A VISION BASED CONGESTION SENSING SYSTEM FOR DRIVER ASSISTANCE

A PROJECT REPORT

Submitted by

AKSHAY KUMAR C (312215106009)
GUNUPATI SUMADHURA (312215106032)
HARINI S (312215106034)

in partial fulfillment for the award of the degree
of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

SSN COLLEGE OF ENGINEERING, KALAVAKKAM

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2019

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"A VISION BASED CONGESTION SENSING SYSTEM FOR DRIVER ASSISTANCE"** is the bonafide work of "**AKSHAY KUMAR C (312215106009), GUNUPATI SUMADHURA (312215106032) and HARINI S (312215106034)"** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr.S.RADHA	**Dr.W.JINO HANS**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Professor	Associate Professor
Department Of Electronics and	Department Of Electronics and
Communication Engineering,	Communication Engineering,
SSN College of Engineering,	SSN College of Engineering,
Kalavakkam - 603 110	Kalavakkam – 603 110

Submitted for the project viva-voce examination held on ________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENTS

First, we would like to extend our profound gratitude to **Padma Bhushan Shiv Nadar, Chairman, SSN Institutions** and **the management** for instilling a mission and vision in our minds.

We express our sincere thanks to our beloved principal **Dr.S.Salivahanan** and our HOD, **Dr.S.Radha** for giving us the opportunity to take on this project and for providing us ample time and conducting periodic reviews to keep us engaged in our work.

We also wish to show our appreciation to our panel members **Dr.R.Jayaparvathy and Dr.I.Nelson** for their valuable and innovative suggestions during our meetings that enabled us to improve our project.

Finally, we would like to extend our heartfelt gratitude to our project guide **Dr.W. Jino Hans** for providing us constant guidance and being a pillar of support in order to ensure a successful completion of our project.

AKSHAY KUMAR C

GUNUPATI SUMADHURA

HARINI S

ABSTRACT

Traffic congestion refers to a situation where the vehicles move at a very slow rate or do not move at all. This problem is increasing at an alarming rate day to day. It will be really helpful if the driver is given information about traffic updates at a place before taking a route. The proposed system gives a solution wherein the traffic updates will be uploaded to the cloud. This data can then be retrieved by the user by signing into the cloud. Initially, dataset containing images of cars are collected and features are extracted. Here, HOG (Histogram of Oriented Gradients) along with Adaboost classifier is used for image classification and identification. A dash board camera is used to get the images of the vehicles on road. Count of the vehicles is found out from video by feeding it to algorithm and using heat maps. The count thus calculated is uploaded to Dropbox, along with the images of the congestion at regular intervals of time. Along with images, a text and an audio file with number of cars is also uploaded. Thus, the driver is updated regularly about the traffic updates.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

API	Application Program Interface
HAAR	High Altitude Acute Response
HOG	Histogram of Oriented Gradients
LBP	Local Binary Patterns
SVM	Support Vector Machine
TTS	Text To Speech
URL	Uniform Resource Locator
XG	Extreme Gradient

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Traffic has become a major area of concern in the recent times. With the increase in the number of vehicles being used, the problem of controlling traffic is getting tedious day-by-day. Traffic congestion also called as Traffic Jam is a condition where, vehicles travel at a very slow speed. There are many reasons behind traffic congestions. Some of them are presence of obstacles or mergers, increased volume of vehicles on a roadway, inadequate road system etc. At times, managing congestions become so typical that it takes hours to clear and bring back the situation to normal. Time and fuel will be wasted inevitably because of traffic congestions. Thus, Congestion control is being given utmost importance in the present days.

Intelligent Transport System is a new application which is aiming at better ways of managing transport and traffic. Various innovative ideas are being suggested each one using a distinct type of technology. Google Maps is one such system which provides information about the status of traffic at a place using crowd sourcing. Initially it used some traffic sensors and the data collected from these sensors would be used to provide real time updates. Later it started collecting data by comparing speed of our phone with that of others, to improve accuracy. Phones users need to turn their GPS on to enable their location. As the number of users increase the accuracy also increases. But one of the disadvantages with google maps is that it does not give a clear idea of the number of vehicles present at that place. It just classifies the traffic as either Green route (no traffic roads) or Yellow route(slow moving and moderate traffic) or Red route(high traffic). Vehicles that are moving at a slower velocity, would be concluded as high traffic in some cases. Also, real time images of the traffic congestion are not available in google maps. Another system which is used for providing traffic updates uses RFID technology. RFID tags are employed in vehicles which contain the model and other details of the vehicle. When the vehicle passes the RFID reader, it reads the information that is present in the tag. This data is used to control the traffic lights according to the volume of traffic present. But this system also had many demerits due to RFID tags.

The key disadvantages of RFID tags are that,

- RFID systems are often more expensive than barcode systems
- RFID technology is harder to understand, less reliable, and
- Tags are application specific. No one tag fits all.

Some systems use IR sensors to sense traffic and input this data to a microcontroller. Microcontroller will give its decisions depending on the inputs it receives. GPS module is used to obtain the location details. A communication app is developed to which the data is to be uploaded. Limitations of such systems are due to the sensors that are used. IR sensors are not always reliable and accurate. Certain systems develop an app for uploading the data while some use IOT to connect many devices and send the data that is collected. The proposed project uses a different technology and is innovative to overcome many of the disadvantages of the present systems. This system uses machine learning to detect vehicles from real time videos and then consolidates the data in a more detailed manner. Raspberry Pi is used to implement all the process in real time. This data is uploaded to cloud which is then used by the user effectively.

1.2 MOTIVATION AND OBJECTIVES

With the increase in number of vehicles at a alarming rate, there is a need to develop automated systems that are capable of playing a vital role in controlling traffic congestions. The loss of economy and fuel is a major problem created due to congestions. According to an article published in the Times of India, congestion in major cities like Delhi, Mumbai, Kolkata and Bangalore costs the economy Rs.1.47 lakh crore annually. Many projects and research works were done on vehicle localization and identification. But these works were not extended to use in congestion control. This project concentrates on using the vehicle detection and localization algorithms for deducing an efficient way of providing traffic updates.

The main objectives include:

- To reduce the time and fuel wasted during traffic congestions.

According to a study done by IIT Madras, traffic congestion in Delhi costs around Rs 60,000 crore annually. This is on account of fuel wasted due to idling of vehicles, productivity loss, air pollution and road crashes. As the vehicular population rapidly grows in the capital city, the study projects that congestion cost would increase to Rs.98,000 crore by 2030 unless steps are taken to daily mayhem. So overcome this huge loss, this project provides an effective and efficient solution to this problem.

- To increase the traffic efficiency by increasing road safety and reducing environmental pollution.

According to findings by nation’s top research agencies, Central Road Research Agency (CRRI), Delhi’s eight most erratic bottlenecks are guzzling at least 40000 kms of fuel daily while waiting in vehicles during peak hours of rush. And moreover, the amount of fuel burnt in just idling of vehicles at traffic intersection amounts to 39,806kg (diesel, petrol and CNG combined) and the quantity of carbon dioxide produced equals 1,15,609 kg, that is 115 tonnes of unhealthy black soot. The situation is quite alarming and a proper solution is required at the earliest.

- To enable direct relay of traffic updates to the driver through the information communicated between the vehicles.

It is not possible to cover all the areas by camera. So, a person with a dashboard camera travelling to all the areas provides a better view of the area. Thus, the traffic present in the area is clearer and more understandable to the user in a better way. Thus, information communicated by multiple vehicles gives a better idea of the situation of traffic in that location

- To provide count of vehicles and real time images of the traffic congestion to the user.

The count of vehicle gives an insight to the driver of how long it would take for him to cross the area or whether he should take an alternate route to reduce tie consumption etc. Thus, count of vehicles and the real time images are seen as an add-on, to the driver in wisely making his choice for the particular route.

- To create an audio file that gives the count, which will be of use in cases, where the user is not in a position to look at the count or image.

This audio file helps the driver to get a count of the number of vehicles present in that situation without him having to count it manually. It also reduces the risk of accidents due to driver having to pay minimum attention to know about the traffic update in a particular location.

- To develop a system that assists the drivers in taking smart decisions by providing proper updates (E.g. Toll Gates).

A driver may not be fully aware of which lane to choose at a toll gate. So in order to make him fully aware of the traffic so that time spent on toll gates is less, the above proposed system could be used. It also helps in saving fuel and helps keep control of the environmental pollution as suggested above. The above suggested system need not only be used in toll gates but also in areas where there is lane splitting as in parking lots, airports etc. It helps in reducing the traffic as well as save time for all of us. The traffic in India is moving at a really fast rate which is shown by as suggested by the graph in Figure 1.1

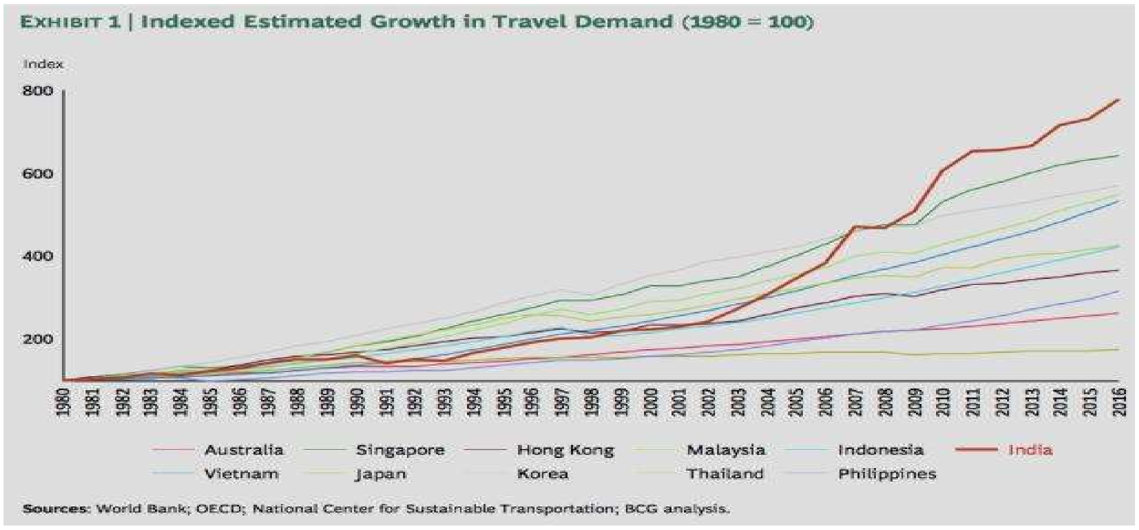


Figure 1.1 Indexed Estimated Growth in travel demand (1980 = 100)

It is seen from the graph that there has been an exponential rise in traffic after 2002-2003. This is due to the population explosion and the increase in people using the vehicles. So there is a need for a system to keep a check on traffic jam for better use of time, fuel while causing minimal negative effect on the environment.

This system aims at uploading valid data about traffic conditions and concentrates on providing real time images to the users. The count of vehicles, at any area where there exists a traffic congestion, gives an idea about the intensity of situation for the driver. A real time image helps in having a better insight of the situation and in taking proper decisions. Road blockages are also intimated about priory. The data that is being uploaded will be updated at regular intervals giving no chance to faulty or untimely information. As the number of users using this system increases, the accuracy of the data uploaded will also increase.

1.3 ORGANIZATION OF THESIS

The thesis is divided into the following chapters:

Chapter 2 contains the review of various methods present for vehicle detection.

Chapter 3 deals with the methodology of the proposed system and highlights the advantages of the system.

Chapter 4 presents the results and observations obtained by implementing the proposed methodology.

Finally, Chapter 5 will summarize the complete project and also highlights the future work.

CHAPTER 2

REVIEW OF LITERATURE

The papers published present an idea on the number of vehicles detected by various methods. To take this process to new level the count of vehicles (in text and mp3) and also its images were sent to the cloud. Here are a few literature papers that gave an insight about the idea of vehicle classification and counting.

In the paper published by Zi Yang. Lilian, S.C. Pun-Cheng. (2018), 'Vehicle detection in intelligent transportation systems and its applications under varying environments: A review, Image and Vision Computing' in Elsevier focuses on Vehicle detection in intelligent transportation systems. They have given an in depth insight on Vehicle detection by appearance based methods and motion based methods, including traffic surveillance objectives. In vehicle detection by appearance based methods, they have discussed Hypothesis generation which is further split into low level features, local feature descriptors (HOG, Haar, Gabor, SIFT, SURF) and Hypothesis verification which contains classifiers like SVM, Adaboost and Neural Networks. In Traffic Surveillance objectives, they have discussed on vehicle counting, detection of accidents and vehicle overtaking. Thus, this journal gave us a basic idea for our project , from which HOG was adapted using Adaboost classifier and implemented vehicle counting too.

In another paper published by Lee, C and David, (2007), 'Boosted Classifier for Car Detection' from Carnegie Mellon University focuses on Adaboost use in car dataset for localization. Here the writer has given an insight into adaboost used in Viola-Jones face detection algorithm and how it has been modified further for vehicle detection. Moreover, the results that are 96% accurate were quite worthy for us to zero onto adaboost for classification.

Further, the paper published by Wang, Y. and De Lin, C. (2007), ‘Learning by Bagging and Adaboost based on Support Vector Machine,’ gives us a lucid idea of boosting algorithm. It also provides us the error percentage in SVM and adaboost classifiers. This further helped us to adopt adaboost in our work based on the error percentages.

The paper published by Chandel, H.S. and Vatta, S. (2015). ‘Vision based vehicle detection with occlusion handling’ discusses the major problem of occlusion handling in vehicle detection. But since the accuracy score for Adaboost was better, the ptoject was proceeded with it. Occlusion is a major problem when it comes to counting vehicles in our project also. Finally, implementation of car detection was performed similarly as specified in paper Vision based motorcycle detection using HOG features.

Heat maps generally define the behaviour of the data. A paper by Fang, J. Zhou, Y. Yu, Y. and Du, S. (2017) ‘Fine-Grained Vehicle Model Recognition Using A Coarse-to-Fine Convolutional Neural Network Architecture,’ describes the use of heatmaps to recognise various models of vehicles. Convolutional Neural Networks is used to generate feature maps and has five layers.

The feature maps are then used to detect the regions of interest. These ROI’s are again taken as input and processed further to obtain more subtle regions until there are no more regions to obtain. Convolution layers are used to convolve previous layer’s feature maps with multiple filter masks to extract features and feed the activation function to generate the output feature maps, which are also the input of the next layer.

Likewise other layers are used to obtain good generalisation and also to locate other important regions. Finally, a heatmap of an average map is generated to indicate which regions contribute more to the final prediction, i.e., the regions with brighter colour are considered to be more discriminative for recognition among similar categories. Thus using the techniques used in papers mentioned above particular algorithm was chosen and was used to count the vehicles and also upload all the images to the cloud (dropbox, google drive etc.)

CHAPTER 3

PROPOSED SYSTEM METHODOLOGY

3.1 OVERVIEW

In our proposed methodology video is captured using the dashboard camera fixed in the car. To localize something as a specific object, it should be identified first. Humans do this by just looking at the object and its features. Thus, for identifying an object its features should be known. Feature descriptor does the job of extracting features. There are many descriptors like Haar Cascade descriptor, HOG descriptor etc. To select a particular descriptor for our work, all the descriptors are to be tested with data and the one which gives maximum accuracy is to be selected.

Once done with describing the main features, it is now time to classify the object correctly based on its features. This work is fulfilled by the classifiers. These classifiers differentiate the distinct objects based on their features. Various classes of classifiers are used which include Adaboost, SVM classifiers etc. Again, selecting a proper classifier is based on the purpose of our work and the accuracy score it gives for our data. With description and classification done, the next step would be to localize the object. Before that, the most important process in any machine learning algorithm is to train it with large datasets.

Localization is the next step, wherein the location of the object is to be found out. A boundary is drawn around the identified object which means that the object in that particular image is within the boundary. For this purpose, sliding window technique is adopted followed by Heat maps.

The process described so far is for images. In case of videos, frames are taken out at regular intervals which are given as inputs and the process is executed in the same way as it is done for normal images. As localization is completed, the count of number of vehicles is calculated by labeling the final windows. The next process which is to be done is uploading this information to Dropbox. These files have to be deleted regularly in order to upload the files from time to time providing updates. For real time implementation Raspberry Pi will be used. Thus, traffic updates are uploaded every now and then which are very useful for the person who is driving a vehicle.

3.2 METHODOLOGY OF THE PROPOSED PROJECT

The block diagram of the methodology shown in Figure 3.1 contains two main tasks. The first task is Computer vision where the features of the cars are extracted. From the features obtained the next step is to use them to detect the presence of a car in the image or video of interest. After localizing them using a bounding box counting the number of cars in that region is done. Once the number of cars is known the second main task is uploading it to the cloud for usage. This is done by creating a Zip file. This Zip file will contain all details notifying people about the traffic in the region. And finally, the required data is retrieved by the user.

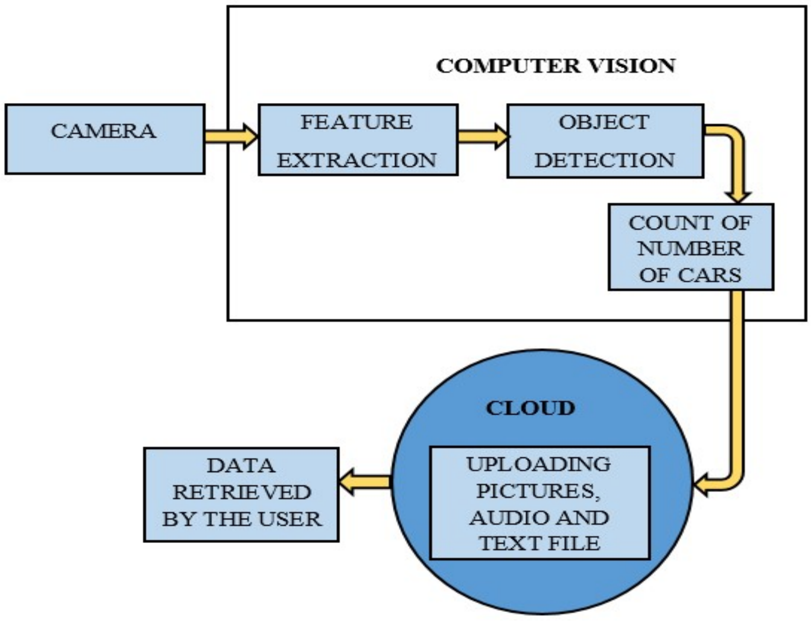


Figure 3.1 Block diagram of the methodology

3.2.1 Computer Vision

The main aim is to count the number of vehicles present in an area. For this the first step is to localize the vehicles. In order to do that training our computer with the features of the car is necessary. Once it knows the features of the car, it becomes easy for it to find if a car is present or not. So for achieving this information about various feature descriptors, color spaces and classifiers is needed.

3.2.1.1 Feature Descriptors

In the aspect of computer vision and image processing, a feature is a piece of information which is relevant for solving the computational task related to a certain application. So it is basically an algorithm which takes an image as its input and gives back the key points of the features. By describing the area around the feature, this information is used to do object recognition, real time tracking, image retrieval etc. The algorithm which takes these key points as its input and outputs the descriptors for them is called the feature descriptor. So here are some very important feature descriptors:

- *Histogram of oriented gradients:* The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for object detection (Dalal, N et.al) (2005). The following are the steps involved in calculating the HOG descriptor:

Preprocessing: Before calculating HOG features it is necessary to pre-process the image to obtain square blocks of histograms which are used for a fast HOG calculation. For example, if there is an image of size 750x475 and have a selected patch of size 100x200 for calculating the HOG feature descriptor. This patch is cropped out of the image and resized.

Gradient calculation: To calculate a HOG descriptor, calculation of horizontal and vertical gradients is needed. This is done by using a Sobel operator in

OpenCV. Next the magnitude and direction of gradient are calculated. The gradient image removes a lot of non-essential information but highlights the outlines. At every pixel, the gradient has magnitude and direction. For color images the gradient of three channels are evaluated. The magnitude of gradient at a pixel is the maximum of the magnitude of gradients of the three channels, and the angle is the angle corresponding to the maximum gradient. In this step, the image is divided into n×n cells and a histogram of gradients is calculated for each n×n cells. The reason for dividing into nxn cells is for compact representation and calculating a histogram over a patch makes this representation more robust to noise. The patch of the image overlaid with arrows shows the gradient - the arrow shows the direction of gradient and its length shows the magnitude. The direction of arrows points to the direction of change in intensity and the magnitude shows how big the difference is.

Block normalization : Ideally, our descriptor should be independent of lighting variations. So normalizing the histogram will help it not being affected by lighting variations. For example: A 16×16 block has 4 histograms which are concatenated to form a 36 x 1 element vector and it is normalized. The window is then moved by n pixels and a normalized 36×1 vector is calculated over this window and the process is repeated.

Calculation of Histogram Feature Vector: Here the final feature vector for entire patch is calculated. All the vectors calculated before are concatenated into one giant vector.

- *HAAR like feature descriptor:* Haar like features are features used mainly for object recognition. Historically, working only with only image intensities made the calculation computationally expensive. Thus an alternate feature set based on Haar wavelets instead of the usual image intensities was used. A Haar-like feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image (Banerji, S et.al) (2013). A window of the target size is moved over the input image, and for each subsection of the image the Haar-like feature is calculated. This difference is then compared to a learned threshold that separates non-objects from objects. Because such a Haar-like feature is only a weak learner or classifier a large number of Haar-like features are necessary to describe an object with sufficient accuracy. The Haar-like features are therefore organized in something called a classifier cascade to form a strong learner or classifier.

- *Local binary patterns:* Local binary patterns (LBP) is a type of visual descriptor used for classification in computer vision. It is found to be a powerful feature for texture classification. And it has further been determined that when LBP is combined with the Histogram of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

3.2.1.2 Color Models

The purpose of a color model (also called color space or color system) is to facilitate the specification of colors. In essence a color model is specification of a coordinate system and a subspace within that system where each color is represented by a single point. Most colors used today are oriented towards the hardware (such as monitors and printers) or toward applications where color manipulation is the goal. There are numerous color models in use today due to the fact that color science is broad field that encompasses many areas of application.

The color models analyzed are:

- *RGB color model:* It's probably the most popular color space. It stands for Red, Green, and Blue. This model is based on a Cartesian coordinate system.

It is an additive color space where various colors are obtained by linear combination of red, green and blue. Each value ranges between 0 and 255. The correlation between the three channels depend on the amount of light hitting the surface. All values of R,G,B are in the range [0 1]. The number of bits used to represent each pixel in RGB space is called the pixel depth.

- *YUV color model:* The YUV model defines a color space in terms of luma (Brightness) component (Y') and two chrominance (UV) components which provide color information and are color difference signals of blue minus luma (B-Y') and red minus luma (R-Y'). This works well in many applications because the human visual system perceives intensity information very differently from color information. The main reason to implement Y′UV would be for interfacing with analog or digital television or photographic equipment that conforms to certain Y′UV standards.
- *YCrCb Color Model:* Y is the luma component of the color. Cb is the blue component relative to the green component. Cr is the red component relative to the green component. YCbCr was used for digital encoding of color information suited for video and still-image compression and transmission such as MPEG and JPEG.
- *HSV Color Model:* H- Hue which refers to the color, S- Saturation refers to the brightness of the color and V- Value refers to the lightness of the color. This is a cylindrical that represents those using different channels. This is closely related to how the human visual system understands color. This color space is frequently used in color selection tools in software and for web design.

3.2.1.3 Classifiers

- *Support Vector Machine (SVM):* Support Vector Machine (SVM) is a supervised machine learning algorithm which is used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space (where n is number of features that are present) with the value of each feature being the value of a particular coordinate (Cengil, E et.al) (2017). Then, classification is done by finding the hyper-plane that differentiate the two classes very well.

- *Boosting:* Boosting is a method of converting a set of weak learners into strong learners. To convert a weak learner into strong learner, a family of weak learners is taken, combined and voted. This turns this family of weak learners into strong learners. The idea here is that the family of weak learners should have a minimum correlation between them. The different types of boosting algorithms are:
- *Ada Boost*: Ada-boost is an ensemble classifier which is made up of multiple classifier algorithms and whose output is combined result of output of those classifier algorithms. This classifier is generally used for binary classification. Ada-boost classifier combines weak classifier algorithms to form a strong classifier (Wang, Y et.al) (2006). A single algorithm may classify the objects poorly. But by combining multiple classifiers with selection of training set at every iteration and assigning right amount of weight in final voting, good accuracy score for overall classifier is achieved. Each weak classifier is trained using a random subset of overall training set. After training a classifier at any level, Ada-boost assigns weight to each training item. Misclassified item is assigned higher weight so that it appears in the training subset of next classifier with higher probability (An, T et.al) (2010). After each classifier is trained, the weight is assigned to the classifier as well based on accuracy. More accurate classifier is assigned higher weight so that it will have more impact in final outcome. The classifier with 50% accuracy is given a weight of zero, and the classifier with less than 50% accuracy is given negative weight. Thus Ada-boost like random forest

classifier gives more accurate results since it depends upon many weak classifiers for final decision.

- *XG Boost*: It is called EXTREME Gradient Boosting. XG Boost provides a parallel tree boosting, that solve many data science problems in a fast and accurate way. It provides good execution speed and model performance. The trees have a varying number of terminal nodes and left weights of the trees that are calculated with less evidence is shrunk more heavily. The extra randomization parameter is used to reduce the correlation between the trees. The lesser the correlation among classifiers, the better our ensemble of classifiers will turn out.
- *Random Forest Algorithm*: Random Forest is a flexible, easy to use machine learning algorithm that produces a great result most of the time. It is also one of the most used algorithms, because of its simplicity and the fact that it is used for both classification and regression tasks. Random Forest is a supervised learning algorithm. Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Random Forest has nearly the same hyper parameters as a decision tree or a bagging classifier.

  Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

  The trees are made more random, by additionally using random thresholds for each feature rather than searching for the best possible thresholds. It being an ensembled algorithm, gives more accurate results. Random Forest is also considered as a very handy and easy to use algorithm, because its default hyper parameters often produce a good prediction result. The number of hyper parameters is also not that high and they are straightforward to understand.

3.2.1.4 Efficiency Score calculation

Having the knowledge of various color spaces and classifiers, combinations of them were used to calculate an efficiency score. Table 3.1 shows the efficiency score obtained for the combination of the color spaces with various classifiers. Efficiency score here is the ability of the classifier to differentiate between car and non-car images.

Table 3.1 Efficiency score for combination of color spaces and classifiers

Sl.No	Color Spaces	No.of Channels Used	SVC (%efficiency)	XG Boost (%efficiency)	AdaBoost (%efficiency)
1	YUV	3	98.73	98.82	99.38
2	YCrCb	3	98.98	98.51	99.27
3	HSV	1(Value)	94.65	97.38	99.04
4	RGB	3	96.34	97.41	98.96

The following were the observations from the efficiency scores:

- HOG outperforms other feature descriptors like Haar, LBP etc.
- Ada-boost classifier is better than SVM classifier with better accuracy.
- Among Color Spaces YUV performs better than others for cars with better accuracy score.

Therefore Ada-boost classifier and YUV color space were chosen for detection, classification and counting of vehicles. Before starting, data set containing 8792 pictures of cars and 8968 pictures of non-cars was collected. The purpose of these datasets is to train the classifier to improve accuracy and reduce errors. The data sets were stored in two different folders with one containing only vehicle images and the other one containing non-vehicle images.

3.2.1.5 Preprocessing

Resizing of images in dataset: All the images in the dataset are first resized to 64x64 and these images are again stored in a separate folder. Figure 3.2 shows the resized images of vehicles obtained from the dataset and Figure 3.3 shows the resized images on non-vehicles from dataset. Resized images are then read from the respective folders and were appended to a list separately.

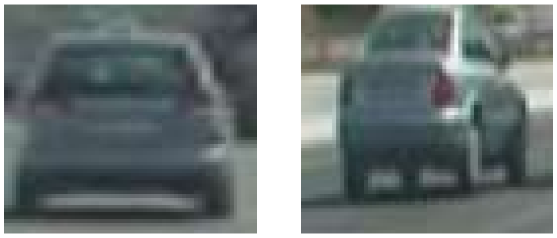


Figure 3.2 Resized images of vehicles

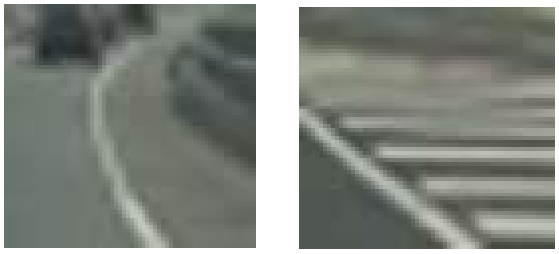


Figure 3.3 Resized images of non-vehicles

3.2.1.6 Feature Extraction

Now that the resized dataset is obtained, the next step is to extract features to train our model for automatic detection and localization. In order to create a machine learning pipeline three types of features were used:

- Binned color (color and shape features)

- Color histogram features (color only features) which is a representation of the distribution of colors in an image.
- HOG (Histogram of Oriented Gradients) is a feature descriptor used in computer vision and image processing for the purpose of object detection.

*Color Spaces: C*olored Histogram was calculated. The Histogram for all the three channels is calculated separately. A picture from resized vehicles folder is chosen. The image is changed into the desired color space (YUV in this case). Then the colored histogram is extracted. Thus Figure 3.4 shows the colored histogram for image of a car. Similarly Figure 3.5 shows the colored histogram for image of a non-car.

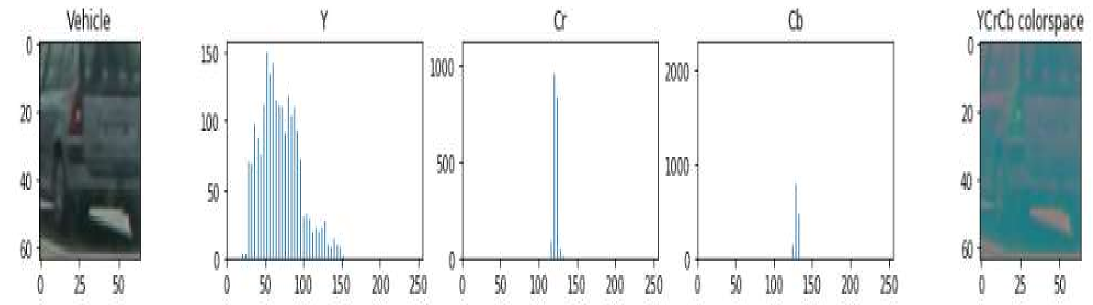


Figure 3.4 HOG for the image of a vehicle

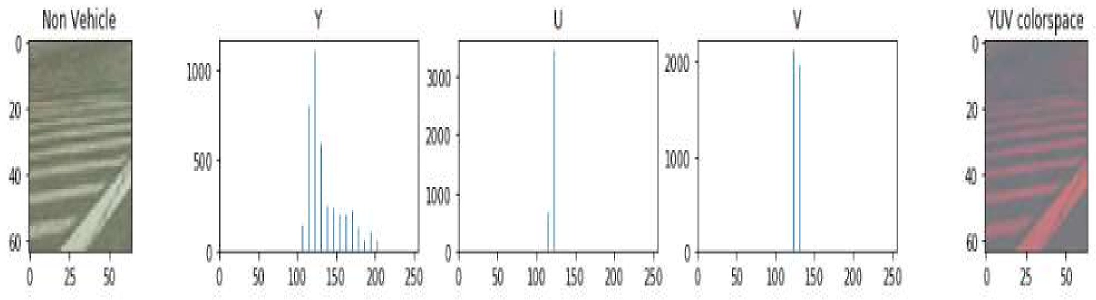


Figure 3.5 HOG for the image of a non-vehicle

Spatial binning: Here an image is resized to extract features and also to reduce the feature vector size (Banerji, S et.al) (2013). Number of features before spatial binning was 12288. After performing binning the number of features reduced to 768.

Histogram of oriented gradients: This procedure is tested for a vehicle and non-vehicle images separately. First images were converted to the desired color space and then the HOG features were calculated. In general method of obtaining the HOG features the following parameters are needed to be given:

i. Image for which the HOG features are needed
ii. Orientation
iii. Number of Cells per block
iv. Number of Pixels per Cell
v. Visualize = False/True
vi. feature_vector_flag =True

Generating data of features : First of all conversion to desired color spaces should be done. Then features are extracted for each image in the vehicle and non-vehicle folders separately. After Extracting all the features from vehicle and non-vehicle images, the final feature set was created by stacking them together and then created the final labels by stacking ones equal to length of vehicle data and stacking zeros equal to length of non-vehicle data.

3.2.1.7 Classifier training

After stacking all the features extracted from vehicle and non-vehicle dataset, the next part is to train the classifier. But before training there are a lot of data preprocessing steps that were performed-

1. Splitting of Data into Training and Test Set- Data was split into training and test set to combat overfitting and test our model on test set. Also before splitting it was made sure that the data is properly shuffled.

2. Normalization and Scaling of Data- Data was normalized using Standard Scaler function of Sklearn.preprocessing class. So 20% of data set was used as test images and remaining 80% is used for training.

For training the machine, Ada-Boost classifier and Random forest classifier were imported from sklearn.ensemble. This is made to run over the test and training data. Finally the accuracy score was calculated.

3.2.1.8 Sliding Window approach

Before going into the procedure let us first understand what exactly a sliding window approach is. In order to use our predictive model on a video feed, need an appropriate and consistent partitioning algorithm that lets us search our video feed image is needed. The searching algorithm should return sub-images from the image which are then fed into our model to make a prediction. One such searching algorithm is the sliding-window search technique. Sliding windows play an integral role in object classification, as they allow us to localize exactly where in an image an object resides. In the context of computer vision, a sliding window is a rectangular region of fixed width and height that slides across an image. For each of these windows, window region was taken normally and an image classifier was applied to determine if the window has an object that interests us.

To detect a car in a test input image, sliding window of size (x) was picked and input region (x) was fed to trained convent by sliding window over every part of input image. For each input region, convent outputs whether it has a car or not. Sliding window is run multiple times over the image with different window size, from smaller to larger, hoping a window size would fit the car and allow convent to detect it. There are some important points to be kept in mind for implementing a sliding window:

i. Cars will always be only in the lower half of the image so searching will be done in the lower half alone. The cars near the chosen horizon will be small and will increase in size as one moves from the horizon towards the car.
ii. It makes sense to search for 64x64 car in only one window starting from horizon and increase the window size as one moves from horizon towards car.
iii. For Windows of different sizes resizing the window back to 64x64 is needed because while training the classifier training was done on features extracted from 64x64 image. The overlap selected was based on the fact that all possible points were covered.

A function to draw windows of desired thickness was defined. The parameters required are - img : source image array like/list or image files list to be examined ; bboxes : bounding box diagonal coordinates ; color : one color or random color if random ; thick : line diagonal coordinates. Then a copy of the image is made. Finally a rectangle is drawn given the bbox coordinates. And then the image copy with boxes drawn is returned.

Another function to find the windows on which the classifier should run was written. The parameters used were - img : image selected ; x_start_stop: Region Of Interest in x direction ; y_start_stop: Region Of Interest in y direction ; xy_window: size to draw ; xy_overlap: how much the window is to overlap with that of the adjacent one.

The span of the region to be searched was then computed. Number of pixels per step and number of pixels per step in x/y were also computed. Finally a list was initialized to append the window positions. A function to return a refined window was created. Refined window is the one where the classifier predicts the output to be a car. Finally, this algorithm was checked by giving an image as input. For this first windows were declared by using the coordinates mentioned above. Figure 3.6 shows the window coverage obtained using windows of different sizes and overlaps. And finally Figure 3.7 gives refined window output.

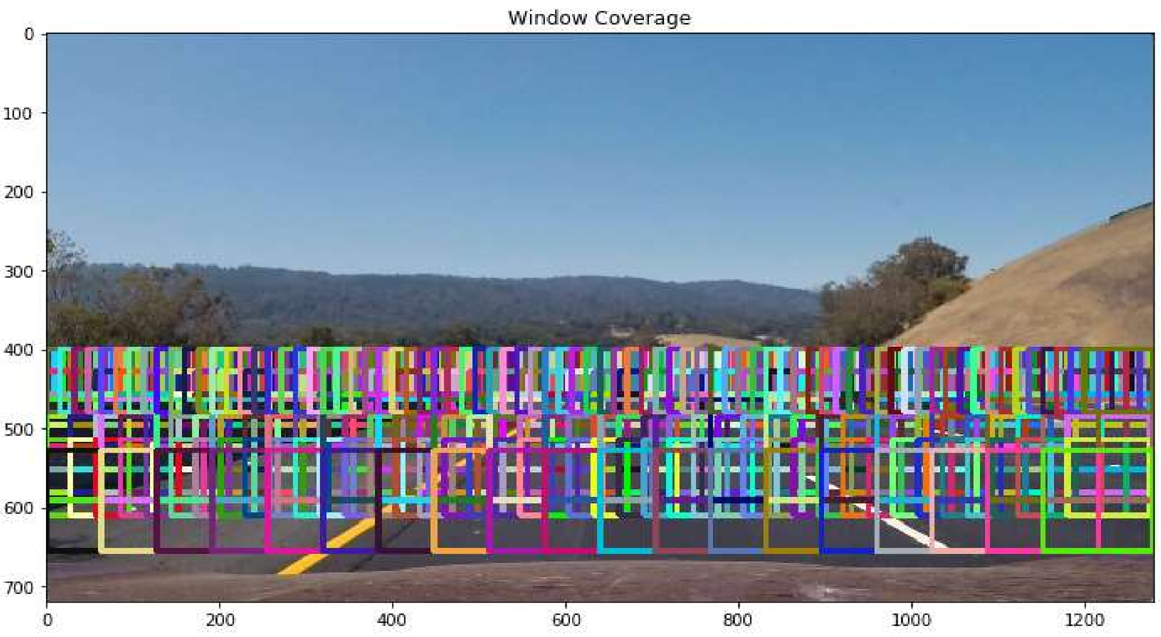


Figure 3.6 Window coverage of sliding windows

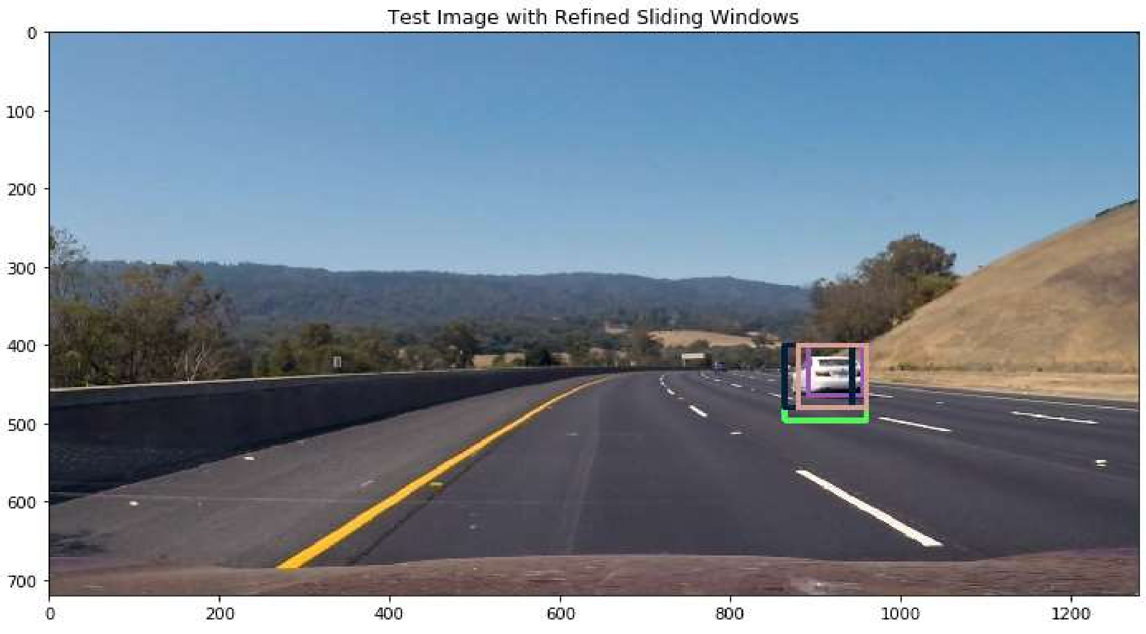


Figure 3.7 Refined sliding window for a test image

3.2.1.9 Heatmaps

As multiple boxes were obtained around a single car by implementing sliding window which is undesirable, Heatmaps were used to optimize the condition. Now given the simplicity of the classifier model, some detections were expected to be false positives. In order to filter out these incorrect detections, one approach is to threshold our positive windows such that it only picks areas where more than one window overlap. In essence a heatmap of the positive windows was generated. In addition, using the generated heatmap, calculation on bounding boxes on the detected object was done using the location of the pixels furthest away from the center of the heatmap. These bounding boxes are used to visually identify our detected cars.

First a function was defined which increments the pixel value of an black image to the size of the original image at the location of each detected window which is called as refined Window. This increments through the list of bboxs. It adds =+1 for all pixels inside the each box.

The next step was to remove the false positives. Averaging approach was used in the video pipeline that sums all the heats coming in the past 15 frames and then apply threshold using the function apply_threshold. This function blacks out the pixels which have value less than threshold value. And finally returns the threshold map. Finally the next point was to draw bounding boxes on the final image. The scipy.ndimage.measurements.label() function collects spatially contiguous areas of the heatmap and assigns each a label.

STEPS:

i. Iterate through all detected cars.
ii. Find pixels with each car_number label value.
iii. Identify x and y values of those pixels

iv. Define a bounding box based on min/max x and y

v. Draw the box on the image

vi. Return the image.

Now finally this was tested by giving a test image as input and these were the results obtained. Figure 3.8 shows the localization of the car using heatmaps and Figure 3.9 highlights the refined window around car.

```
Number of Cars found -  1
Text(0.5, 1.0, 'Final Image after applying Heat Map')
```

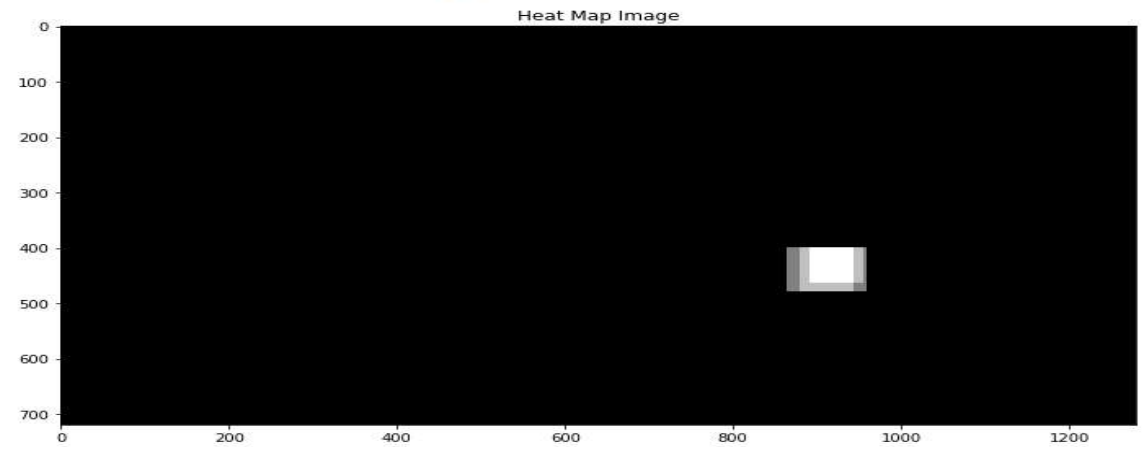


Figure 3.8 Heatmap application

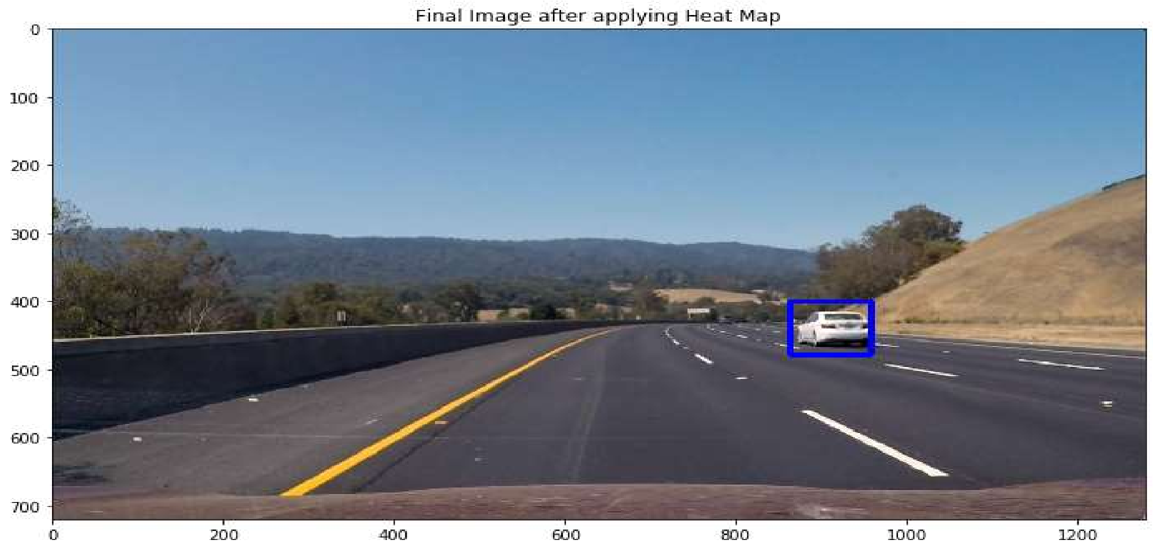


Figure 3.9 Refined box after application of heatmap

3.2.1.10 Pipeline

First was defining parameters required for the pipeline. Given the required parameters for each window in the sliding window function, the first step is to extract all search windows from the image. Then convert each window image to YUV color space to extract features. Each obtained feature is fed into the classifier model and non-positives are filtered out. Finally heatmap thresholding was applied to positive matches and bounding boxes were drawn.

Defining a pipeline function for processing of video frames is where both the summing and averaging code was taken into account. The following were the steps followed for this approach:

- A function pipeline for the processing of video frames was defined. In this code storing previous refined windows is done in a class called KeepTrack. This class stores the refined frames found from the last 15 frames.
- The threshold was changed to 25 + len(windows)/2. This was done to remove the false positives detected. The positions of positive detections in each frame of the video was recorded. From the positive detections creation of a heatmap is done and then thresholded that map to identify vehicle positions. Then scipy.ndimage.measurements.label() is used to identify individual blobs in the heatmap.
- These were corresponded to a vehicle and bounding boxes were constructed to cover the area of each blob detected. For Combining Bounding Boxes heatmap were used. draw_labeled_bboxes is the function that has the responsibility to draw the final bounding boxes.

Finally there is another function for analyzing a test image. A function PipelineImage was defined for this purpose. After defining this function testing this approach was done by a test image. The final result was a bounding box around the detected car in the image given for testing. The final results are highlighted in Figure 3.10 which shows the original image, localization of car using heatmaps and final refined window showing the presence and location of car.

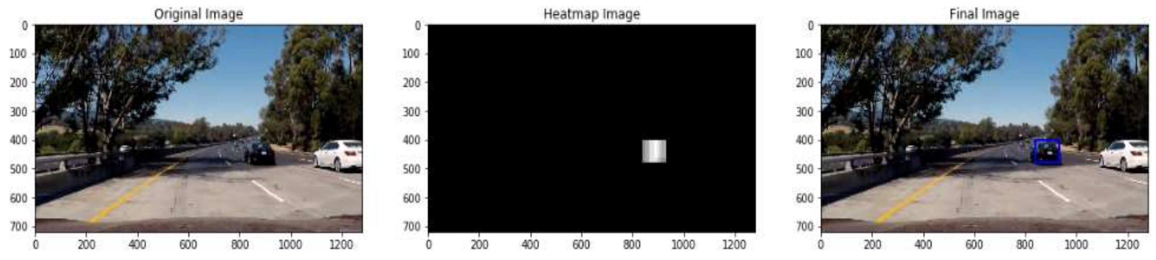


Figure 3.10 Localization of car

3.2.1.11 Frame Splitting

For separating frames from a video to be tested the following procedure was used:

i. A function extractFrames was defined which has following parameters:
   a. pathIn : It is the argument for passing the input video.
   b. pathOut : This is the argument that indicates the folder where the extracted frames are to be saved.
ii. os.mkdir(pathOut) creates a path to the output folder.
iii. cap = cv2.VideoCapture(pathIn) is used to load the input video.
iv. frameRate = cap.get(5) gives the frame rate. Sometimes, cap may not have initialized the capture. In that case, this code shows error.
    To check if it is initialized or not cap.isOpened() method is used.
v. frameId = cap.get(1) gives the frame number.
vi. cap.read() returns a bool (True/False). If frame is read correctly, it will be True. So if this returns false it breaks the execution here. Then if

(frameId % (5*math.floor(frameRate)) == 0) is satisfied then frame obtained is correct ,the output is printed TRUE.

vii. cv2.imwrite(os.path.join(pathOut,"frame{:d}.jpg".format(count)), frame) this was to save frame as JPEG file.

extractFrames('project_video.mp4','data') is where project_video.mp4 is the test video and data is the folder where the frames of the video are saved. Figure 3.11 shows the storage of frames in the folder created.

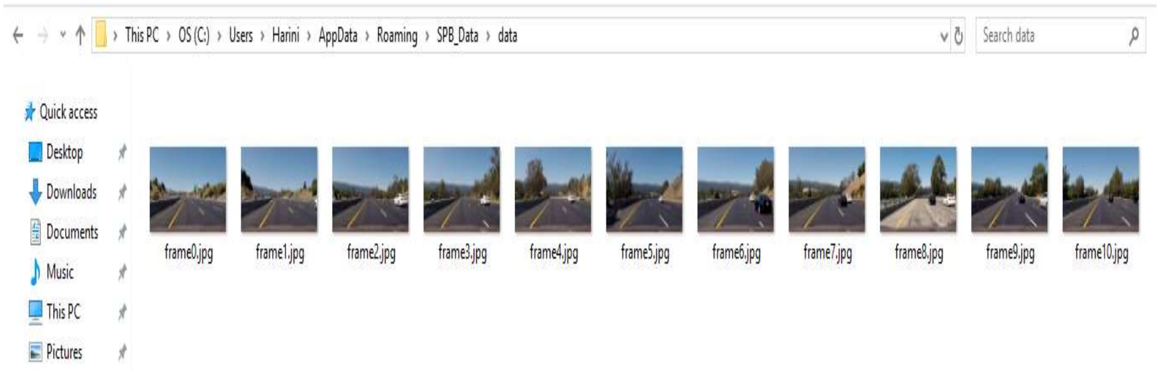


Figure 3.11 Frames stored in a folder named data

Detection of vehicle in the frames obtained: The extracted frames were given as input for detection and the procedure followed as described below and Figure 3.12 shows the processed frames stored in the folder and Figure 3.13 shows the localization of the car in the frames obtained.

i. A variable was initialized to 1 (say i=1)
ii. Frames were read one by one and the colorspace was changed to RGB.
iii. finalPic,heatmap = PipelineImage(image) is where the detection is done by using the function PipelineImage as described explained above.
iv. cv2.imwite('./Final/%d.jpg'%i,finalPic) is where a folder named Final is created and the processed frames are stored.
v. The variable is incremented after that.

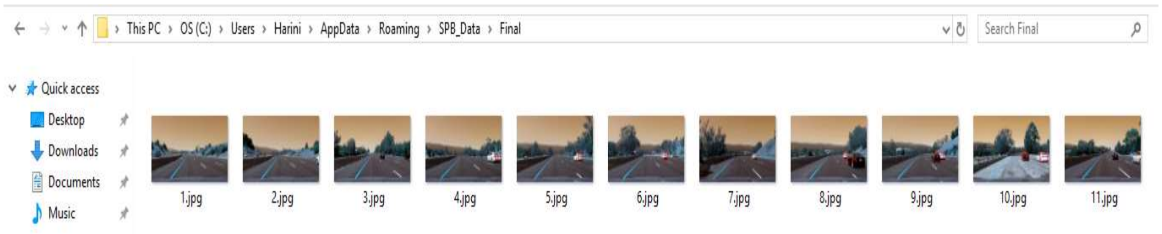


Figure 3.12 Output frames with localized cars

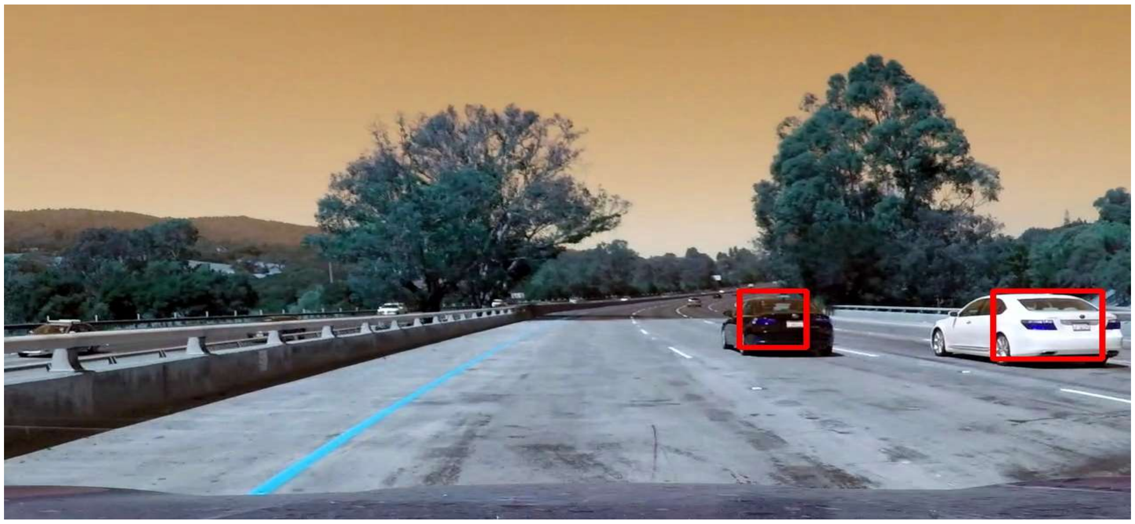


Figure 3.13 Processed frame from video

3.2.2 Cloud Upload

Now that the number of cars is known in the area, next step was to upload this information to the Dropbox. For this the first step is to know the location information. After knowing this a Zip file is created. This file is named after the town name. Then an audio and text files containing the information about the traffic were created. Finally they are sent into the Zip file which is then uploaded to the Dropbox for driver's usage.

3.2.2.1 Location Details

For getting a location name given the latitude and longitude values some files were needed to be installed like geopy, geocoder and googleplaces.

For installing the following commands pip install geopy, pip install geocoder and pip install python-google-places were used.

Geopy : geopy is a Python 2 and 3 client for several popular geocoding web services. geopy makes it easy for Python developers to locate the coordinates of addresses, cities, countries, and landmarks across the globe using third-party geocoders and other data sources. *Geocoding :* Geocoding is the computational process of transforming a physical address description to a location on the Earth's surface (spatial representation in numerical coordinates). Reverse geocoding, on the other hand, converts geographic coordinates to a description of a location, usually the name of a place or an addressable location. *Google-places :* Google Places API Web Service allow the user to query for place information on a variety of categories, such as establishments, prominent points of interest, geographic locations, and more. One searches for places either by proximity or a text string. Nominatim from geopy.geocoders was imported. Each geolocation service that might be used, such as Google Maps, Bing Maps, or Nominatim, has its own class in geopy.geocoders abstracting the service's API.

- Then from a set of latitude and longitude, address is obtained using geolocator.reverse. This reverse method resolves a pair of coordinates to an address. For example : If the latitude and longitude values were 52.509669, 13.376294 , the location obtained was Potsdamer Platz, Tiergarten, Mitte, Berlin, 10785, Deutschland.
- a=location.raw['address'] will give the entire complete address with the details about the country, city, city_district and much more.
- From the above detailed address suburb alone was extracted using a.get('suburb', None) as Tiergarten and it stored as town. Having got the location information, next step was to create a Zip file.

3.2.2.2 Zip file creation

The files that are uploaded to cloud include a text file, and audio file and an image. All these three files are combined into a single zip file instead of uploading as three different files. For this another set of latitude and longitude belonging to Siruseri was used. A zip file with the town name was created using zip = zipfile.ZipFile('%s.zip'%(str(town)),'w'). The zip file was saved in the same directory as to where the anaconda was installed.

The text file contains a statement about number of cars present. Using this text file an audio is created and then both the files are uploaded to Dropbox for driver assistance. A car image is also uploaded to the Zip file. For this it is needed to first install the gTTS (Text To Speech) using the command pip install gTTS in the anaconda prompt.

- A given string is converted into an audio using tts = gTTS(text='The number of cars present is %d'%labels[1], lang='en').
- This audio was then saved as tts.save("cars.mp3").
- Finally the audio file was saved into the zip file using zip.write('cars.mp3').
- A text file was created using f=open("Car Information.txt","w+").
- Content to be written is updated using f.write("The number of cars present is:%d"%labels[1])
- This text file was written to the zip file using the command zip.write('Car Information.txt').
- Now the image of the car has to be uploaded to the Zip file.
- After getting the frame from a video, it is given a name as name="car.jpg".
- It is then uploaded into the zip file using the command zip.write("car.jpg").

3.2.2.3 Dropbox Upload

For uploading files to the Dropbox all that is needed is a Dropbox-access-token. The rudimentary requirement for this procedure is a config.json file. Dropbox allows users to create a separate app on it as per their requirements. The variables which the app needs are stored in this file. Thus, using this dropbox_access_token it becomes easy to upload files to our file. Use the token generated for the app that was created for traffic updates. Then client was configured to the desired dropbox_access_token. The files are finally uploaded to Dropbox.

3.2.2.4 Pickle file

For real time implementation Raspberry Pi 3 B+ is used. Pickle file is to be used for training the large dataset. Python pickle module is used for serializing and de-serializing a Python object structure. Any object in Python is pickled so that it is saved on disk. What pickle does is that it serializes the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script.

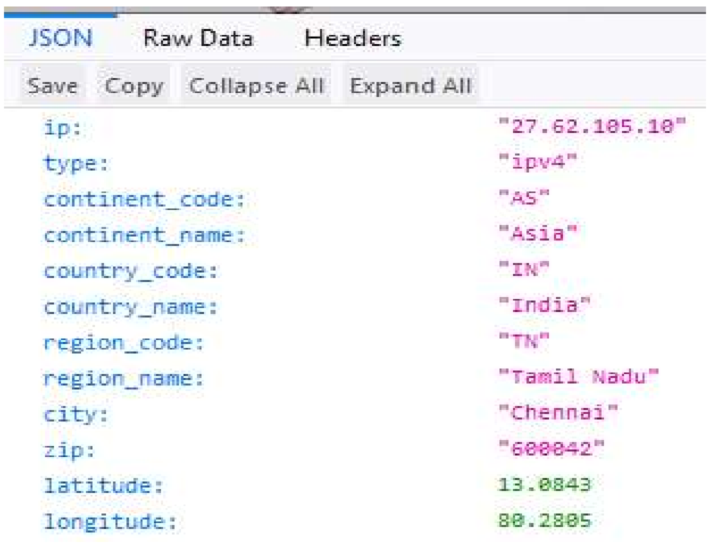
So in this case first the desired dictionary was loaded into the pickle file using the command pickle.dump(classifier2,open(filename,’wb’)) where filename = final_model.sav where sav is a file extension used for the saved data. Then again the dictionary is loaded from the pickle file using the command classifier2 = pickle.load(open(filename, 'rb')).

cPickle FILE : In Python 2, pickle access is speeded up with cPickle. (In Python3, importing pickle will automatically use the accelerated version if it is available.) This is assessed using the same commands by using cPickle in place of Pickle.

3.2.2.5 Model Finalization using joblib

Joblib is part of the SciPy ecosystem and provides utilities for pipelining Python jobs. It provides utilities for saving and loading Python objects that make use of NumPy data structures, efficiently. This is very useful for some machine learning algorithms that require a lot of parameters or store the entire dataset. So using this finalizing of our model was done by the command joblib.dump(classifier2, 'classifier.save'). Location in real time is found using the following method:

i. For this first an URL of a json file was specified. To get the basic information about a given visitor IP address, make a call to the URL. The response of the URL will look like in Figure 3.14.



```
ip:                  "27.62.105.10"
type:                "ipv4"
continent_code:      "AS"
continent_name:      "Asia"
country_code:        "IN"
country_name:        "India"
region_code:         "TN"
region_name:         "Tamil Nadu"
city:                "Chennai"
zip:                 "600042"
latitude:            13.0843
longitude:           80.2805
```

Figure 3.14 Details obtained from the URL

ii. Then the information is received using the .get command.

iii. This received data was loaded as a text in a json file.

iv. From this the latitude and longitude were obtained.

After completing all the required steps, the Zip file containing the information about the number of cars present was finally uploaded to the Dropbox.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 OVERVIEW

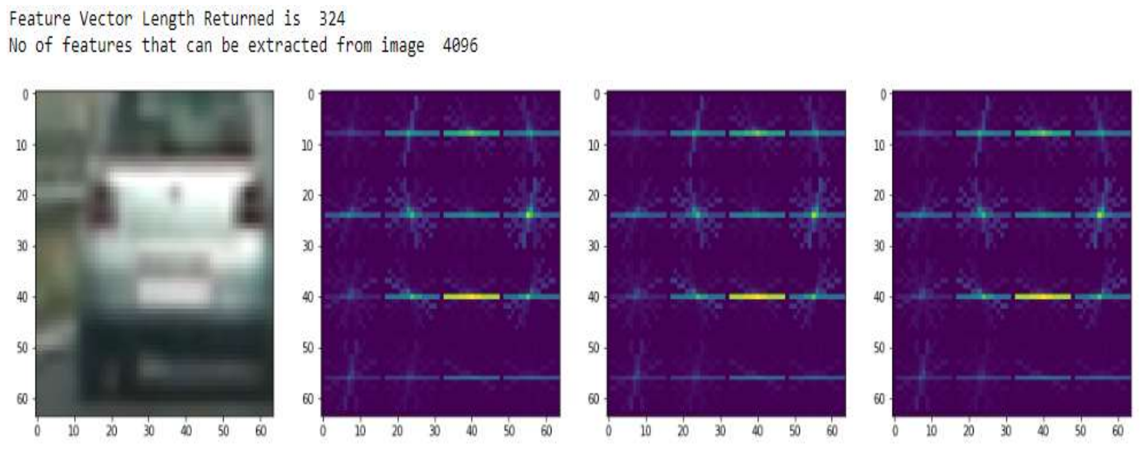
In this chapter the results of implementing the proposed methodology to reduce traffic congestion are discussed. By using this methodology finally training the algorithm the features of car for automatic detection and localization was made easy. Using this information counting the number of cars present in the area of interest was achieved. After this creation of an audio and text file notifying the driver about the number of cars present in the region was done. Finally, these contents were uploaded to Dropbox for driver assistance.

4.2 FEATURE EXTRACTION USING HOG

Once after obtaining the dataset, the first task is to extract features from it using Histogram of oriented gradients. These features will help in training the model for detection and localizing a car. HOG parameters used are extracting features are shown in Table 4.1 and HOG features are shown in Figure 4.1 and Figure 4.2.

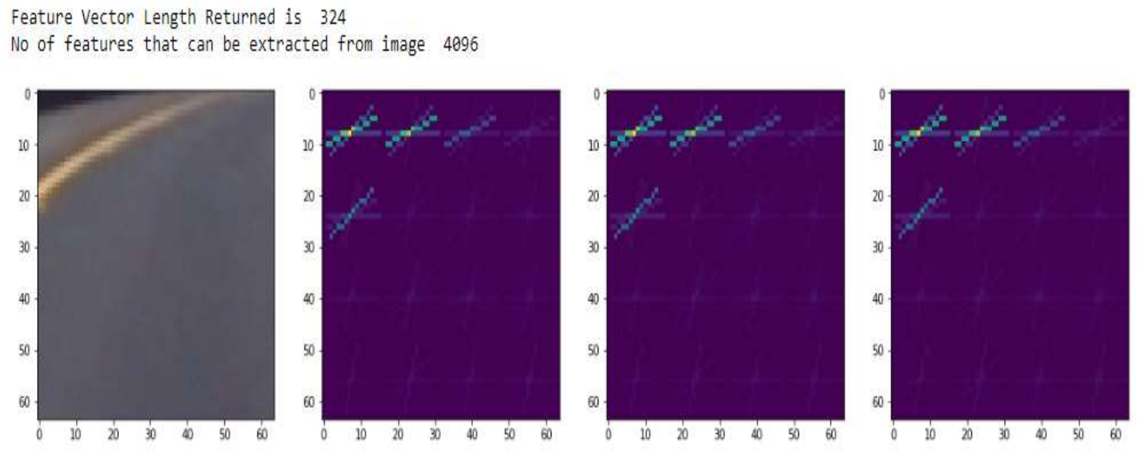
Table 4.1 Parameter values chosen for calculating HOG features

PARAMETER	VALUE
Orientation	9
Cells per block	2
Pixels per cell	16
Color space	YUV



```
Feature Vector Length Returned is  324
No of features that can be extracted from image  4096
```

Figure 4.1 HOG features of a vehicle image



```
Feature Vector Length Returned is  324
No of features that can be extracted from image  4096
```

Figure 4.2 HOG features of a non-vehicle image

4.3 LOCALIZATION

After extracting the features they are trained using Ada-Boost classifier. Sliding window technique was then used to detect the presence of car and localizing it. For this various window sizes with different overlap percentages were used. The windows and overlap percentages chosen are shown in Table 4.2. The start and stop of sliding window along the x axis and y axis were also specified as shown in Table 4.3.Using this the vehicle was localized and bounding boxes were drew around it. The results obtained are shown in Figure 4.3 and Figure 4.4.

Table 4.2 Various window sizes and overlap percentages

SIZE OF THE WINDOW	**OVERLAP (%)**
64x64	85
80x80	80
96x96	70
128x128	60

Table 4.3 Details about window size, overlap, start and stop value

WINDOW SIZE	OVERLAP	Y START	Y STOP
64x64	85	400	464
80x80	80	400	480
96x96	70	400	612
128x128	60	400	660

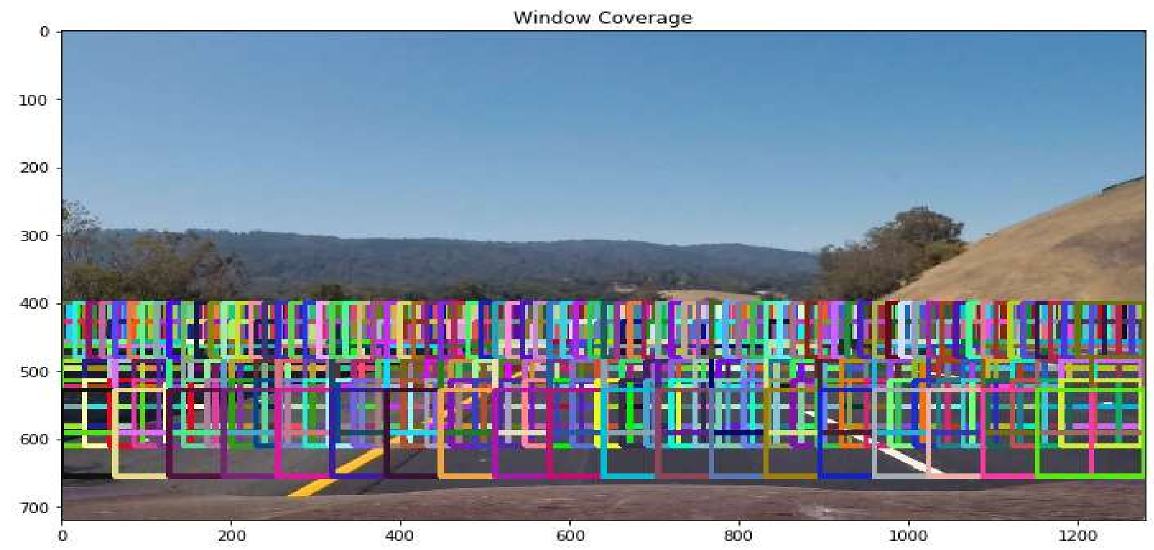


Figure 4.3 Window coverage

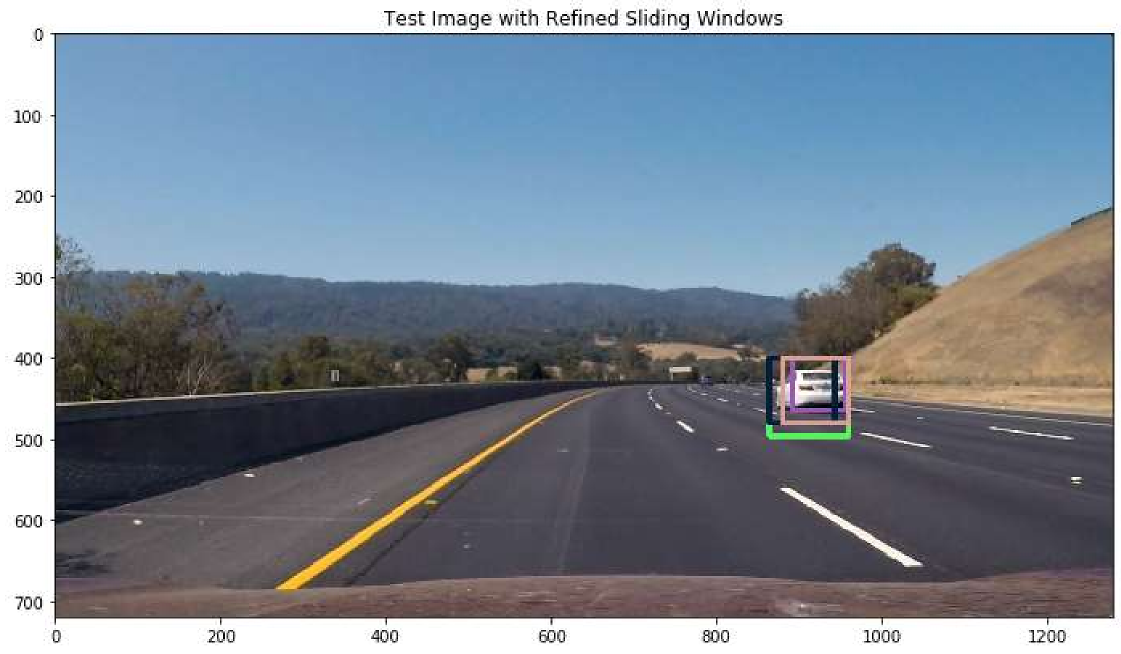
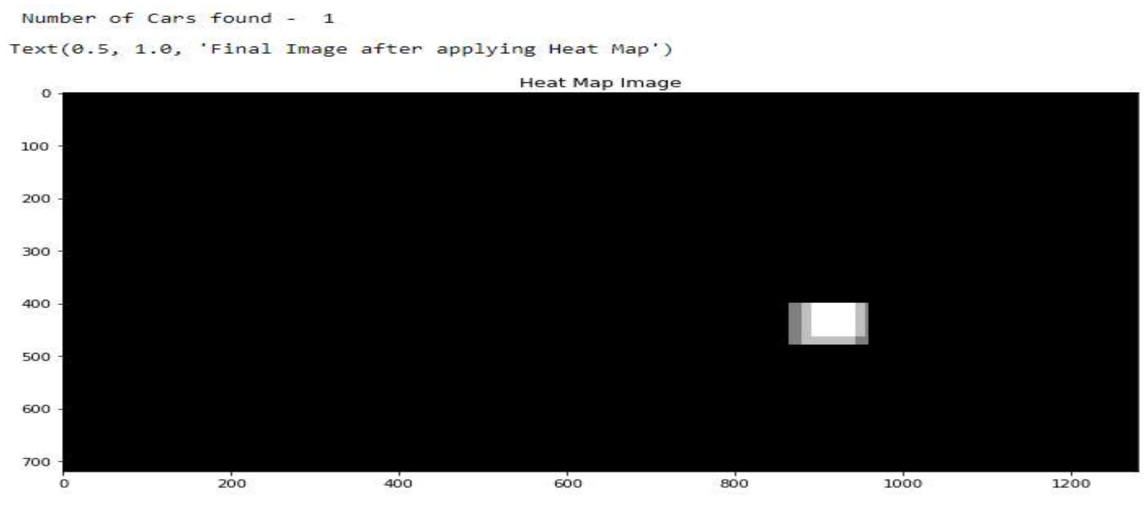


Figure 4.4 Refined sliding window for a test image

4.4 REFINED LOCALIZATION

As seen in the above sliding window approach that there are 3 boxes drawn around a single car which is undesirable. Thus optimization of the algorithm was done by using Heatmaps as shown in Figure 4.5. This finally gives the single refined window around the car shown in Figure 4.6.



```
Number of Cars found -  1
Text(0.5, 1.0, 'Final Image after applying Heat Map')
```

Figure 4.5 Heatmap application

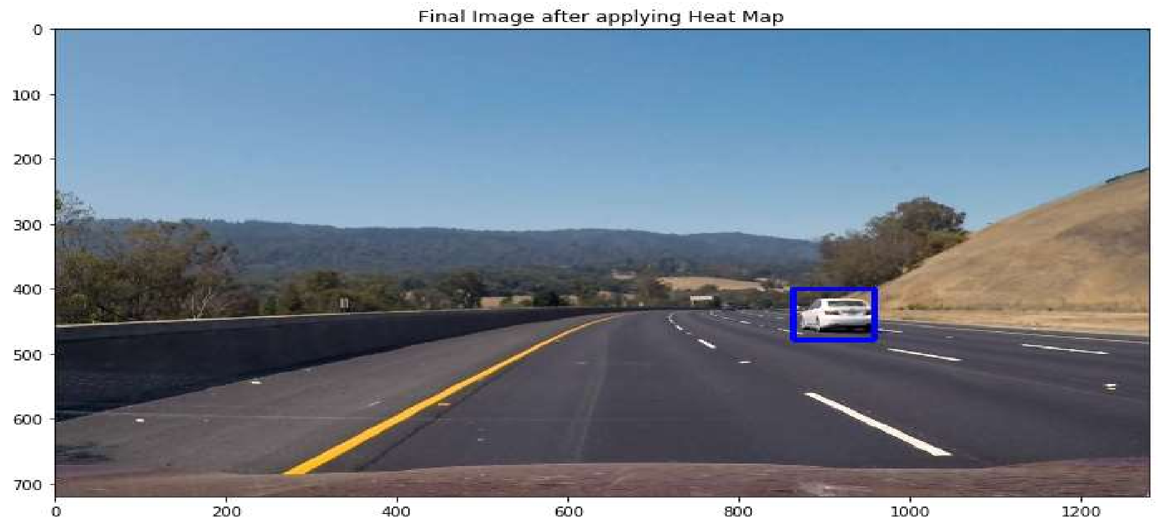


Figure 4.6 Refined box after application of heatmap

4.5 LOCALIZING A CAR IN A VIDEO

The procedure explained till now is only for images. As the input for us is a video, it is necessary to split it into frames first and then detection and localization be performed on that. Thus the output from the real time implementation is shown in Figure 4.7.

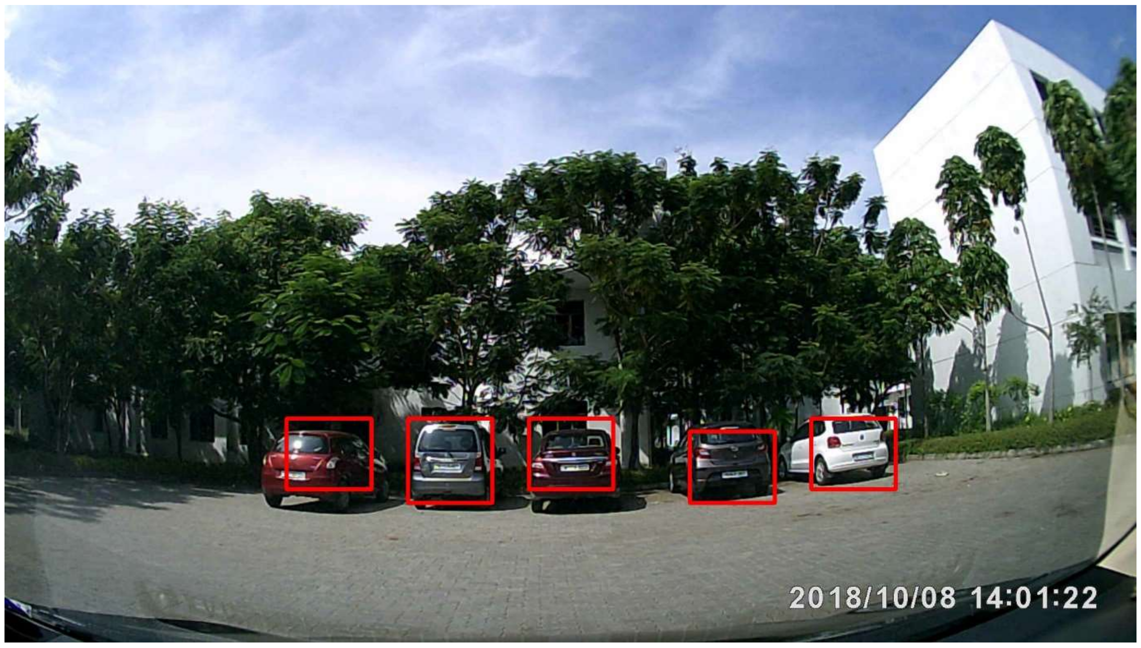
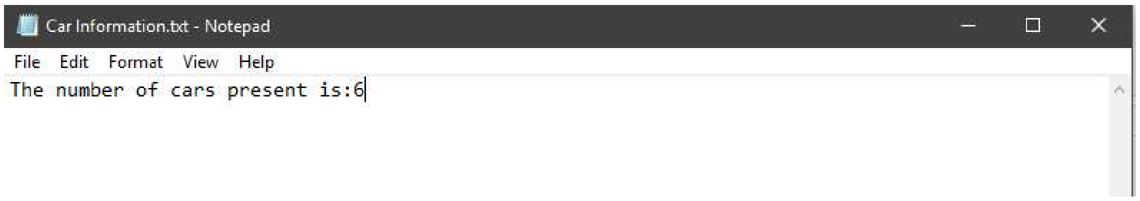


Figure 4.7 Localization of cars from a real time video

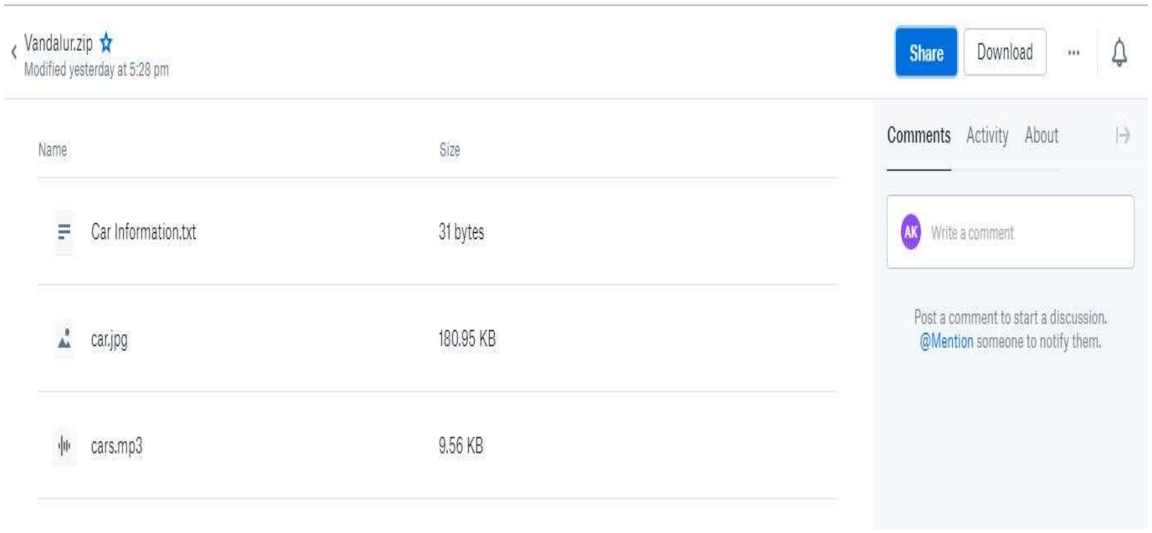
4.6 ZIP FILE UPLOADS

As the number of cars is known, an audio file and a text file containing the information about the same were created as shown in Figure 4.8. This will assist the driver by updating him about the traffic time to time. These are uploaded using a Zip file which will also contain a frame image too as in Figure 4.9.



```
The number of cars present is:6
```

Figure 4.8 Contents in the text file

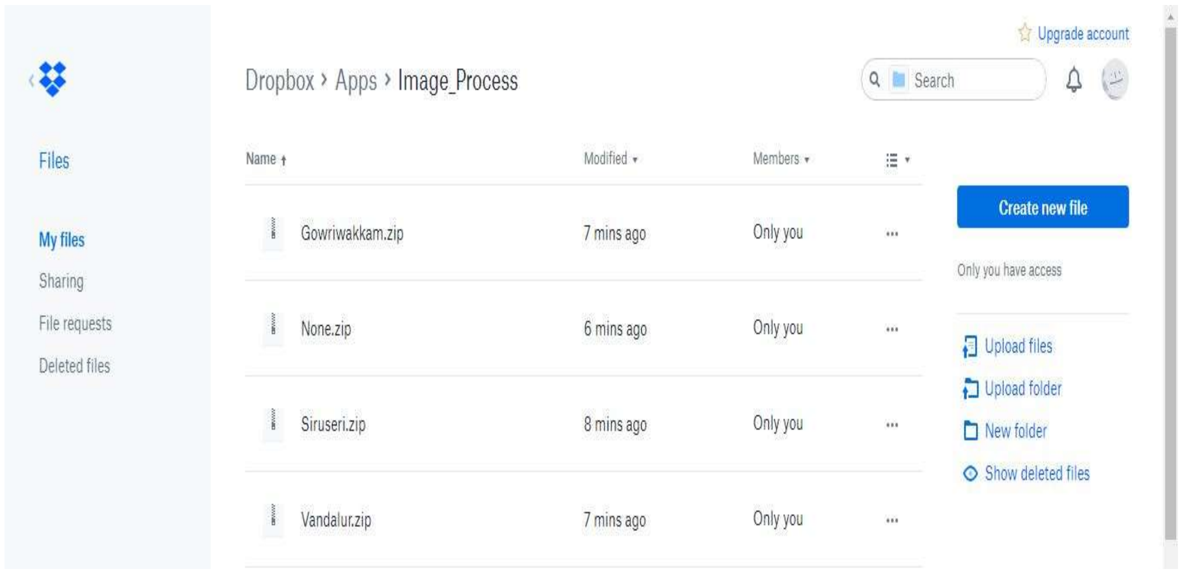

Vandalur.zip
Modified yesterday at 5:28 pm
Share
Download
Comments
Activity
About
AK
Write a comment
Post a comment to start a discussion.
@Mention someone to notify them.

Name	Size
Car Information.txt	31 bytes
car.jpg	180.95 KB
cars.mp3	9.56 KB

Figure 4.9 Files inside a Zip folder

4.7 DROPBOX UPLOAD

Finally Zip files containing information about traffic at various locations are uploaded to the Dropbox as shown in Figure 4.10. the zip file uploaded has the name of the place which is found using IP address. These files have to be deleted regularly in order to upload the files from time to time providing updates.



Name	Modified	Members
Gowriwakkam.zip	7 mins ago	Only you
None.zip	6 mins ago	Only you
Siruseri.zip	8 mins ago	Only you
Vandalur.zip	7 mins ago	Only you

Figure 4.10 Zip files uploaded to Dropbox

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 CONCLUSION

A system was developed to provide real time traffic updates on a common platform which is then retrieved by any person and used. The video obtained by camera was split into frames and applied to a pipeline that was developed for this purpose. This pipeline performed the task of localization of vehicles. HOG Descriptors and Adaboost classifiers were used to classify the object as a vehicle and gave maximum accuracy. These descriptors and classifiers were also trained with datasets to obtain greater accuracy. Heatmaps were implemented successfully to complete the localization of vehicles.

Once the localization was done, it was easier to calculate count of vehicles using labels. Using Google Text to speech converter the count was converted into an audio file. Finally, a zip file was created which combines an audio file, count of vehicles and an image of the vehicles in the traffic congestion. This zip file was uploaded to Dropbox which is retrieved by the user by signing in deletion of the current folders is scheduled properly to give room for new updates. The real time implementation of the process was done in Raspberry Pi. Thus, real time traffic updates were successfully provided to the user.

5.2. FUTURE WORK

Proposed system can be improved in certain areas which makes it more reliable. There are certain features which when improved makes the system better and gives good performance. The problems include occlusion, false positives and errors in location search. As a result of these drawbacks, there might be localization errors or count errors thus leading to reduction in accuracy. This requires new methods and powerful algorithms which serves the purpose.

- While detecting the vehicles, in some cases the algorithms might detect a non-vehicular object as a vehicle. These are called false positives. These errors can be minimized by training the algorithms more and also by improving the quality of the video frames. Using Deep learning also produces more accurate and reliable results.
- For getting the location, this project uses IP address of the device. IP address is used to obtain the latitude and longitude value internally which is then reverse coded to give the name of the location. But using IP address, limits the location to a Suburb. That is the location obtained is more general and cannot be narrowed down to more smaller parts. Also, sometimes the change in geo co-ordinates will be very less that even when location changes it might not be detected. Thus, it leads to errors in the data being uploaded. Google Maps on the contrary provide a very precise and pin-pointed locations. One can actually zoom in the maps and have a glance at the streets. So when, both are combined, that is implement the present system in Google Maps, this would serve as the best Intelligent Transport System. One can see the actual images of vehicles on any street however small it is and take proper decisions.
- Occlusion is a condition where the required object is blocked because of the presence of another object. While detecting vehicles, if one vehicle is blocked by another one, then it cannot be detected. Thus, while counting the number of vehicles present, it will be excluded. So, the count that the system uploads is faulty. It is just a minor problem when only one or two vehicles are missing. While in cases where vehicles are being blocked, the seriousness of occlusion increases. Uploading such faulty data will only make the situation worse. So, eliminating occlusion is to be more importance to improve the correctness of the data being uploaded.

REFERENCES

1. An, T. and Kim, M. (2010) ‘A New Diverse AdaBoost Classifier’, International Conference on Artificial Intelligence and Computational Intelligence, Sanya, pp.359-363.

2. Banerji, S., Sinha, A. and Liu, C. (2013) ‘HaarHOG: Improving the HOG Descriptor for Image Classification,’ *IEEE International Conference on Systems, Man, and Cybernetics*, Manchester, pp.4276-4281.

3. Chandel, H.S. and Vatta, S. (2015) ‘Vision based vehicle detection with occlusion handling’ International Journal of Scientific & Engineering Research, Vol.6, No.7, pp.548-555.

4. Cengil, E. and Cinars, A. (2017) ‘Comparison of HOG (Histogram of Oriented Gradients) and Haar Cascade Algorithms with A Convolutional Neural Network Based Face Detection Approach’, Vol.3, No.5, pp.244-255.

5. Dalal, N. and Triggs, B. (2005) ‘Histograms of oriented gradients for human detection,’ *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, pp. 886-893.

6. Fang, J., Zhou, Y., Yu, Y. and Du, S. (2017) ‘Fine-Grained Vehicle Model Recognition Using A Coarse-to-Fine Convolutional Neural Network Architecture,’ in *IEEE Transactions on Intelligent Transportation Systems*, Vol.18, No.7, pp.1782-1792.

7. Geurts, P., El Khayat, I. and Leduc, G. (2004) ‘A machine learning approach to improve congestion control over wireless computer networks,’ *Fourth IEEE International Conference on Data Mining (ICDM'04)*, Brighton, UK, pp.383-386.

8. Kachouane, M., Sahki, S., Lakrouf, M. and Ouadah, N. (2012) ‘HOG based fast human detection’, *24th International Conference on Microelectronics (ICM)*, Algiers, pp.1-4.

9. Kaiyang Zhong, Zhaoyang Zhang and Zhengyu Zhao. (2018) ‘Vehicle Detection and Tracking Based on GMM and Enhanced Camshift Algorithm’, Journal of Electrical and Electronic Engineering, Vol.6, No.2, pp.40-45.

10. Lingala, Thirupathi et al. (2014) ‘Traffic Congestion Control through Vehicle-to-Vehicle and Vehicle to Infrastructure Communication’, International Journal of Computer Science and Information Technologies, Vol.5, No.4, pp.5081-5084.

11. Malhi, M.H., Aslam, M.H., Saeed, F., Javed, O. and Fraz, M. (2011) ‘Vision Based Intelligent Traffic Management System’, *Frontiers of Information Technology*, Islamabad, pp. 137-141.

12. Mohandas, B.K. Liscano and Yang, O.W.W. (2009) ‘Vehicle traffic congestion management in vehicular ad-hoc networks’, *IEEE 34th Conference on Local Computer Networks*, Zurich, pp.655-660.

13. Sivaraman, S. and Trivedi, M.M. (2013) ‘Looking at Vehicles on the Road: A Survey of Vision-Based Vehicle Detection, Tracking, and Behavior Analysis,’ in *IEEE Transactions on Intelligent Transportation Systems*, Vol.14, No.4, pp.1773-1795.

14. Wang, Y. and De Lin, C. (2007) ‘Learning by Bagging and Adaboost based on Support Vector Machine,’ 5th IEEE International Conference on Industrial Informatics, Vienna, pp.663-668.

15. Wang, Y. Han, P. Lu, X. Wu, R. and J. Huang, (2006) ‘The Performance Comparison of Adaboost and SVM Applied to SAR ATR,’ *CIE International Conference on Radar*, Shanghai, pp.1-4.

16. Zi Yang. Lilian, S.C. Pun-Cheng. (2018) ‘Vehicle detection in intelligent transportation systems and its applications under varying environments: A review, Image and Vision Computing’,Vol.69, pp.143-154.