Vehicle Detection & Speed Tracking using Image processing

A Project presented to the National University in partial fulfillment of the requirement for
The degree of Bachelor of Science (Hon's) in Computer Science & Engineering

Supervised By

Mizanur Rahman

Lecturer (Dept. of CSE)

Daffodil Institute of IT

Submitted By

Mehedi Hasan Nayem

Registration no: 17502005087

Session: 2017-2018

Department of Computer Science & Engineering
Daffodil Institute of IT, Dhaka
Under National University, Bangladesh

September, 2023

DECLARATION

We pledge that the project work titled “**Vehicle Detection & Speed Tracking using Image Processin**” being submitted in partial fulfillment for the degree of B.Sc. (Hon’s) in Computer Science & Engineering is the original work carried out by me. It has not formed the part of any other project work submitted for any degree or diploma, either in this or any other University.

Submitted by :

Mehedi Hasan Nayem
Registration no: 17502005087
Session: 2017-2018

APPROVAL

The Project “**Vehicle Detection & Speed Tracking using Image Processing**” is submitted to the Department of Computer Science & Engineering, DIIT under National University of Bangladesh in absolute fulfillment of the requirements for the degree of Bachelor of Science (Hon’s) in Computer Science and Engineering and approved as to its style and content.

Examiner

Examiner

Project Supervisor
Mizanur Rahman
Lecturer (Dept. of CSE)
Daffodil Institute of IT

Md. Imran Hossain
Head
Department of CSE
Daffodil Institute of IT

ACKNOWLEDGEMENTS

I would like to express my profound gratitude to Almighty Allah. With the blessing of Almighty Allah, I have successfully planned my project.

My sincere thanks to **Prof. Dr. Mohammed Shakhawat Hossain,** Principal of DIIT who has allowed me to do this project and encouragement given to me.

I express my gratitude to our supervisor **Mizanur Rahman**, Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.Also, thanks for his valuable guidance and support to meet the successful completion of my project.

My heartiest thanks **Md. Imran Hossain**, Head of Department, Computer Science & Engineering, DIIT, Dhaka, for his patronage and giving me an opportunity to undertake this Project.

I express my gratitude to Our Batch Co-ordinator **Saidur Rahman**, Senior Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Poly Bhoumik,** Senior Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

I express my gratitude to **Nusrhat Jahan Sarkar,** Lecturer, DIIT, Dhaka, for having provided us the facilities to do the project successfully.

Last but not the least, I extend my sincere thanks to my family members and my friends for their constant support throughout this project.

ABSTRACT

The Vehicle Detection & Speed Detection Project is a computer vision project that involves detecting and tracking vehicles in real-time from a video feed or CC Tv Camera. The project utilizes deep learning algorithms, such as Convolutional Neural Networks (CNNs), Object Tracking etc to detect the presence of vehicles in a frame and to locate their position. Once the vehicles are detected, the project tracks their movement over time to estimate their speed and direction. The ultimate goal of the project is to provide accurate and reliable vehicle detection and tracking for use in various applications, such as traffic management, surveillance, and autonomous driving systems.

The vehicle detection project aims to develop a system that can automatically detect and track vehicles in real-time using computer vision and machine learning techniques. The system employs a combination of object detection algorithms, such as YOLO, SSD, or Faster R-CNN, and image processing techniques, such as edge detection, segmentation, and feature extraction, to identify and track vehicles in a video stream. The system can also perform various tasks, such as vehicle classification, counting, and speed estimation, to provide useful information for traffic management and surveillance applications. The project requires extensive data collection and preprocessing, as well as model training and optimization, to achieve high accuracy and real-time performance.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

CHAPTER 1 : INTRODUCTION

1.1 Introduction :

Detection of vehicle and tracking of speed if the crucial part of town planning. One of the significant applications of video-based supervision systems is the traffic surveillance. So, for many years the researches have investigated in the Vision-Based Intelligent Transportation System (ITS), transportation planning and traffic engineering applications to extract useful and precise traffic information for traffic image analysis and traffic flow control like vehicle count, vehicle trajectory, vehicle tracking, vehicle flow, vehicle classification, traffic density, vehicle velocity, traffic lane changes, license plate recognition, etc. Vehicle speed detection system finds the speed of the vehicle accurately and helps to discover, if any vehicle is moving with the speed greater than the speed limit in real time [10]. It can also be used to generate e-challan. It is cost effective and can be used 24*7. The main concern of the authority dedicated to the road safety is vehicle speed control. This is the important reason to control speed of the vehicles to save the lives of people. There are various application areas of this system like public road safety agencies, private road safety department, government. The first priority of most of the government all over the world is road safety. To overcome this problem and to decrease the death rate due to road accident the development of new traffic enforcement is required. This system captured the image of the vehicle and find the speed of the vehicle.

1.2 Objective :

Vehicle detection and tracking applications play an important role for civilian and military applications such as in highway traffic surveillance control, management and urban traffic planning. Vehicle detection process on road are used for vehicle tracking, counts, average speed of each individual vehicle, traffic analysis and vehicle categorizing objectives and may be implemented under different environments changes. In this review, we present a concise overview of image processing methods and analysis tools which used in building these previous mentioned applications that involved developing traffic surveillance systems. More precisely and in contrast with other reviews, we classified the processing methods under three categories for more clarification to explain the traffic systems.

1.3 Benefits :

There are several benefits of a vehicle detection project that can be used for traffic monitoring, safety applications, and more. Here are some potential benefits:

- Improved traffic flow: By monitoring the number of vehicles on the road and their speed, a vehicle detection system can help improve traffic flow by identifying areas where congestion is likely to occur and suggesting alternate routes or traffic management strategies.
- Enhanced safety: A vehicle detection system can help improve safety on the roads by detecting vehicles that are speeding or driving recklessly, and alerting law enforcement or other authorities to take action.
- Reduced fuel consumption: By optimizing traffic flow and reducing congestion, a vehicle detection system can help reduce fuel consumption and emissions, which can be beneficial for the environment.
- Cost savings: A vehicle detection system can help reduce the cost of traffic management by providing real-time data on traffic conditions, allowing authorities to allocate resources more efficiently and reduce unnecessary expenditures.

- Improved emergency response: In the event of an accident or other emergency, a vehicle detection system can help emergency responders by providing real-time data on traffic conditions and the location of vehicles, allowing them to respond more quickly and effectively.

Overall, a vehicle detection project has the potential to provide significant benefits for traffic management, safety, and the environment, making it a valuable tool for transportation authorities and other organizations.

1.4 Limitation :

There are several limitations of a vehicle detection project that should be considered. Here are some potential limitations:

- Limited accuracy: The accuracy of a vehicle detection system may be affected by various factors such as lighting conditions, weather, and camera positioning, which can impact the ability of the system to accurately detect and track vehicles.
- Limited coverage: The coverage of a vehicle detection system may be limited by the placement of cameras or sensors, which can impact the ability of the system to detect and track vehicles in certain areas.
- Limited functionality: A vehicle detection system may be designed to detect and track vehicles, but may not have additional functionality such as recognizing license plates, identifying specific types of vehicles, or detecting pedestrians or cyclists, rickshaw.

1.5 Features :

A vehicle detection and speed tracking project can have several features, depending on the specific requirements and goals of the project. Here are some possible features:

- Real-time detection and tracking of vehicles: The system can detect and track vehicles in real-time using computer vision techniques, such as object detection and tracking.
- Vehicle speed calculation: The system can calculate the speed of each vehicle by analyzing its position across multiple frames.
- Speed threshold detection: The system can be configured to trigger an alert if a vehicle exceeds a specified speed limit.

- Multiple camera support: The system can support multiple cameras to cover larger areas and capture more data.
- User interface: The system can have a user interface to allow users to view real-time data and generate reports.
- Data storage and retrieval: The system can store data on vehicle positions and speeds, allowing it to generate reports and perform analysis.
- Integration with other systems: The system can be integrated with other systems, such as traffic management systems or emergency response systems.
- Customization: The system can be customized to meet specific requirements and goals, such as detecting specific types of vehicles or monitoring specific areas of interest.

Overall, a vehicle detection and speed tracking project can have several features that make it a valuable tool for traffic management, safety, and other applications. The specific features will depend on the requirements of the project and the needs of the users.

CHAPTER 2
BACKGROUND STUDY

CHAPTER 2 : BACKGROUND STUDY

2.1 Introduction to Background Processing :

In this project, we have used the methods discussed in Dimililer et al. (2020) paper of Vehicle detection and tracking to built a system using OpenCV and Python for both images and videos that is able to detect and track the vehicles automatically. We have used a dataset consisting of 8792 vehicle images and 8968 non-vehicle images from a combination of the GTI vehicle image database and the KITTI vision benchmark suite. We have utilized Histogram of Oriented Gradients (HOG) feature extraction on a labelled training set and trained one of Support vector machines (SVMs) classifier.

We have implemented a sliding-window technique and used the trained classifier to search for vehicles in images. From the object detection result, we have assigned an object tracker, the tracker would be following the target by re-detecting it in the sequence frame following the first point of the target. The identification of vehicles is shown by drawing a bounding box around it.

2.2 Feasibility :

Vehicle detection using computer vision techniques such as object detection and tracking is a well-established field, and there are many commercially available systems that can accurately detect and track vehicles. Therefore, it is feasible to implement a vehicle detection project.

However, the feasibility of a vehicle detection project will depend on several factors such as the specific requirements of the project, the available resources (including budget and personnel), and the infrastructure in the area where the system will be deployed.

For example, the accuracy of a vehicle detection system can be affected by lighting conditions, weather, and camera positioning. Therefore, it may be necessary to carefully consider these factors during the design and implementation of the system.

2.3 Applicability of Our System :

A vehicle detection project can have a wide range of practical applications in various industries. Here are some examples of how a vehicle detection project can be applied:

- Traffic management: A vehicle detection project can be used to monitor traffic flow, detect congestion, and optimize traffic signals.
- Safety and security: A vehicle detection project can be used to detect vehicles in restricted areas, monitor vehicle speeds and behaviors, and detect accidents or incidents.
- Parking management: A vehicle detection project can be used to monitor parking spaces and provide real-time information about available parking spots.
- Logistics and transportation: A vehicle detection project can be used to track the movement of vehicles, monitor cargo, and optimize delivery routes.
- Environmental monitoring: A vehicle detection project can be used to monitor emissions from vehicles and identify areas of high pollution.
- Autonomous vehicles: A vehicle detection project can be used as part of an autonomous driving system to detect and track other vehicles on the road.

Overall, a vehicle detection project can have numerous practical applications in various industries, and its applicability will depend on the specific requirements and goals of the project.

CHAPTER 3
LITERATURE REVIEW

CHAPTER 3 : LITERATURE REVIEW

3.1 Introduction to Literature Survey :

Recognition of change in location of a non-stationary object in a series of images captured of a definite region at equal intervals of time is considered as an interesting topic in computer vision. A plethora of application from multiple nuances are deployed to function in real time environments; video surveillance, identifying objects lying underwater, diagnosing abnormalities in patient and providing proper treatment in the medical department. Among these, one of the applications is detection of vehicle in traffic and identifying the speed of the vehicle.

However, there are certain factors which should be considered for detection of constantly moving vehicles at every interval of time. It mainly comprises of three techniques to detect a vehicle namely:

1) Background Subtraction Methods

2) Feature Based Methods

3) Frame Differencing and motion-based methods

3.2 Background Subtraction Methods

The method of retrieval of a mobile object from a definite image (fixed background) is called background subtraction and the retrieved object is the resulted as threshold of image differencing [1]. This technique is pre dominantly used in detection of vehicle in an image frame. However the results are affected in poor lighting or bad climatic conditions and acts as a drawback to this method. BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.

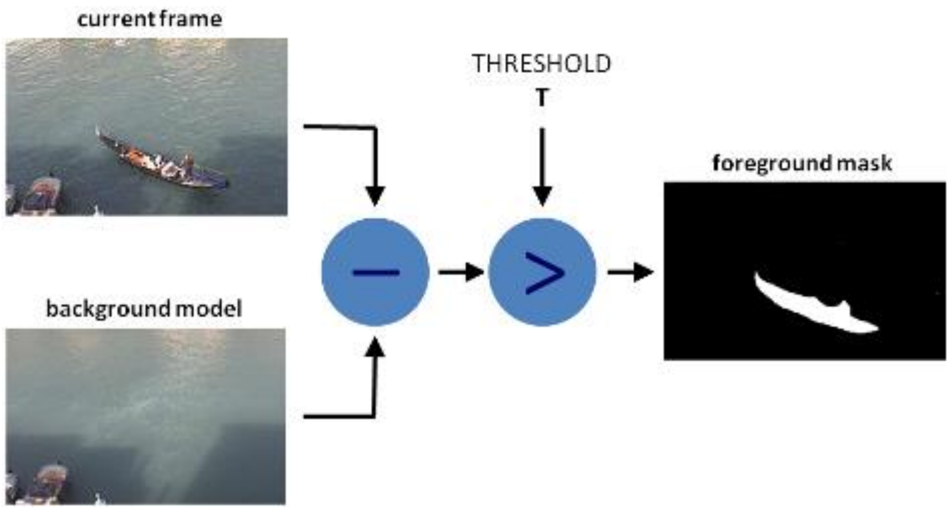


Figure 1 : Background Subtraction Methods

3.3 Feature Based Methods

This technique of identifying image displacements which are easiest to interpret is feature based modelling. The technique helps in identifying edges, corners and other structures in an image which are restricted properly in a two-dimensional plane and trace these objects as they transit between multiple frames. This technique comprises of two stages; finding the features in multiple images and matching these features between the frames: Stage 1: In this step, the features are found in a series of two or more images. If carried out perfectly, with no overhead cost; it may work efficiently with less overload and reduce the extraneous information to be processed Stage 2: Features found in stage 1 are matched between the frames. Under most common scenarios, two frames are used and two sets of features are matched to a resultant single set of motion vectors. These features in one frame are used as seed points which use other techniques to determine the flow.

3.4 Frame Differencing & Motion-based Methods :

Frame differencing is a method of finding the difference between two consecutive images from a sequence of images to segregate the moving object (vehicle) from the background. If there is a change in pixel values, it implies that there was a change in position in the two image frames.The motion rectification step of detecting a vehicle in a trail of images by alienating the moving objects, also known as blobs based on its speed, movement and orientation**.**It is recommended to use an intraframe, interframe and tracking levels as frameworks to identify and control the motion of vehicles in frame. Using quantitative evaluation this paper illustrated that interframe and intraframe can be used to control and handle partially detected images and tracking level can be used to handle full blocked images efficient.

3.5 Camera Calibration Approacehs

Measuring the vehicle speed and precision of vehicle tracking methods rely on well camera calibration performance, and the camera calibration setup may be done in semi-automatically matter or by hand. Camera calibration is a vital procedure for well video-based surveillance systems.

A new automatic method for segmenting and tracking vehicles applied on a video taken by camera at low angle level relatively to the ground on highway road [3]. In this paper, expectation of high features is calculated by joining of region-based grouping procedure, background subtraction, projective transformation and using of plumb line projection (PLP).

A novel automatic traffic system which uses 2D spatio-temporal images has suggested by [6]. This system uses a TV camera to keep track of vehicles for the highway traffic within two slice windows for each traffic lane. The purpose of this system was classifying the passing vehicles by using these 3D measurements (height, width and length) in addition to count the vehicles and assessment their speeds. Also, this system showed a robust performance when it tested under different light situations involving vehicles lights at night and shades in the day (Figure 2).

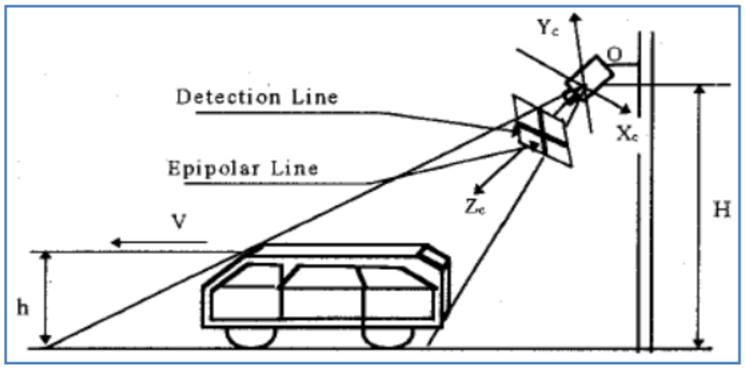


Figure 2 : Camera Calibration Approaches

3.6 Vehicle Tracking Approaches :

The object tracking in video processing is an important step to tracking the moving objects in visual-based surveillance systems and represents a challenging task for researchers [43]. To track the physical appearance of moving objects such as the vehicles and identify it in dynamic scene, it has to locate the position, estimate the motion of these blobs and follow these movements between two of consecutive frames in video scene [44]. Several vehicle tracking methods have been illustrated and proposed by several researchers for different issues, it consists *of:*

1. Region-Based Tracking Methods
2. Contour Tracking Methods
3. 3D Model-Based Tracking Methods

3.6.1 Region-Based Tracking Methods

In these methods, the regions of the moving objects (blobs) are tracked and used for tracking the vehicles. These regions are segmented from the subtracting process between the input frame image and prior stored background image. A proposed research paper introduced a model-based automobile recognizing, tracking and classification which is efficiently working under most conditions [17]. The model provided position and speed knowledge for each vehicle as long as it is visible, in addition, this model worked on series of traffic scenes recorded by a stable camera for automobiles monocular images. The processing algorithms of this model represented of three levels: raw images, region level, and vehicle level.

3.6.2 Contour Tracking Methods

These methods depend on contours (the boundaries of vehicle) of vehicle in tracking vehicle process [39]. The authors [46]have proposed a novel real time traffic supervision approach which employs optical movement and uncalibrated camera parameter knowledge to detect a vehicle pose in the 3D world. In this paper, the proposed approach uses two new techniques: color contour based matching and gradient based matching, and it showed well results when it tested for tracking, foreground object detection, vehicle recognition and vehicle speed assessment methods. A real-time vehicles tracking and classification technique on highway have offered by [8]. A few traffic criterions (lane change recognition, vehicle numbering and vehicle classification) are extracted by above technique. In addition, the proposed technique supported the occlusion detection and tracking which caused from multiple vehicles poses in the crowding situation.

3.6.3 3D Model-Based Tracking Methods

An occlusion detection approach based on generalized deformable model. In this paper, the occlusion of vehicles detection process used a 3D solid cuboid form with up to six vertices, and this cuboid used to fit any different types and sizes of vehicle images by changing the vertices for a best fit. Therefore, vehicle detection, segmentation and tracking can be achieved efficiently due to changes in the region proportion, prototype width and height with consideration to previous images. A unified multi-vehicle tracking and categorization system for various types of vehicles such as motorcycles, cars, light trucks and heavy trucks on highway and windy road video sequences has recommended by [48]. In this paper, a vehicle anisotropic distance measurement achieved through the 3D geometric shape of vehicles.

Chapter 4

REQUIREMENTS

CHAPTER 4 : REQUIREMENTS

4.1 Introduction to Requirements

The process to gather the software requirements from clients and analyze and document them is known as requirement engineering. The goal of requirement engineering is to develop and maintain a sophisticated and descriptive 'System Requirements Specification' document.

4.1.1 Hardware Requirements

- Intel Core i5
- 8GB RAM
- Camera

4.1.2 Software Requirements

● Pycharm powered by Jetbrains / Visual Studio

● OpenCV

● Tensorflow

● Python IDE (Python v3.10)

● Object Detection API

4.2 Python and pip

Python is a high-level, interpreted programming language that is widely used for developing a wide range of applications, including web development, scientific computing, artificial intelligence, data analysis, and more. The most important thing is that it is an interpreted language which means that the written code has not been translated into a computer-readable format at execution time whereas, the major programming languages do this translation before the program runs. This type of language is also known as "scripting language" because it is intended for use for little projects.

Pip is a package manager for Python that helps you easily install, manage, and update third-party packages and libraries that are not included in the standard Python library. It allows you to install and manage packages from the Python Package Index (PyPI) and other repositories.

4.3 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. It is an open-source computer vision and machine learning software library that provides various tools and algorithms for image and video processing, object detection, feature extraction, and more.

OpenCV is written in C++, but it also provides bindings for various programming languages, including Python, Java, and MATLAB. It can run on various operating systems, including Windows, Linux, macOS, Android, and iOS. It is a powerful tool for building and training machine learning models, particularly neural networks. TensorFlow provides a wide range of tools and APIs for developing machine learning models.

4.4 TensorFlow

Tensor operations: TensorFlow represents data as tensors, and provides a set of operations for manipulating them, including arithmetic operations, matrix multiplication, and more.

High-level APIs: TensorFlow provides high-level APIs for building common machine learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and more.

Low-level APIs: TensorFlow also provides low-level APIs for building custom machine learning models, including defining custom loss functions, creating custom layers, and more.

GPU support: TensorFlow can run on GPUs, which can significantly speed up the training of machine learning models.

Distributed training: TensorFlow can distribute the training of machine learning models across multiple machines, allowing for faster training and larger models.

4.5 TensorFlow Object Detection API:

The TensorFlow Object Detection API is a set of tools and libraries built on top of TensorFlow that allows developers to easily build, train, and deploy object detection models. Object detection is the task of detecting and localizing objects in an image or video, and is a fundamental problem in computer vision.

The TensorFlow Object Detection API provides pre-trained models that can detect a wide variety of objects, such as people, animals, vehicles, and more. It also provides tools for training custom object detection models on new datasets, as well as tools for evaluating and fine-tuning existing models.

Pre-trained models: The API includes several pre-trained models that can be used for detecting objects in images or videos, including the popular SSD (Single Shot Detector) and Faster R-CNN (Region-based Convolutional Neural Network) models.

Custom training: The API includes tools for training custom object detection models on new datasets, which can be useful for detecting objects that are not covered by the pre-trained models.

Data augmentation: The API includes tools for augmenting training data, such as randomly flipping or rotating images, which can help improve the performance of the trained models.

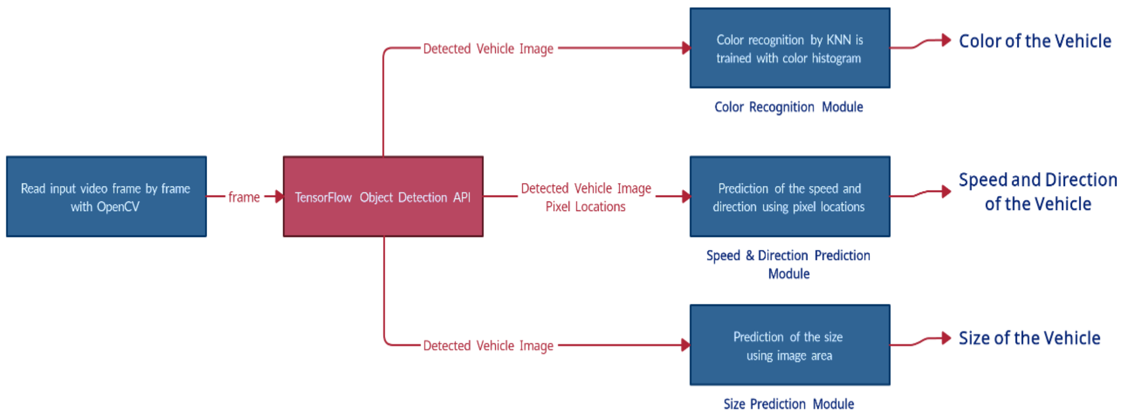
Distributed training: The API can distribute the training of object detection models across multiple GPUs or machines, which can significantly speed up the training process.

CHAPTER 5
DESIGN OF THE SYSTEM

CHAPTER 5 : DESIGN OF THE SYSTEM

5.1 System Architecture :

- Vehicle detection and classification have been developed using TensorFlow Object Detection API, see for more info.
- Vehicle speed prediction has been developed using OpenCV via image pixel manipulation and calculation, see for more info.
- Vehicle color prediction has been developed using OpenCV via K-Nearest Neighbors Machine Learning Classification Algorithm is Trained Color Histogram Features, see for more info.



Read input video frame by frame with OpenCV
frame
TensorFlow Object Detection API
Detected Vehicle Image
Color recognition by KNN is trained with color histogram
Color Recognition Module
Color of the Vehicle
Detected Vehicle Image Pixel Locations
Prediction of the speed and direction using pixel locations
Speed & Direction Prediction Module
Speed and Direction of the Vehicle
Detected Vehicle Image
Prediction of the size using image area
Size Prediction Module
Size of the Vehicle

Figure 3 : System Architecture

5.2 Workflow for Processing Input:

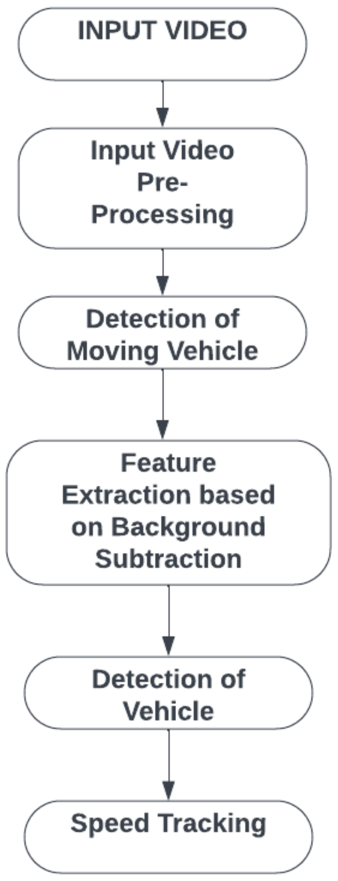

INPUT VIDEO
Input Video Pre-Processing
Detection of Moving Vehicle
Feature Extraction based on Background Subtraction
Detection of Vehicle
Speed Tracking

Figure 4 : Workflow Diagram

5.3 Block Diagram of Vehicle Detection :

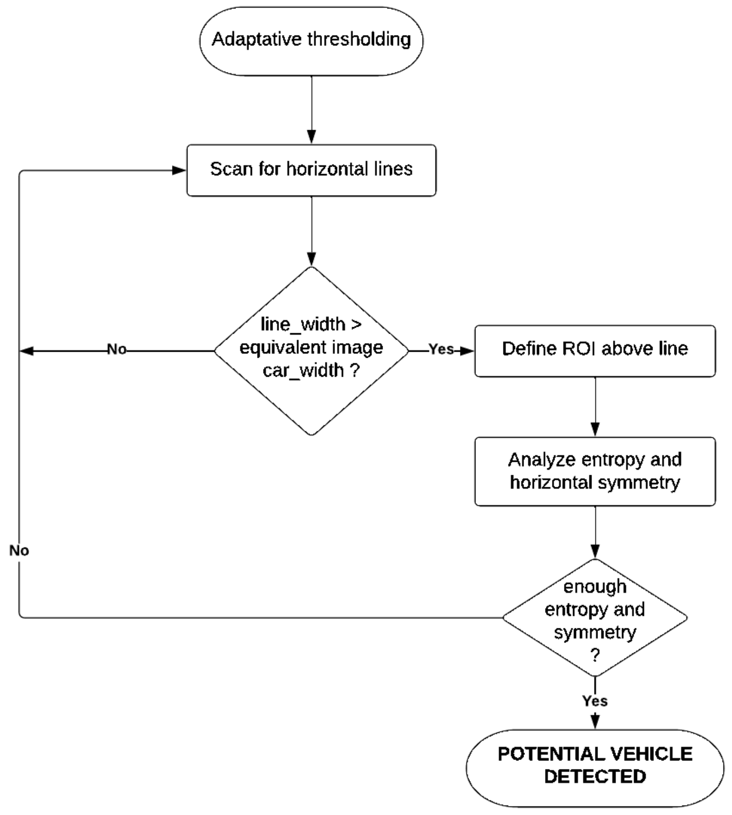

Adaptative thresholding
Scan for horizontal lines
line_width > equivalent image car_width ?
No
Yes
Define ROI above line
Analyze entropy and horizontal symmetry
No
enough entropy and symmetry ?
Yes
POTENTIAL VEHICLE DETECTED

Figure 5 : Block Diagram

5.4 Tracker

Source video is read frame by frame with OpenCV. Each frames is processed by "SSD with Mobilenet" model is developed on TensorFlow. This is a loop that continue working till reaching end of the video. The main pipeline of the tracker is given at the above Figure.

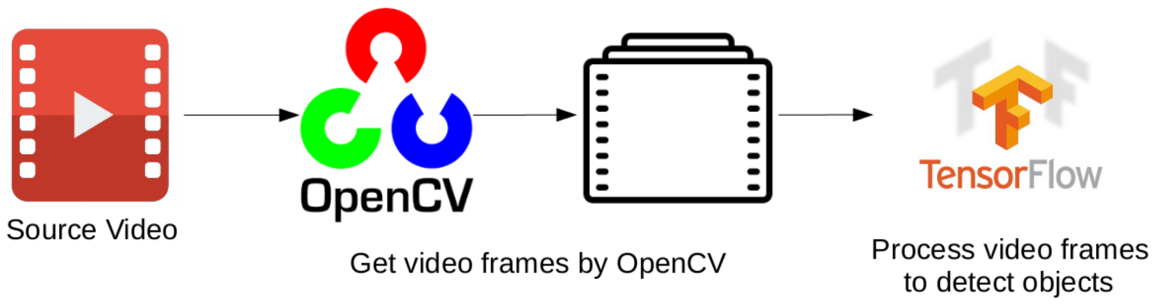


Figure 6 : Tracker

5.5 Model

By default we use an "SSD with Mobilenet" model in this project. You can find more information about SSD in here. See the detection model zoo for a list of other models that can be run out-of-the-box with varying speeds and accuracies.

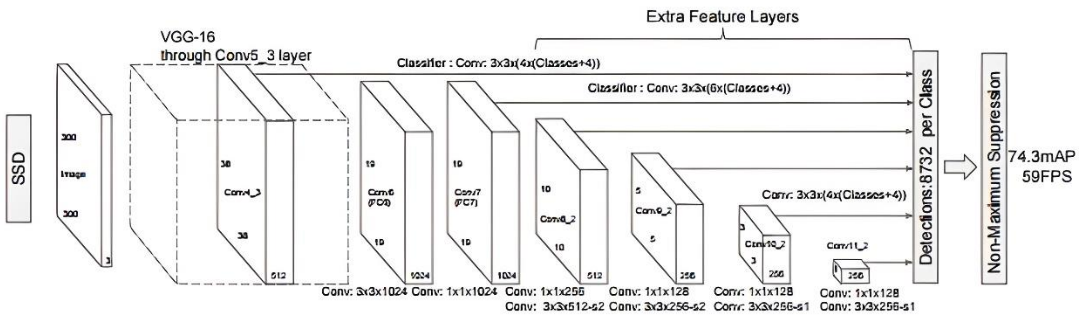


Figure 7 : Model

SSD MobileNet is a real-time object detection model that combines the Single Shot Detector (SSD) architecture with a MobileNet neural network. The model is designed to be lightweight and efficient, making it well-suited for deployment on mobile devices and embedded systems.The SSD architecture is a type of object detection model that can detect objects in an image with a single forward pass through a neural network. It does this by predicting a set of bounding boxes and class probabilities for each anchor point in the image, and then using non-maximum suppression to remove overlapping detections.

5.6 Multi-Object Tracking Diagram

Multi-object tracking is a computer vision technique used to track multiple objects simultaneously in a video or image sequence. The goal is to identify and track objects of interest over time, while maintaining the identity of each object and estimating its position, velocity, and other properties.

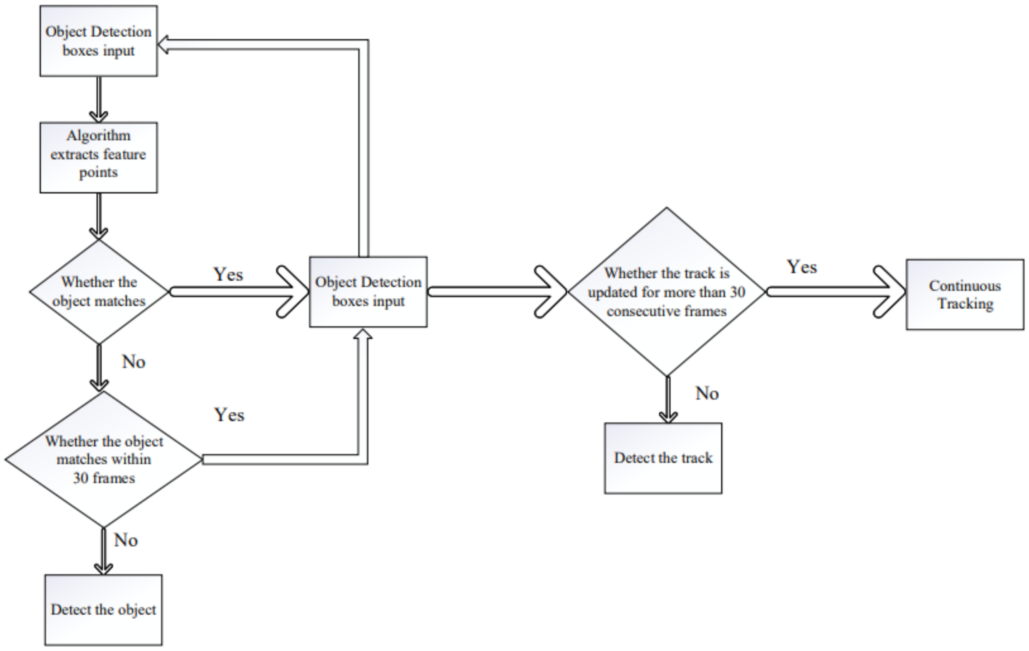
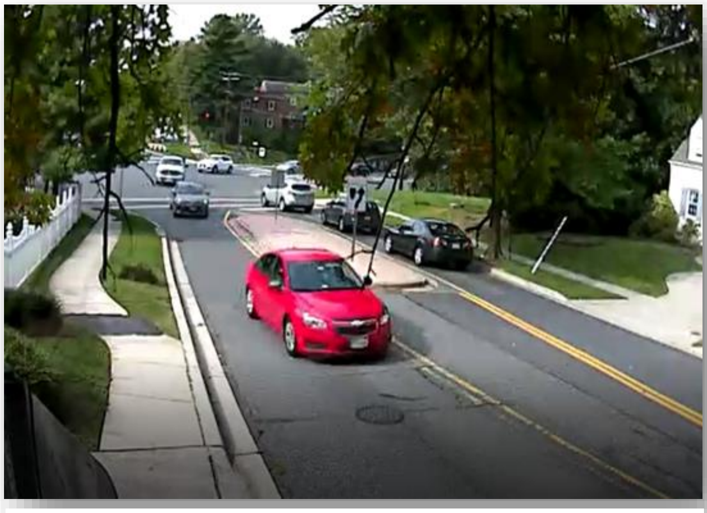

Object Detection boxes input
Algorithm extracts feature points
Whether the object matches
Yes
No
Object Detection boxes input
Whether the object matches within 30 frames
Yes
No
Detect the object
Whether the track is updated for more than 30 consecutive frames
Yes
No
Continuous Tracking
Detect the track

Figure 8 : Multi-Object Tracking

CHAPTER 6
IMPLEMENTATION

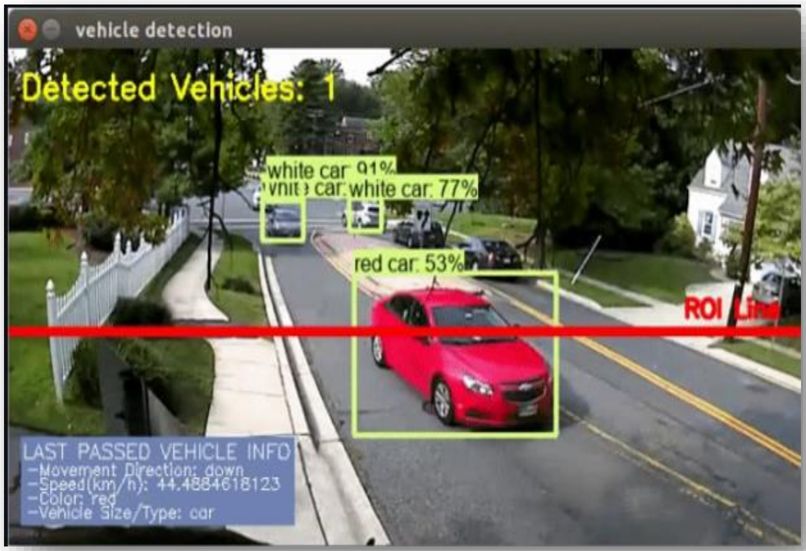
CHAPTER 6 : IMPLEMENTATION

6.1 Input Video



Source Video 1

6.2 Output Video

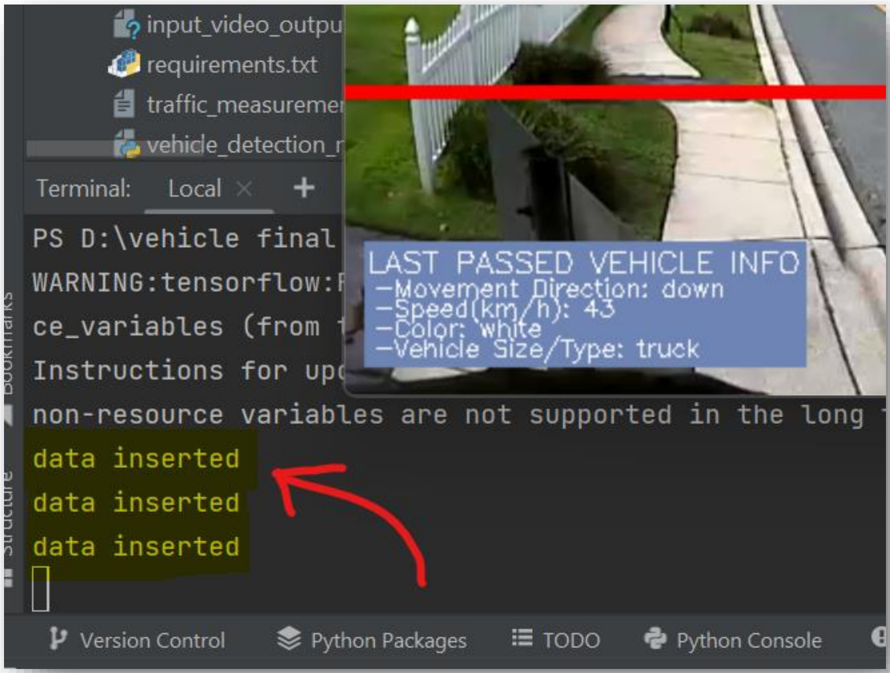


```
Detected Vehicles: 1
```

```
LAST PASSED VEHICLE INFO
 -Movement Direction: down
 -Speed(km/h): 44.4884618123
 -Color: red
 -Vehicle Size/Type: car
```

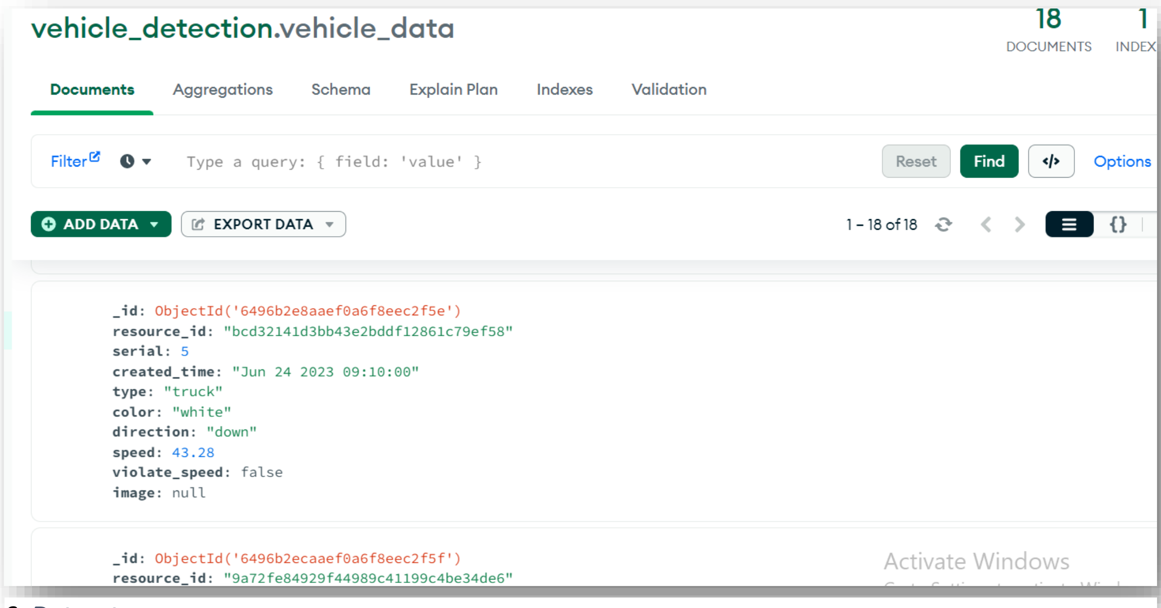
Output Result 1

6.3 Data Base connections



```
LAST PASSED VEHICLE INFO
-Movement Direction: down
-Speed(km/h): 43
-Color: white
-Vehicle Size/Type: truck
```

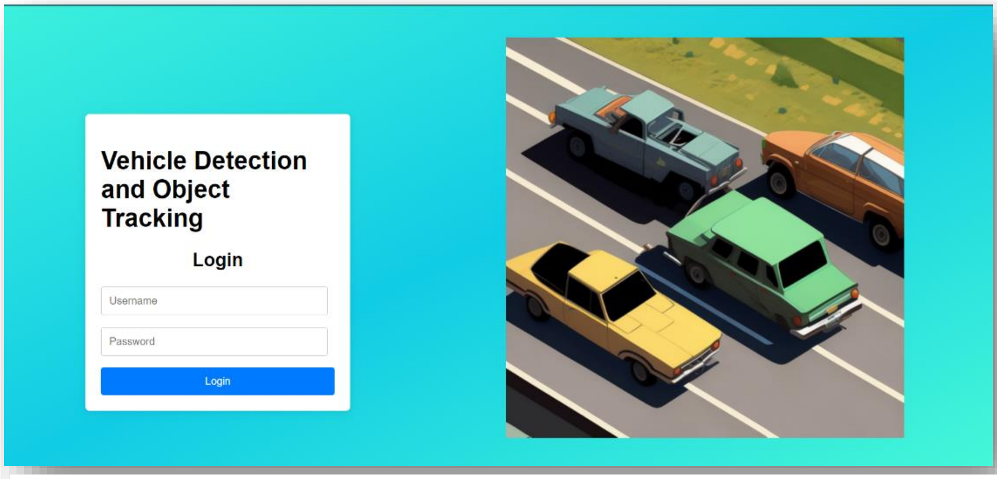
```
PS D:\vehicle final
WARNING:tensorflow:F
ce_variables (from t
Instructions for upd
non-resource variables are not supported in the long t
data inserted
data inserted
data inserted
```



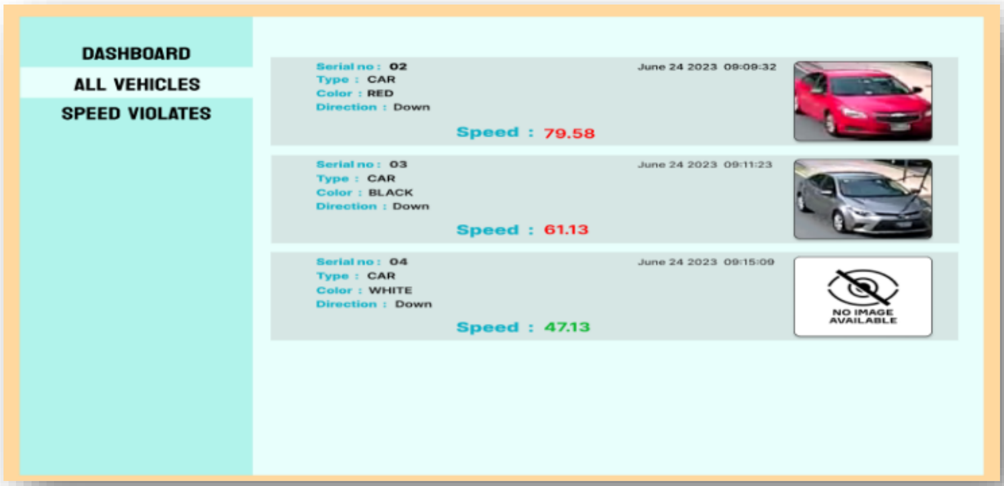
```
_id: ObjectId('6496b2e8aaef0a6f8eec2f5e')
resource_id: "bcd32141d3bb43e2bddf12861c79ef58"
serial: 5
created_time: "Jun 24 2023 09:10:00"
type: "truck"
color: "white"
direction: "down"
speed: 43.28
violate_speed: false
image: null

_id: ObjectId('6496b2ecaaef0a6f8eec2f5f')
resource_id: "9a72fe84929f44989c41199c4be34de6"
```

9. Dataset

6.4 Web-App view



10. Login Page



Serial no : 02
Type : CAR
Color : RED
Direction : Down
June 24 2023 09:09:32
Speed : 79.58

Serial no : 03
Type : CAR
Color : BLACK
Direction : Down
June 24 2023 09:11:23
Speed : 61.13

Serial no : 04
Type : CAR
Color : WHITE
Direction : Down
June 24 2023 09:15:09
Speed : 47.13
NO IMAGE AVAILABLE

11. Index page

CHAPTER 7

CONCLUSION AND RECOMMENDATION

CHAPTER 7 : CONCLUSION AND RECOMMENDATION

This project provides a summarizing study on the proposed techniques which have used in traffic video. It focuses in these areas, namely vehicle detection, tracking, and classification with appearance of shadow and partial occlusion. Also, we present and classify the traffic surveillance systems to three types based on specific methods which used for developing it. These types shows the detailed information about how the traffic surveillance systems used the image processing methods and analysis tools for detect, segment, and track the vehicles. In addition, shadow and partial occlusion matters and its available solutions are discussed. More specifically, this review gives better understanding and highlights the issues and its solutions for traffic surveillance systems.

Appendix

- vehicle_detection_main.py

```
tf.disable_v2_behavior()
import zipfile
import cv2
import numpy as np
import csv
import pymongo
import uuid
import time
import base64
from packaging import version
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image
# Object detection imports
from utils import label_map_util
from utils import visualization_utils as vis_util
from database import mongo

SPEED_LIMIT = 60

client = mongo.db_connect();
# initialize .csv
with open('traffic_measurement.csv', 'w') as f:
    writer = csv.writer(f)
    csv_line = \
        'Vehicle Type/Size, Vehicle Color, Vehicle Movement Direction, Vehicle
Speed (km/h)'
    writer.writerows([csv_line.split(',')])

# input video
source_video = 'input_video.mp4'
cap = cv2.VideoCapture(source_video)

# Variables
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
fps = int(cap.get(cv2.CAP_PROP_FPS))
```

```
total_passed_vehicle = 0  # using it to count vehicles

# By default I use an "SSD with Mobilenet" model here. See the detection model
zoo
(https://github.com/tensorflow/models/blob/master/research/object_detection/g3d
oc/detection_model_zoo.md) for a list of other models that can be run out-of-the-
box with varying speeds and accuracies.
# What model to download.
MODEL_NAME = 'ssd_mobilenet_v1_coco_2018_01_28'
MODEL_FILE = MODEL_NAME + '.tar.gz'
DOWNLOAD_BASE = \
    'http://download.tensorflow.org/models/object_detection/'

# Path to frozen detection graph. This is the actual model that is used for the
object detection.
PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

NUM_CLASSES = 90

# Download Model
# uncomment if you have not download the model yet
# Load a (frozen) Tensorflow model into memory.
detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.compat.v1.GraphDef()
    with tf.compat.v2.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        # od_graph_def = tf.compat.v1.GraphDef() # use this line to run it with
TensorFlow version 2.x
        # with tf.compat.v2.io.gfile.GFile(PATH_TO_CKPT, 'rb') as fid: # use this
line to run it with TensorFlow version 2.x
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

# Loading label map
# Label maps map indices to category names, so that when our convolution
network predicts 5, we know that this corresponds to airplane. Here I use internal
utility functions, but anything that returns a dictionary mapping integers to
appropriate string labels would be fine
```

```
label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map,
                                                             max_num_classes=NUM_CLASSES,
use_display_name=True)
category_index = label_map_util.create_category_index(categories)


# Helper code
def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape((im_height, im_width,
                                              3)).astype(np.uint8)


# Detection
def object_detection_function(command):
    total_passed_vehicle = 0
    speed = 'waiting...'
    direction = 'waiting...'
    size = 'waiting...'
    color = 'waiting...'

    if (command == "imwrite"):
        fourcc = cv2.VideoWriter_fourcc(*'XVID')
        output_movie = cv2.VideoWriter(source_video.split(".")[0] + '_output.avi',
fourcc, fps, (width, height))

    with detection_graph.as_default():
        with tf.compat.v1.Session(graph=detection_graph) as sess:
            # with tf.compat.v1.Session(graph=detection_graph) as sess: # use this
line to run it with TensorFlow version 2.x

            # Definite input and output Tensors for detection_graph
            image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

            # Each box represents a part of the image where a particular object was
detected.
            detection_boxes =
detection_graph.get_tensor_by_name('detection_boxes:0')

            # Each score represent how level of confidence for each of the objects.
            # Score is shown on the result image, together with the class label.
            detection_scores =
detection_graph.get_tensor_by_name('detection_scores:0')
            detection_classes =
detection_graph.get_tensor_by_name('detection_classes:0')
            num_detections =
detection_graph.get_tensor_by_name('num_detections:0')

            # for all the frames that are extracted from input video
            while cap.isOpened():
                (ret, frame) = cap.read()

                if not ret:
                    print('end of the video file...')
                    break

                input_frame = frame

                # Expand dimensions since the model expects images to have shape:
[1, None, None, 3]
                image_np_expanded = np.expand_dims(input_frame, axis=0)

                # Actual detection.
                (boxes, scores, classes, num) = \
                    sess.run([detection_boxes, detection_scores,
                             detection_classes, num_detections],
                            feed_dict={image_tensor: image_np_expanded})

                # Visualization of the results of a detection.
                (counter, csv_line) = \
                    vis_util.visualize_boxes_and_labels_on_image_array(
                        cap.get(1),
                        input_frame,
                        np.squeeze(boxes),
                        np.squeeze(classes).astype(np.int32),
                        np.squeeze(scores),
                        category_index,
                        use_normalized_coordinates=True,
                        line_thickness=4,
                    )

                total_passed_vehicle = total_passed_vehicle + counter

                # insert information text to video frame
                font = cv2.FONT_HERSHEY_SIMPLEX
```

```
        cv2.putText(
            input_frame,
            'Detected Vehicles: ' + str(total_passed_vehicle),
            (10, 35),
            font,
            0.8,
            (0, 0xFF, 0xFF),
            2,
            cv2.FONT_HERSHEY_SIMPLEX,
        )

        # when the vehicle passed over line and counted, make the color of
ROI line green
        if counter == 1:
            cv2.line(input_frame, (0, 300), (640, 300), (0, 0xFF, 0), 5)
        else:
            cv2.line(input_frame, (0, 300), (640, 300), (0, 0, 0xFF), 5)

        # insert information text to video frame
        cv2.rectangle(input_frame, (10, 275), (230, 337), (180, 132, 109), -1)
        cv2.putText(
            input_frame,
            'ROI Line',
            (545, 190),
            font,
            0.6,
            (0, 0, 0xFF),
            2,
            cv2.LINE_AA,
        )
        cv2.putText(
            input_frame,
            'LAST PASSED VEHICLE INFO',
            (11, 290),
            font,
            0.5,
            (0xFF, 0xFF, 0xFF),
            1,
            cv2.FONT_HERSHEY_SIMPLEX,
        )
        cv2.putText(
            input_frame,
            '-Movement Direction: ' + direction,
            (14, 302),
            font,
            0.4,
            (0xFF, 0xFF, 0xFF),
            1,
            cv2.FONT_HERSHEY_COMPLEX_SMALL,
        )
        cv2.putText(
            input_frame,
            '-Speed(km/h): ' + str(speed).split(".")[0],
            (14, 312),
            font,
            0.4,
            (0xFF, 0xFF, 0xFF),
            1,
            cv2.FONT_HERSHEY_COMPLEX_SMALL,
        )
        cv2.putText(
            input_frame,
            '-Color: ' + color,
            (14, 322),
            font,
            0.4,
            (0xFF, 0xFF, 0xFF),
            1,
            cv2.FONT_HERSHEY_COMPLEX_SMALL,
        )
        cv2.putText(
            input_frame,
            '-Vehicle Size/Type: ' + size,
            (14, 332),
            font,
            0.4,
            (0xFF, 0xFF, 0xFF),
            1,
            cv2.FONT_HERSHEY_COMPLEX_SMALL,
        )

        if (command == "imshow"):
            cv2.imshow('vehicle detection', input_frame)

            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
```

```
            elif (command == "imwrite"):
                output_movie.write(input_frame)
                print("writing frame...")

            if csv_line != 'not_available':
                try:
                    speed_data = round(float(speed), 2)
                    violate_speed = True if float(speed) > SPEED_LIMIT else False
                    image = base64.b64encode(input_frame) if float(speed) >
SPEED_LIMIT else None
                except:
                    speed_data = None
                    violate_speed = None
                    image = None

                if (size != 'waiting...'):
                    vehicle_data = {
                        "resource_id": uuid.uuid4().hex,
                        "serial": total_passed_vehicle,
                        "created_time": time.strftime("%b %d %Y %H:%M:%S",
time.gmtime()),
                        "type": size,
                        "color": color,
                        "direction": direction,
                        "speed": speed_data,
                        "violate_speed": violate_speed,
                        "image": image
                    }

                    mongo.insert_one(client, vehicle_data)
                with open('traffic_measurement.csv', 'a') as f:
                    writer = csv.writer(f)
                    (size, color, direction, speed) = \
                        csv_line.split(',')
                    writer.writerows([csv_line.split(',')])
        cap.release()
        cv2.destroyAllWindows()


import argparse

# Parse command line arguments
parser = argparse.ArgumentParser(description='Vehicle Detection TensorFlow.')
```

```
parser.add_argument("command",
                    metavar="<command>",
                    help="'imshow' or 'imwrite'")
args = parser.parse_args()
object_detection_function(args.command)
```

Data Base Connect

- mongo.py

```
import pymongo

db = 'vehicle_detection'
col = 'vehicle_data'

def db_connect():
    connection_url = 'mongodb://localhost:27017'

    client = pymongo.MongoClient(connection_url)

    return client

def insert_one(client, data):
    vehicle_db = client[db]
    collection = vehicle_db[col]

    collection.insert_one(data)
    print('data inserted')
```

REFERENCES

1 Raad Ahmed Hadi1,Ghazali Sulong and Loay Edwar George, Vehicle detection and tracking techniques :A concise review, in Signal & Image Processing (An International Journal (SIPIJ) Vol.5, No.1) February 2014.

2 Z. Wei, et al., "Multilevel Framework to Detect and Handle Vehicle Occlusion," (Intelligent Transportation Systems, IEEE Transactions on, vol. 9, pp. 161-174) 2008.

3 Nishu Singla,Motion Detection Based on Frame Difference Method, International Journal of Information & Computation Technology. (ISSN 0974-2239 Volume 4) Number 15, 2014

4 B. Suresh, K. Triveni Y. V. Lakshmi, P. Saritha, K. Sriharsha, D. Srinivas Reddy, Determination of Moving Vehicle Speed using Image Processing, (International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Published by, www.ijert.org NCACSPV Conference Proceedings)– 2016 .

5 Genyuan Cheng, Yubin Guo, Xiaochun Cheng, Dongliang Wang, Jiandong Zhao,Real-Time Detection of vehicle speed based on video image12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 2020

6 Jin-xiang Wang, Research of vehicle speed detection algorithm in video surveillance, IIP Lab. Department of Computer Science and Technology, Yanbian University, Yanji, Jilin, China – 2017.

7 Pranith Kumar Thadagoppula, Vikas Upadhyaya,Speed Detection using Image Processing, (International Conference on Computer, Control, Informatics) 2016.