

Vehicle Detection

Jacob Lowe, Jonathan Huggins, Abdul Hadi Imran

Abstract:

As Artificial Intelligence technology continues to evolve, it is no surprise that there is a growing demand for an increased level of autonomy in vehicles. One of the primary challenges associated with developing a fully self-driving car is that it needs to be capable of detecting other vehicles in its vicinity. The goal of this project is to train a machine-learning model that can successfully identify whether an input image contains a car or not. As part of our approach, we used techniques such as Feature detection and Model Training, which are discussed in detail in the section below. Overall, we found that our final implementation delivered on design objectives we sought to accomplish.

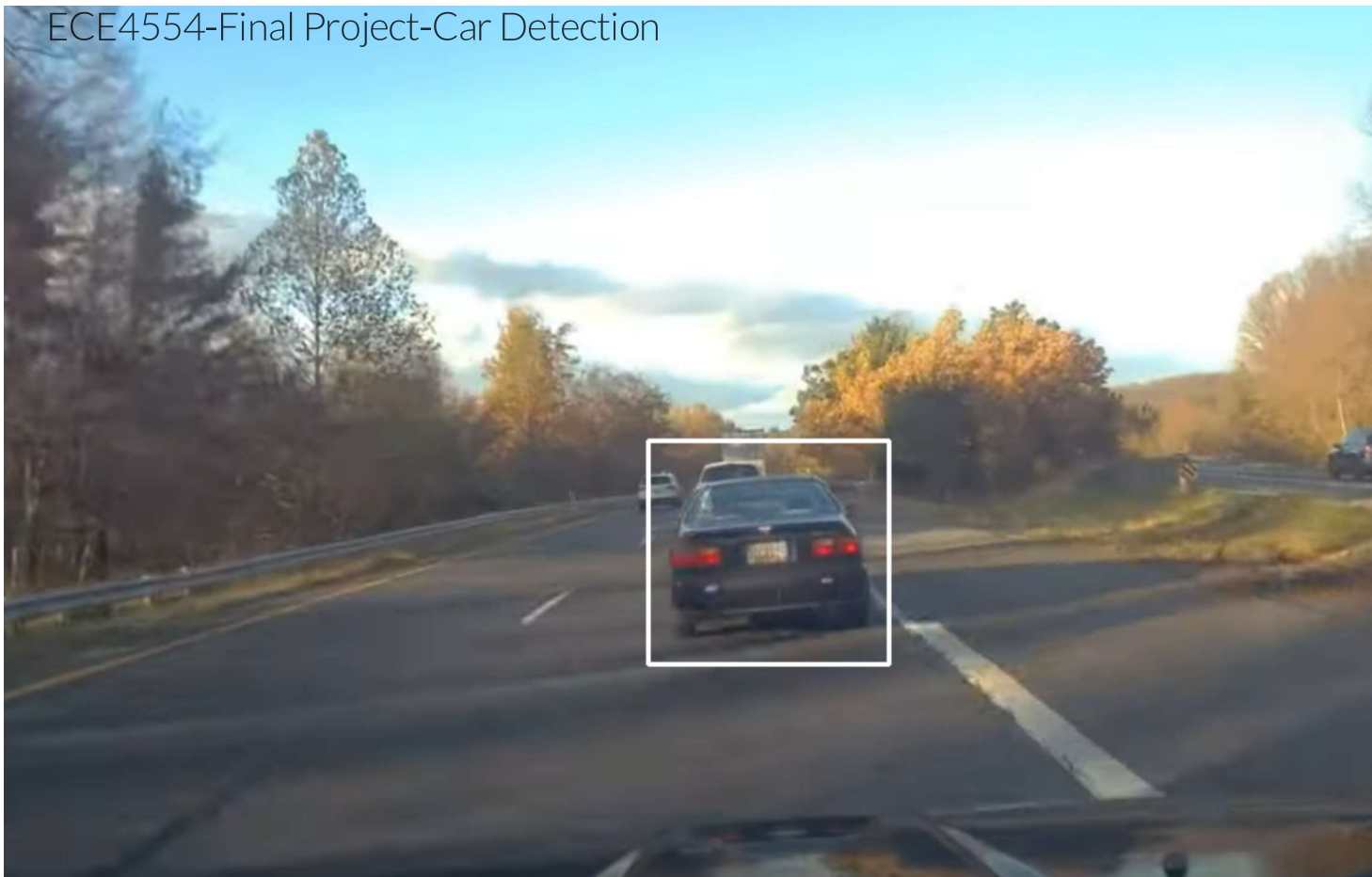


Figure 1: Example of the Final output of the algorithm.

Introduction

ECE4554-Final Project-Car Detection

Object detection in general has a great scope in technological applications moving forward. One of its specific sub-topics - vehicle detection is integral to the successful development of Level 4 and Level 5 autonomous vehicle systems (Cars with High - Complete Driving automation).

Approach:

Our approach is broken down into a number of steps, namely pre-processing, Feature Detection using HOG, the use of Linear SVC for model training, and bounding box application, as can be seen in the process diagram below.

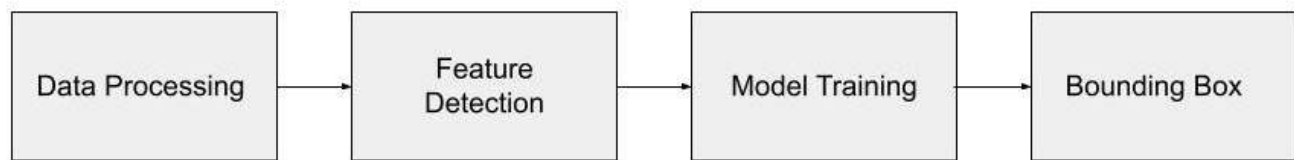


Figure 2. The process diagram of the vehicle detection algorithm

Pre-Processing Steps:

First all the images in the data set are imported into a directory. Initially, the given dataset is divided into a testing and training dataset. This separation is done randomly to ensure that the model does not use the same images from the data set every time it is instantiated.

Feature Detection (HOG):

For feature detection, the Histogram of Oriented Gradients (HOG) algorithm is applied to each input image in both the training and testing datasets. After resizing the input image to a 128x64 and converting it to grayscale, the algorithm computes each pixel's gradient in both the x and y directions, using these to calculate a magnitude and phase value for each image pixel. The magnitude and phase matrices obtained can then be used to find a 9-bin histogram for each 8x8 cell in the 128x64 image. A bin size of 20 can be used to sweep through every angle from 0° to 180° and keep track of the frequency with which each orientation value appears in the image. After normalization, the HOG computed image is returned as shown in the following images.



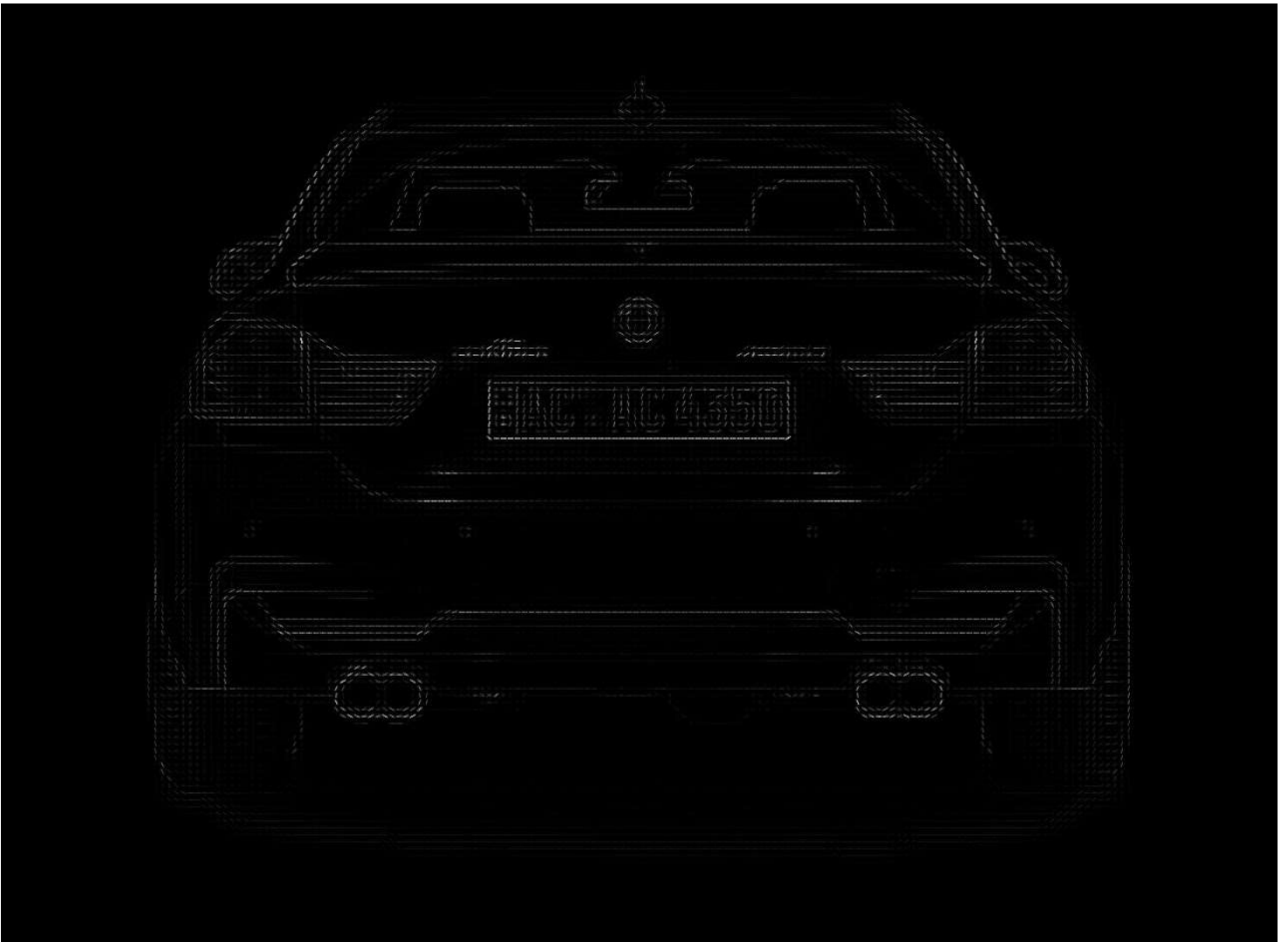


Figure 3. A sample image of the back of a car (left) and its computed HOG output (right)



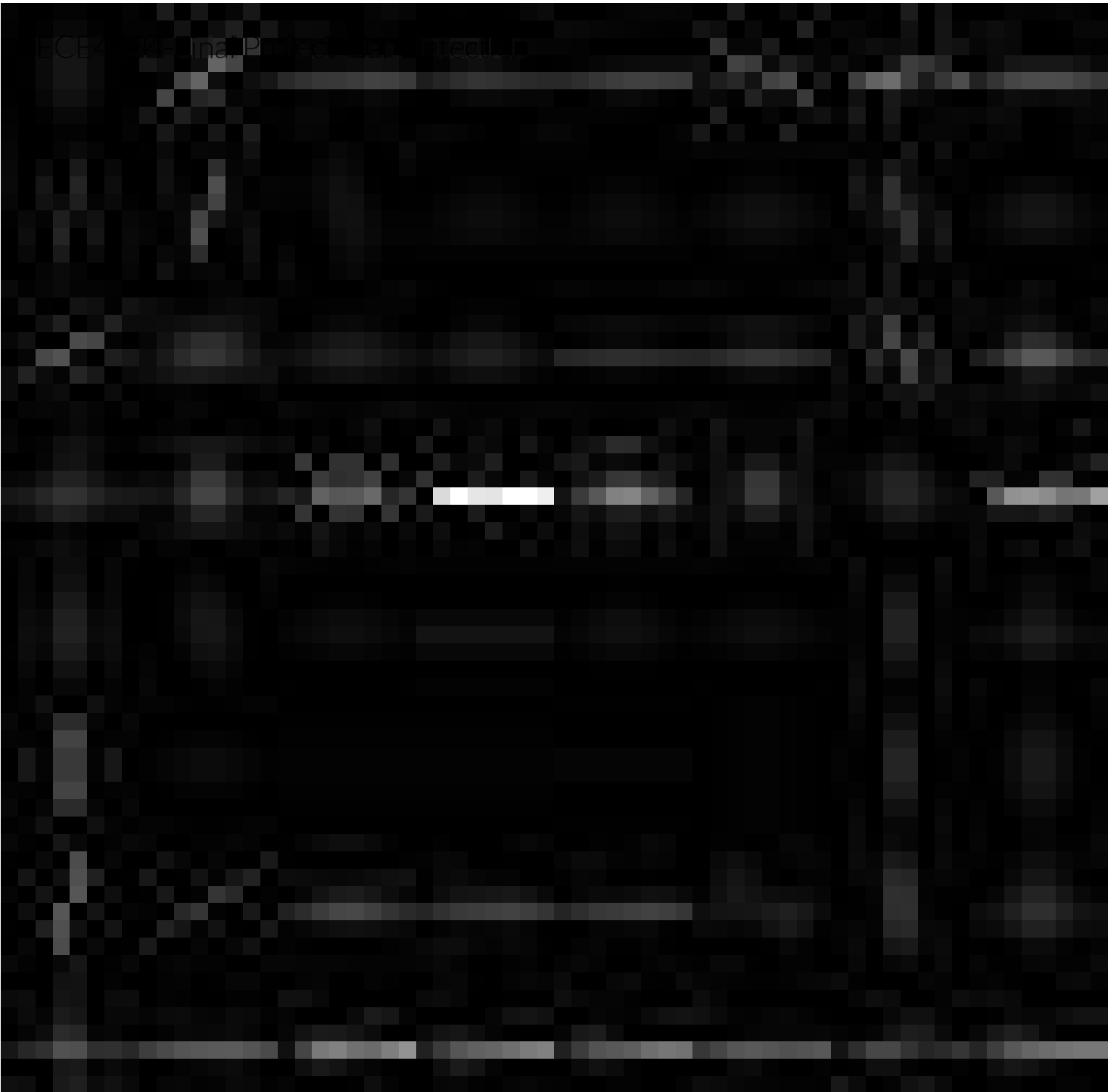


Figure 4. A sample image of the back of a car (left) and its computed HOG output (right) from the data set used to train the SVC. Each image in the data set is 64x64.



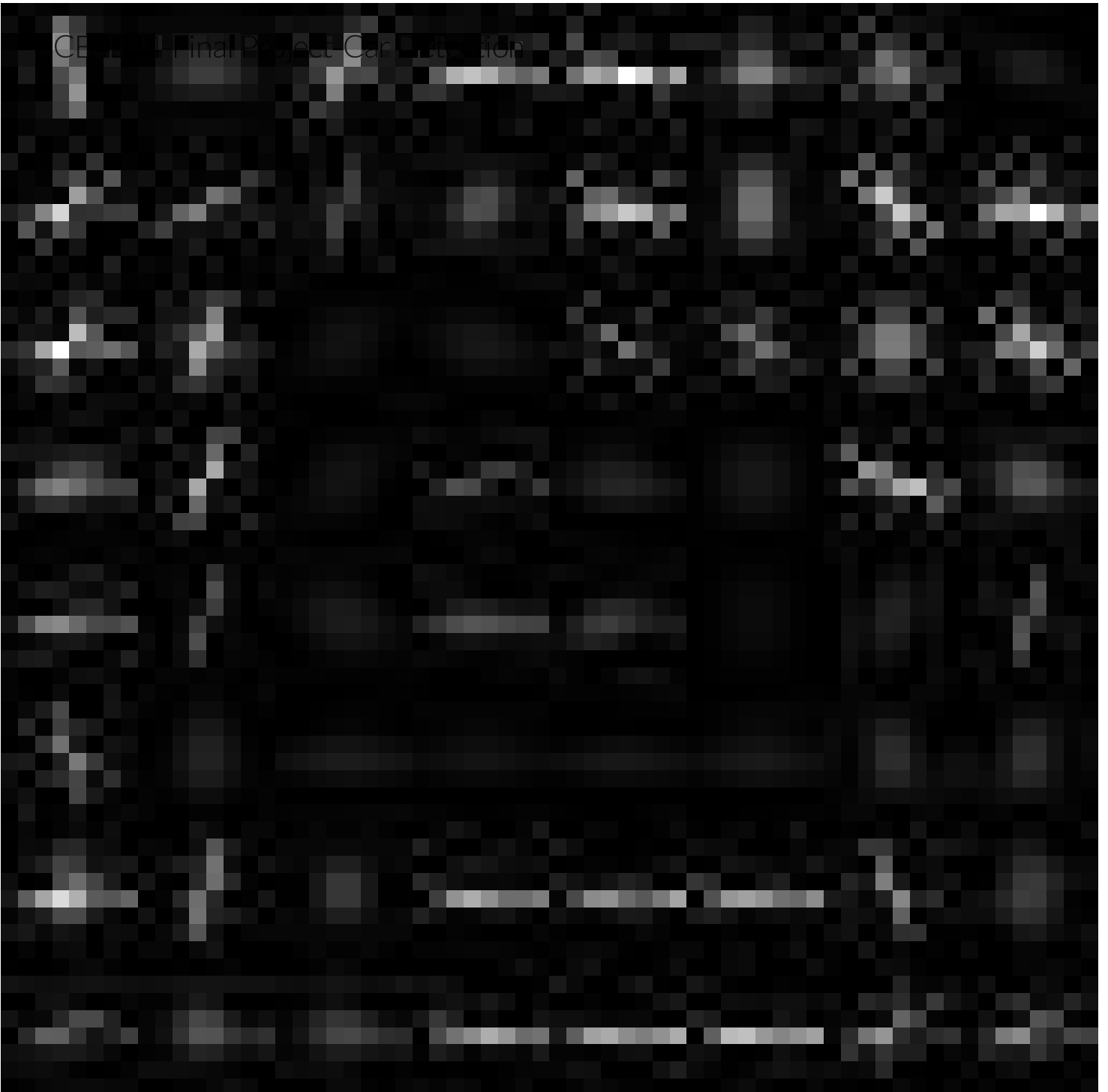


Figure 5. Another sample image (left) and its computed HOG (right).

Linear Support Vector Classification (SVC) Training:

Once a HOG result has been computed for each image, the training data set and a list indicating the desired prediction are passed to the SVC. Our team used the 'sklearn' library in Python to implement the SVC, which trains a linear SVC model using the provided dataset. This model can be tuned using the given parameters. Once trained, the model can successfully classify images to be 'vehicle' or 'non-vehicle' images. This result can be verified using the prediction passed into the Linear SVC and used to compute overall vehicle detection accuracy for the testing dataset.

Bounding Box application:

Upon receiving a trained model in the form of a linear SVC, input data in the form of dashboard camera footage can be passed in. The model then processes the frame to determine whether a car is present. If so, the bounding box algorithm moves through the image to detect the back of the car, drawing a bounding box around it. It does so by using image contours to select a box that is both within the provided constraints and lies horizontally flat - like a license plate or the back of a car. After looking into several methods to create a bounding box around the car, we decided this method was much more efficient than using a sliding window approach - the use of a sliding window to move through the image and manually look for the location of the car.

Bounding Box to SVC pipeline:

After all potential license plates in an image have been found, a larger bounding box will be drawn around each potential license plate. The part of the image in this new bounding box will be cropped and resized into a 64x64-sized image. It will then be HOG filtered and passed into the model to predict whether a car is present in the frame or not. Once the model returns a value of 1 (meaning it thinks that the image given is a vehicle) it will keep that bounding box in the final image.

ECE4554-Final Project-Car Detection Experiments and Results:

Training and Testing Datasets:

The datasets we used to train and test our linear SVC were drawn primarily from vehicle image (417) and non-vehicle image (3483) datasets obtained from [2]. For our total data set, we randomly selected 50% of the vehicle images in the 'GTI_Far' folder and an equivalent number of non-vehicle images.

To obtain the training and testing datasets, we simply divided the existing total dataset into an 80-20 split after randomly shuffling it.

Feature Detection (HOG) parameters:

For feature detection, we tuned several parameters - Orientations, pixels per cell, and cells per block. We used values of (8,8) for both pixels per cell and cells per block, as we found that these were the best values to optimize feature detection while minimizing the loss of information. For the number of orientations, we chose to go with 9, so as to account for angles between 0° and 180° using 20° increments (9 bins).


Linear Support Vector Classification (SVC) parameters:

The combined vehicle datasets of cars are then split into X and Y datasets respectively, the X dataset being the training vectors and the Y dataset being the class labels of the combined images. These X and Y training datasets are then inputted into the Linear SVC to fit the model and start training. Once the model is fitted, it uses the X testing dataset for prediction and the predicted Y label results for comparison - allowing us to then measure overall model accuracy.

Bounding Box parameters:

The width_min, width_max, height_min, and height_max dictate the size of the bounding box boundaries. After displaying every contour's bounding box, we noticed that both bounding boxes inside the car and those that detected the license plate were very long horizontally and not so much vertically. The parameter width_min was set to 20, and the width_max was chosen as a sizeable arbitrary 500 (this can be tuned to be smaller). For rectangle heights, 0-50 are chosen to detect horizontal lines of the back of a car or plates.

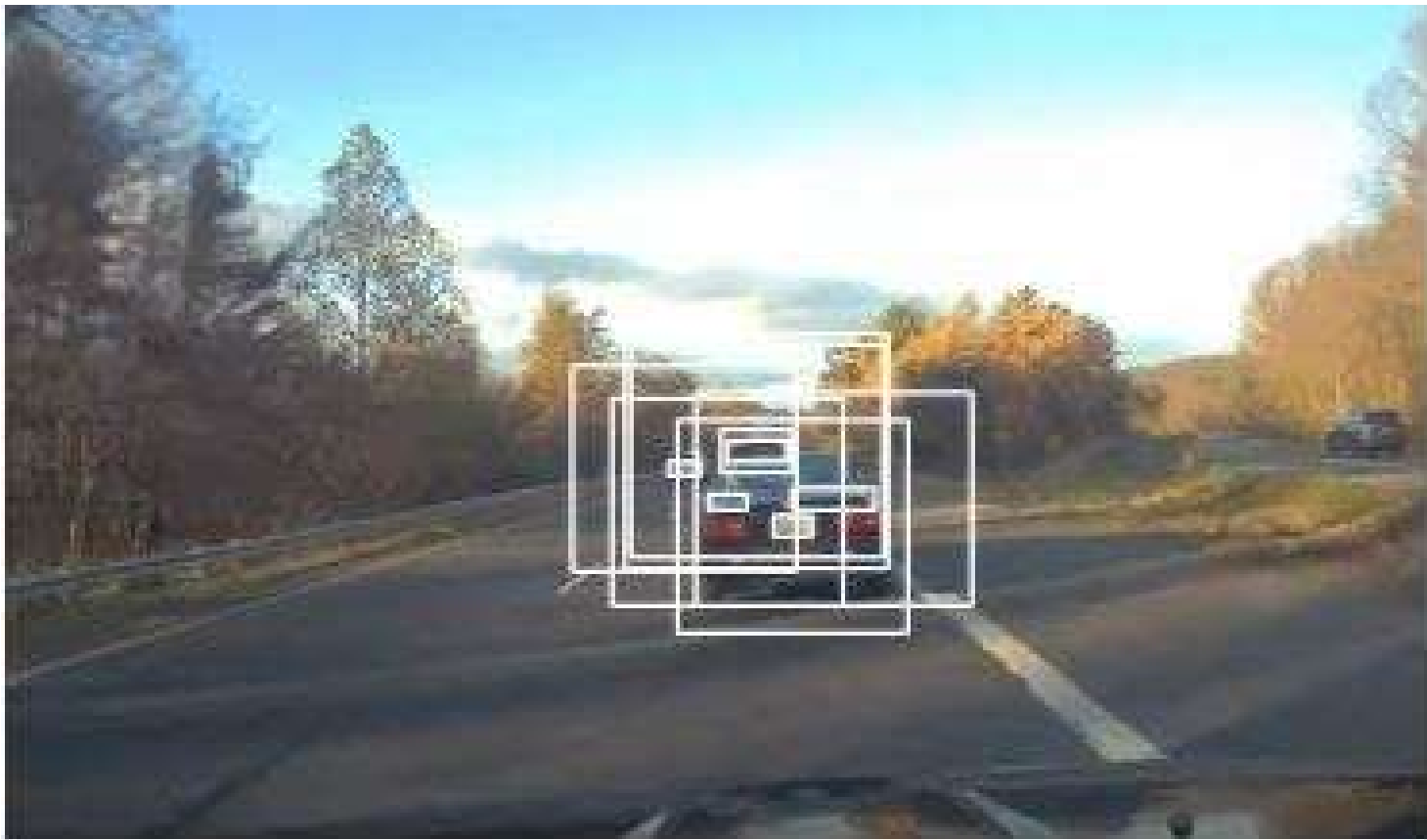
The offset, center_y_min, and center_y_max variables focus on selecting boxes closer to the center front of the car lane. The offset variable grows the bounding box by 64 for each adjacent corner in the rectangle. As a result, most of the car is moved within the bounding box. The center_y_min and center_y_max parameters used are 4/7 and 1/4 respectively. 4/7ths of the y portion of the image (sky, far lane) and 1/4th of the bottom part (hood of the car) of the image are then cropped out. The x portions of these parameters can be tuned manually to be at the center of the image by simply dividing by half.

^A  her parameter used in the bounding box code is the offset value. This value is used to draw a larger bounding box around each potential license plate. For testing, the offset was set to a constant 64 to ensure that

the image was always at least 64x64 in size.
ECE4554-Final Project-Car Detection

Iteration	Low	High	Mean
1	80.96%	86.16%	81.47%
2	80.96%	85.69%	81.47%
3	80.98%	86.44%	81.47%
4	80.92%	86.97%	81.56%
5	80.95%	85.93%	81.47%

Figure 6. Table Displaying the high, low, and avg. Accuracy for running the SVC model over random images in the data set (1000 shuffles/splits per iteration). This test can be found in the project code.



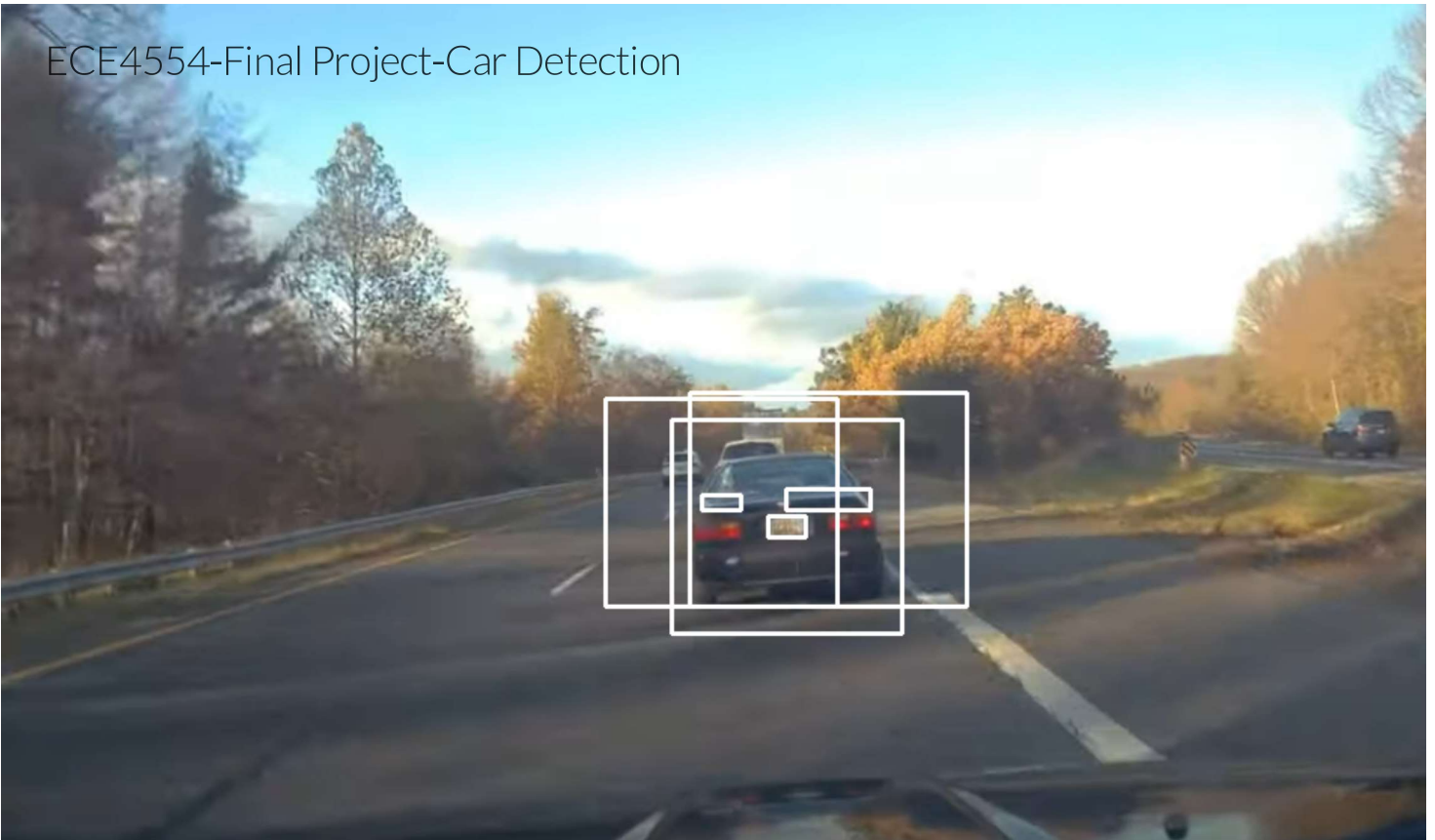
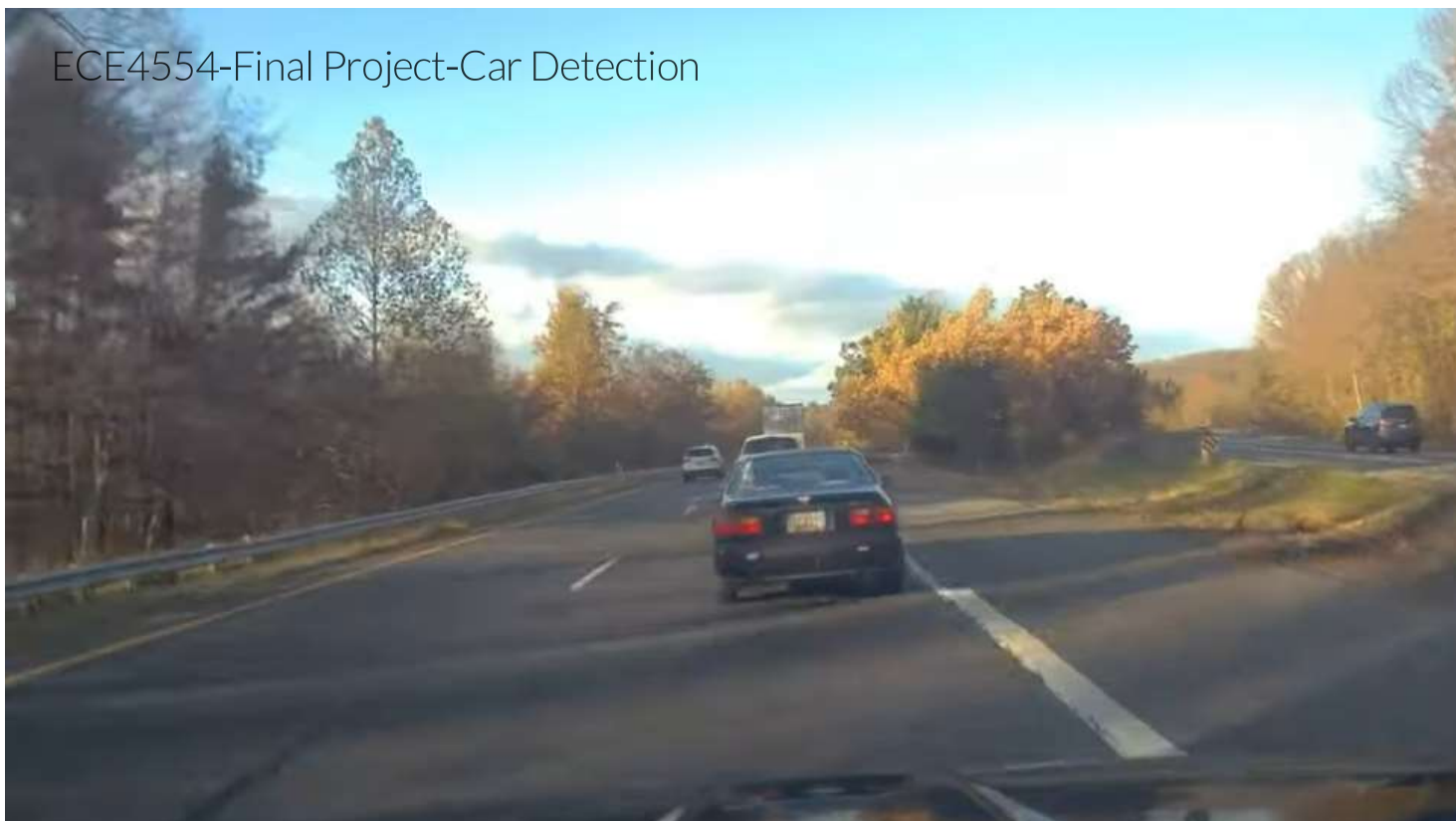


Figure 7. After adding the `center_y_min` and `center_y_max` parameters, the number of potential licenses plates in the image dropped significantly for some images.

Qualitative Results:

Shown below are the final bounding box applied outputs of various images obtained from the dashboard camera videos referenced as [3] and [4].

ECE4554-Final Project-Car Detection



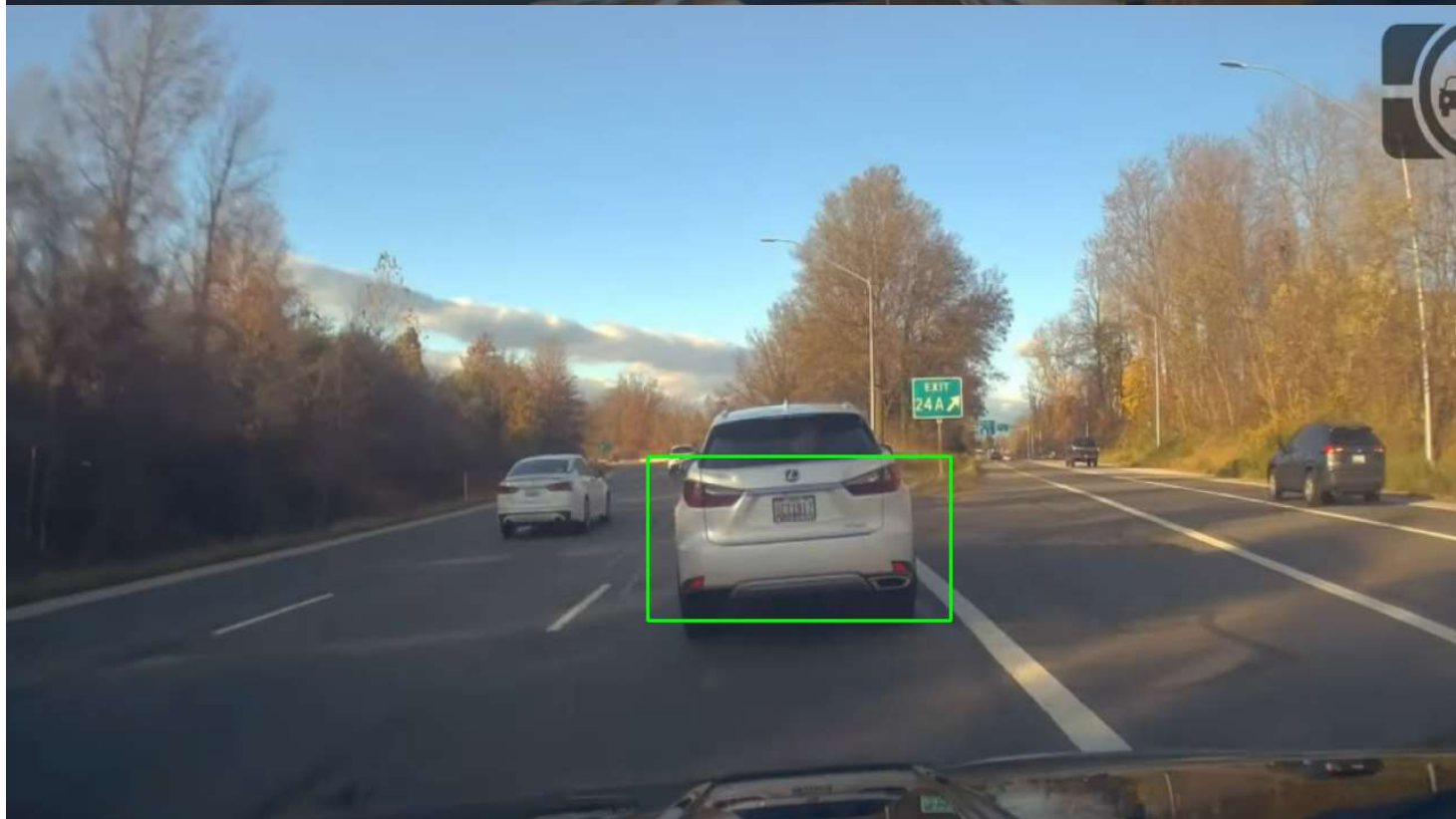
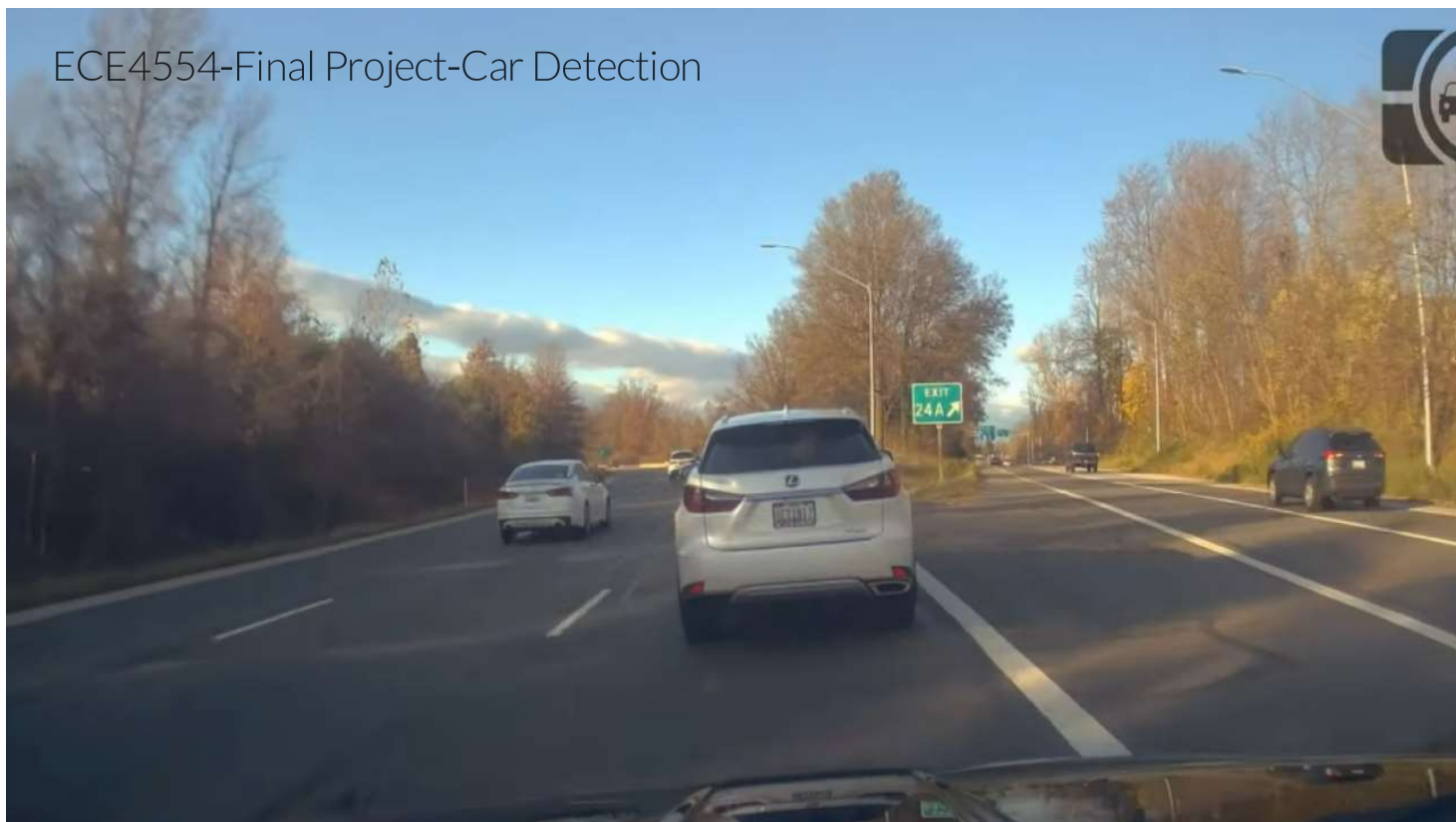




Figure 8. Some sample images (left) and the results obtained after bounding box application

Conclusion:

ECE4554-Final Project-Car Detection

In conclusion, we believe our approach delivered on most of the design objectives we set for the project. Given more time, however, we would have liked to implement our own HoG algorithm to see how feature detection can be further optimized. While we already attempted to implement our own HoG, we had trouble properly creating histograms for 8x8 magnitude and phase cells using input pixel orientations. Additionally, we would have attempted to further tune SVC parameters and experiment with different training data sets to see if we could improve the overall image classification accuracy. Another feature we would like to implement in the future is dynamically changing the size of the bounding box to try and fit the whole vehicle. In some of the results the bounding box does not capture the top part of the vehicle. We believe this change would fix that.

References:

ECE4554-Final Project-Car Detection

Used to obtain a better understanding of the process:

[1] M. Elmasry, “Machine Learning for Vehicle Detection,” *Medium*, Jun. 04, 2018.

<https://towardsdatascience.com/machine-learning-for-vehicle-detection-fd0f968995cf> (accessed Nov. 28, 2022).

[2] B. Djukic, “Detecting vehicles using machine learning and computer vision,” *Medium*, Apr. 28, 2017. <https://towardsdatascience.com/detecting-vehicles-using-machine-learning-and-computer-vision-e319ee149e10> (accessed Nov. 28, 2022).

Used to obtain Dash-cam video footage for bounding box implementation:

[3] “Road Rage USA & Canada | Bad Drivers, Hit and Run, Brake check, Instant Karma, Car Crash | New 2022,” *www.youtube.com*. https://www.youtube.com/watch?v=Udl_ooxx2lk (accessed Nov. 28, 2022).

[4] “BEST OF ROAD RAGE | Brake Check, Karens, Bad Drivers, Instant Karma, Crashes | October USA 2022,” *www.youtube.com*. https://www.youtube.com/watch?v=_4rjH8SPJ7s (accessed Nov. 28, 2022).

[5] “The best dash cam,” *The New York Times*, 24-Jun-2013.

[Online]. <https://www.nytimes.com/wirecutter/reviews/best-dash-cam/>. [Accessed: 28-Nov-2022]

Datasets from [2]

[6] *Amazonaws.com*, 2022. https://s3.amazonaws.com/udacity-sdc/Vehicle_Tracking/vehicles.zip (accessed Nov. 29, 2022).

[7] *Amazonaws.com*, 2022. https://s3.amazonaws.com/udacity-sdc/Vehicle_Tracking/non-vehicles.zip (accessed Nov. 29, 2022).