCRS Safe Systems Summit, Durham NC, Apr 23-24, 2019.

Nayak, A., Devanuri, S., Liu, M., Rathinam, S., Gopalswamy, S, and Chrysler, S.T. (2019). *RAVEV - Response of Autonomous Vehicles to Emergency Vehicles.* Poster presented at the 3rd Annual Texas A&M Transportation Technology Conference, College Station TX, May 8, 2018.

Nayak, A., Rathinam, S., Gopalswamy, S, and Chrysler, S.T. (2019). *RAVEV - Response of Autonomous Vehicles to Emergency Vehicles.* Poster presented at the 3rd Annual Texas A&M Transportation Technology Conference, College Station TX, April 30, 2018

Nayak, A., Rathinam, S., Gopalswamy, S, and Chrysler, S.T. (2019). *RAVEV - Response of Autonomous Vehicles to Emergency Vehicles.* Poster presented at the Texas Mobility Summit Demo Day, October 28, 2018.

PIs Dr. S. Rathinam and Dr. S. Gopalswamy also presented a summary of the demonstrations performed in this project at the Autonomous Cars Conference, Brookings Institution, Washington D.C. on July 25, 2019.

Data Products

The following video and audio datasets used for training and testing the algorithms have been uploaded to the Safe-D Dataverse and can be found at https://doi.org/10.15787/VTT1/IVNW9L [25]. They can also be accessed through the project website at https://sites.google.com/tamu.edu/ravev/research-data:

- Two Image Datasets that were used to develop and train EV identification and classification algorithms
    1. RAVEV Dataset I for vision-based detection: 1,000 images annotated with 3 classes – EV, non-EV and pedestrians.
    2. RAVEV Dataset II for vision-based classification: An annotated dataset containing cropped images of EV and non EV divided into two parts—a training set containing 1,000 images and a validation set containing 350 images of each class.

References

[1] "Texas Transportation Code, Title 7, Subtitle C, Section 545.156 (Year 1995)".

[2] Y. Xing, C. Lv, L. Chen, H. Wang, H. Wang, D. Cao, ... and F. Y. Wang, "Advances in vision-based lane detection: algorithms, integration, assessment, and perspectives on ACP-based parallel vision.," *IEEE/CAA Journal of Automatica Sinica,* pp. 5(3), 645-661, 2018.

[3] "Texas Transportation Code, Title 7, Subtitle C, Section 547.305 (Year 1995)".

[4] "Texas Transportation Code, Title 7, Subtitle C, Section 547.702 (Year 1995)".

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

[6] 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.," *16th European Signal Processing Conference,* pp. (pp. 1-5). IEEE., August 2008.

[7] O. Karpis, "System for vehicles classification and emergency vehicles detection.," *IFAC Proceesings Volumes 45(7),* pp. (pp. 186-190), 2012.

[8] S. Gopalswamy and S. Rathinam, "Infrastructure enabled autonomy: A distributed intelligence architecture for autonomous vehicles," *IEEE Intelligent Vehicles Symposium (IV),* pp. (pp. 986-992). IEEE., 2018, June.

[9] 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.

[10] D. Ravipati, K. Chour, A. Nayak, T. Marr, S. Dey, A. Gautam and .. &. S. G. , "Vision Based Localization for Infrastructure Enabled Autonomy.," *IEEE Intelligent Transportation Systems Conference (ITSC),* pp. (pp. 1638-1643). IEEE., 2019 October.

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement.," *arXiv preprint arXiv:1804.02767.,* 2018.

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

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection.," *IEEE computer society conference on computer vision and pattern recognition,* pp. (Vol. 1, pp. 886-893). IEEE., 2005, June.

[14] P. Rybski, D. Huber, D. Morris and R. Hoffman, "Visual classification of course vehicle orientation using a histogram of oriented gradients features.," *Intelligent Vehicles Symposium (IV),* pp. (pp. 921-928), 2010.

[15] O. Chapelle, P. Haffner and V. N. Vapnik, "Support vector machines for histogram-based image classification.," *IEEE transactions on Neural Networks, 10(5),* pp. 1055-1064, 1999.

[16] G. Bradski and A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library., O'Reilly Media, Inc., 2008.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg and J. Vanderplas, "Scikit-learn: Machine learning in Python.," *Journal of machine learning research,* pp. 2825-2830, 2011.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning, 20(3),* pp. 273-297, 1995.

[19] N. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression.," *The American Statistician, 46(3),* pp. pp.175-185, 1992.

[20] L. Breiman, "Random forests.," *Machine learning, 45(1),* pp. pp.5-32, 2001.

[21] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting.," *Journal of computer and system sciences, 55(1),* pp. pp.119-139., 1997.

[22] T. Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection.," in *IEEE international conference on computer vision*, 2017.

[23] F. Chollet, *Keras,* 2015.

[24] M. Liu, S. Rathinam and S. Darbha, "Lateral Control of an Autonomous Car with Limited Preview Information.," in *18th European Control Conference (ECC)*, 2019, June.

[25] A. Nayak, S. Rathinam and S. Gopalswamy, "Response of Autonomous Vehicles to Emergency Response Vehicles (03-051)," 2019. [Online]. Available: https://doi.org/10.15787/VTT1/IVNW9L.

[26] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, "Vision meets robotics: The kitti dataset.," *The International Journal of Robotics Research, 32(11),* pp. 1231-1237, 2013.

[27] A. Nayak, S. Gopalswamy and S. Rathinam, "Vision-Based Techniques for Identifying Emergency Vehicles," *SAE Technical Paper,* pp. (No. 2019-01-0889)., 2019.

Appendix

Nomenclature

Symbol	Description
v_x	Lateral velocity of the autonomous vehicle
v_y	Longitudinal velocity of the autonomous vehicle
X_v	X position coordinate of the autonomous vehicle
Y_v	Y position coordinate of the autonomous vehicle
θ	Heading angle of the autonomous vehicle
$\dot{\theta}$	Yaw rate of the autonomous vehicle
R	Radius of curvature of the desired path
$\tilde{\theta}$	Yaw error
$\dot{\tilde{\theta}}$	Yaw error rate
θ_R	Desired heading angle
e_{lat}	Lateral error of the vehicle
e_t , e_n	Unit vectors of the path reference frame
(X_c,Y_c)	Center of curvature of the reference path