Classification and Identification of vehicle by video Surveillance using Machine Learning

By

Saiyam Shah
17BIT104
Karan Sheth
17BIT105

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
Ahmedabad 382481

Classification and Identification of vehicle by video Surveillance using Machine Learning

Mini Project – III

Submitted in partial fulfillment of the requirements

For the degree of

Bachelor of Technology in Information Technology

By

Saiyam Shah
17BIT104
Karan Sheth
17BIT105

Guided By-
Prof. Gaurang Raval
[DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING]

NIRMA UNIVERSITY
INSTITUTE OF TECHNOLOGY
NAAC ACCREDITED 'A' GRADE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Ahmedabad 382481

CERTIFICATE

This is to certify that the Mini Project -III entitled “**Classification and Identification of vehicle by video surveillance using Machine Learning**” submitted by SAIYAM SHAH (17BIT104), KARAN SHETH (17BIT105) towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Information Technology of Nirma University is the record of work carried out by them under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination.

Dr. Gaurang Raval
Associate Professor
Department of Computer Science & Engg.,
Institute of Technology,
Nirma University,
Ahmedabad

Dr. Madhuri Bhavsar (HOD)
Dept. of Computer Science & Engineering,
Institute of Technology,
Nirma University,
Ahmedabad

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us with the possibility to complete this report. We acknowledge with thanks, the support and motivation rendered by our guide Dr. Gaurang Raval, under whose aegis we were able to complete the task in a given period of time. We also appreciate the constructive suggestions given by our friends to further enhance the content of the report.

At the home front, we are extremely grateful to our family members for the support and encouragement we got from them in successfully completing the report.

ABSTRACT

Vehicle identification using coordination of CCTV cameras are used for Identification of vehicle, measure the speed of a vehicle and to find exactly which route a vehicle used. These goals provide both theoretical and practical ways. So for the theoretical approach, we used a SUMO simulator to estimate the vehicle location inside Nirma University. The location of the vehicle is estimated using vehicle density and OSM is used for Identification of vehicle, to find exactly which route the vehicle uses, and measuring the speed of the vehicle. Nirma University road network has been constructed using OSM. SUMO simulator helps in the simulation process. SUMO simulator imports the .osm file which includes the road network in which the vehicles are running. Different dense scenarios are taken for the simulation. Simulation gives the vehicle parameters like vehicle id, speed, position, location etc. information. So using these vehicle parameters one can identify the vehicle. For the practical approach Mask R-CNN and KNN algorithm is used to classify and identify the vehicles in terms of their color, the direction of the vehicle, speed, for this we have used COCO API.

CONTENTS

Chapter 1
Introduction

In this Modern world due to the increase in the number of vehicles on the road day by day is a very challenging task for the administration to control and manage the traffic, Also to identify the suspect who is not following the rules and regulation. So for that in urban areas CCTV cameras are being installed for monitoring different events and objects taking place. This Video-surveillance also helps to reach out to the person who does not follow traffic rules and regulations. So for the model The same concept is applied to Nirma University campus, Using different cameras throughout the campus at different locations one can observe each activity of vehicles taking place. So an automated model is required for identification and recognition of the vehicles passing through that region in terms of their distinctive functionality like license plate recognition, their color, by their area, by their speed etc. Vehicle matching and identification is the important concept for recognition of the vehicle.

So Vehicle identification using coordination of CCTV cameras are used to identify vehicles, measure the speed of vehicles and to find exactly which route the vehicle used. CCTV Camera will take a picture of the number plate either back or front side whichever is clearly visible or according to the angle of CCTV camera. But if the vehicle is not being identify through its number plate from back or front then its
identification is difficult. That's the reason we can use coordination of CCTV cameras.
Also every vehicle is not systemed with GPS Technology without it identification of vehicle is not possible. So for that the concept of coordination of CCTV cameras comes into picture. Here each CCTV camera will cover one specific region where it is being installed. This all will work like a sensor, collect the data and transmit to all the other devices connected in a network (In our case to pass the data to all the other CCTV camera's of our campus)

So for that we have a virtual map (theoretical approach) of Nirma University created in OSM just to experience how to deal with real life situations. In that, one can measure the optimal distance between each point to every point which we want to know. OSM will provide a .osm file in which the road network is present. so for simulation of this roa network we have used a SUMO simulator through which user can get the experience of road networks, with the help of SUMO road network is open. Vehicles are running on the road network along with that Different - different dense scenarios taken for the simulation. Simulation gives the vehicles parameters like vehicle id, speed, position, location, which route they are taking to reach some position etc. information. So By using these not only one can identify or classify the vehicle but also from different Video - surveillance one can identify which route the vehicle has taken and where it went.

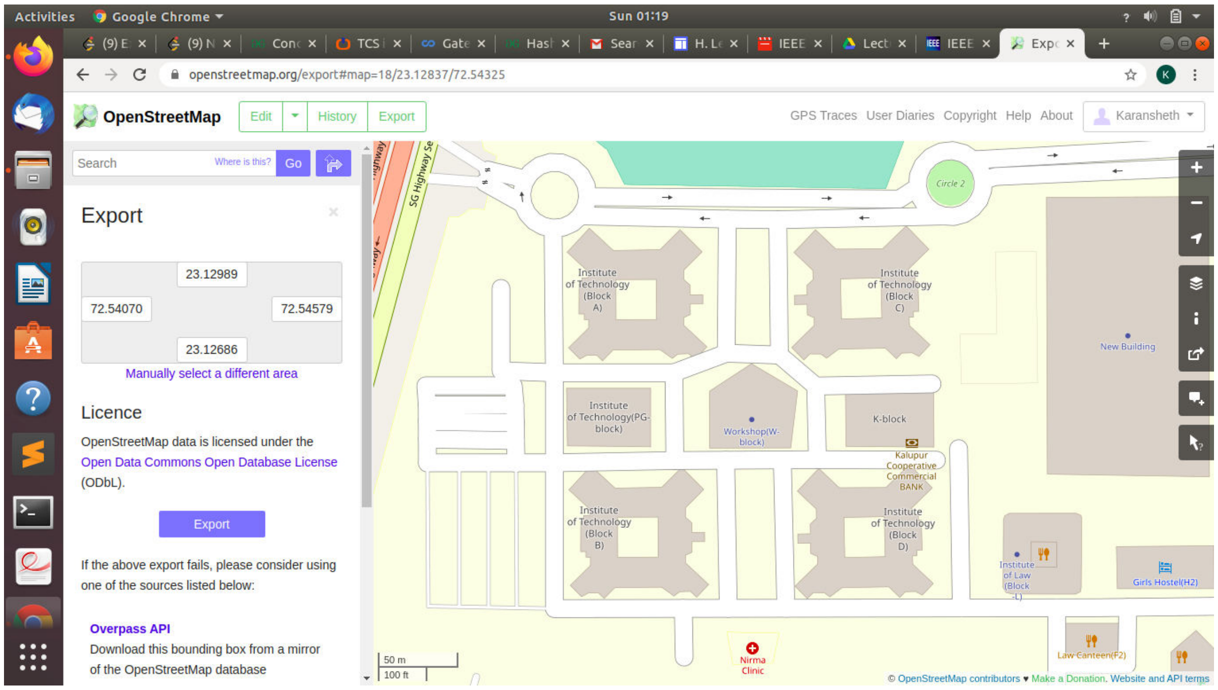
Chapter-2 Literature Survey

From the CCTV cameras installed at the traffic lights, it can be identified which car or truck passed through a signal at any particular time. This is called vehicle identification. Ashwini et al. [1] done vehicle detection using Maximally Stable External Regions (MESR) based on feature vectors collected using feature extraction, feature matching, feature detection, and identification of vehicles. Yang et al. [2] proposed vehicle color identification for real time video surveillance using Support Vector Machine (SVM) to extract wheels, main body, windows, doors and other body parts from the video. Herle et al. [3] removed the background using Gaussian Mixture Model and then vehicles are assigned a unique ID by finding contours in the image frame. Iwaski et al. [4] identified the position of vehicles and also movement of vehicles by capturing thermal images through infrared thermography. Ross Girshick et al. [5] used a technique for vehicle detection and classification called Mask R-CNN. Mask R-CNN used the COCO dataset used for large-scale object detection, segmentation (semantic segmentation and instance segmentation), and captioning dataset.

Chapter-3 OPENSTREETMAP & SUMO SIMULATOR

3.1 OpenStreetMap

OpenStreetMap [6] is an open source platform made to make free editable map available for all from across the world. We can make changes to the map and label any building, roads or any other infrastructure in the way we wanted to use. From this, we were able to export the map of Nirma University with all it’s path in .osm file format which we imported on SUMO (Simulation of Urban Mobility) simulator.


Activities
Google Chrome
Sun 01:19
openstreetmap.org/export#map=18/23.12837/72.54325
OpenStreetMap
Edit
History
Export
GPS Traces
User Diaries
Copyright
Help
About
Karansheth
Search
Where is this?
Go
SG Highway Se
Circle 2
Institute of Technology (Block A)
Institute of Technology (Block C)
New Building
Institute of Technology(PG-block)
Workshop(W-block)
K-block
Kalupur Cooperative Commercial BANK
Institute of Technology (Block B)
Institute of Technology (Block D)
Institute of Law (Block L)
Girls Hostel(H2)
Nirma Clinic
Law Canteen(F2)
50 m
100 ft
© OpenStreetMap contributors ♥ Make a Donation. Website and API terms

Export

23.12989

72.54070

72.54579

23.12686

Manually select a different area

Licence

OpenStreetMap data is licensed under the Open Data Commons Open Database License (ODbL).

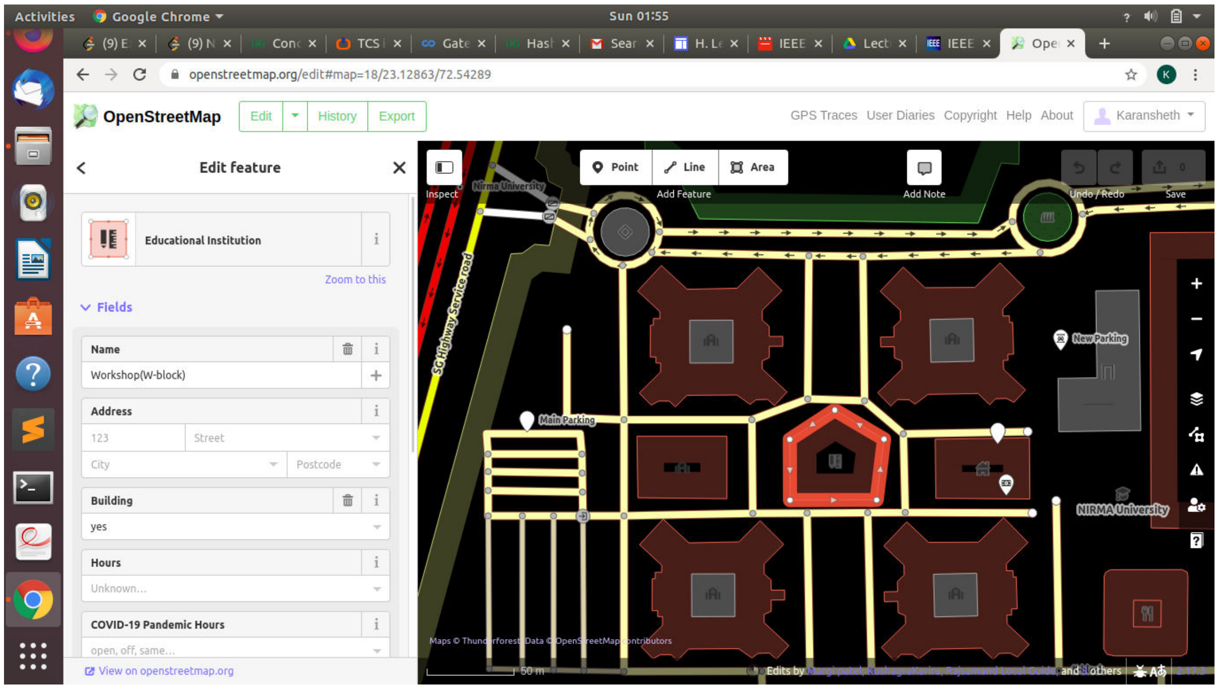
Export

If the above export fails, please consider using one of the sources listed below:

Overpass API

Download this bounding box from a mirror of the OpenStreetMap database

Figure 3.1.1 Nirma University Map in OSM



Edit feature

Educational Institution

Zoom to this

Fields

Name: Workshop(W-block)

Address: 123 | Street | City | Postcode

Building: yes

Hours: Unknown...

COVID-19 Pandemic Hours: open, off, same...

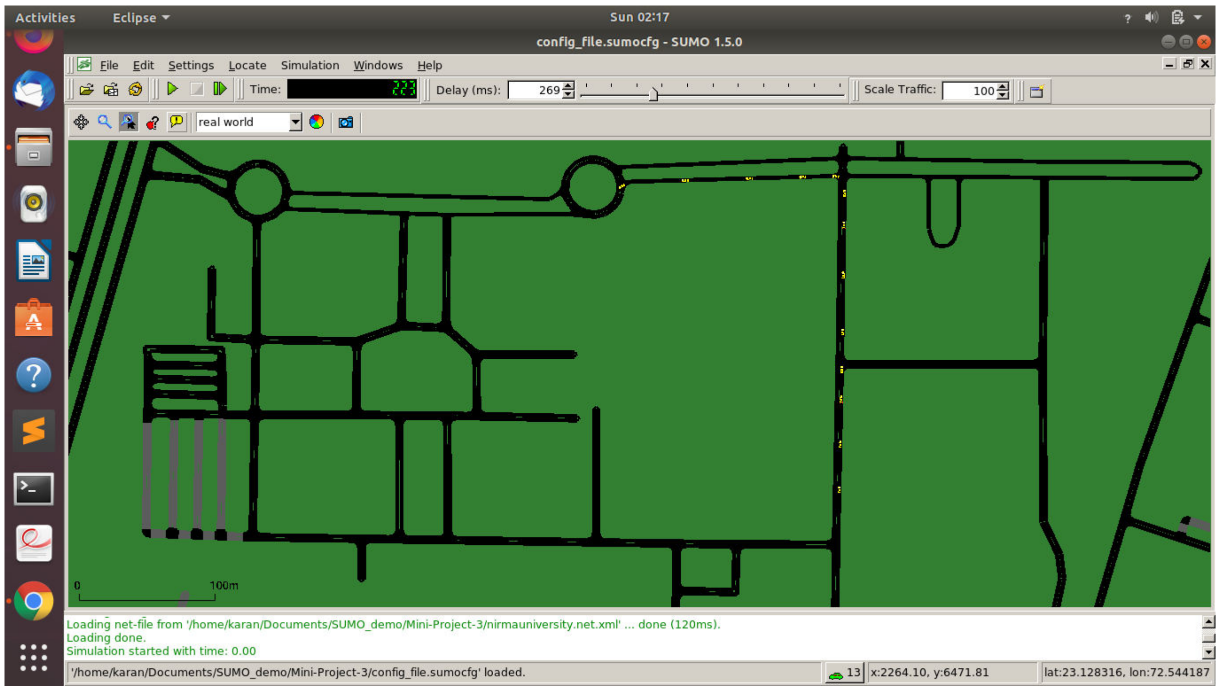
View on openstreetmap.org

Figure 3.1.2 Labeling roads & building in OSM

3.2 SUMO Simulator

SUMO simulator is an open source road network simulator which we used for theoretical examination of the time taken by car to travel from one point to another point on our nirma university map (.osm file) during peak traffic hours and when there is less traffic, hence user can control the number of vehicles according to his/her requirement. This study was done to have an idea about the location of cars after some time during the case of emergency (ambulance - to save life) or during crime (to catch culprit) by understanding the traffic situation in a particular area [7].

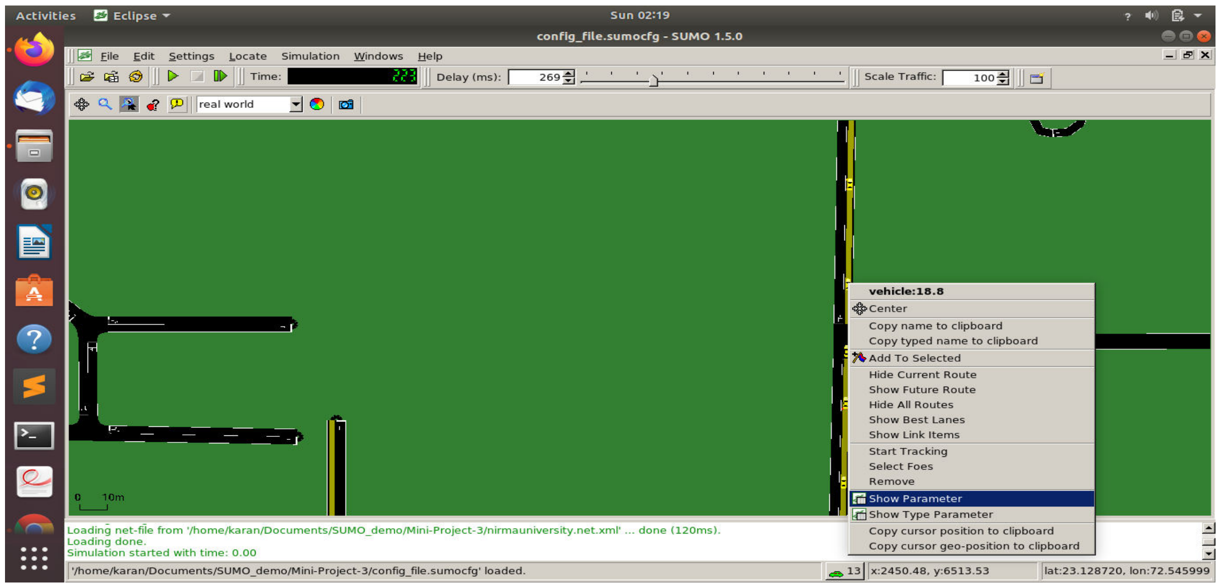
Firstly, we convert the .osm file into an xml file using the NETCONVERT command to make it able to run into SUMO. Then, we used the POLYCONVERT command to import shapes and then we generated random traffic using randomTrips.py file. At last, a SUMO config is create to start the simulation.


Activities
Eclipse
Sun 02:17
config_file.sumocfg - SUMO 1.5.0
File Edit Settings Locate Simulation Windows Help
Time:
Delay (ms): 269
Scale Traffic: 100
real world
0
100m

```
Loading net-file from '/home/karan/Documents/SUMO_demo/Mini-Project-3/nirmauniversity.net.xml' ... done (120ms).
Loading done.
Simulation started with time: 0.00
'/home/karan/Documents/SUMO_demo/Mini-Project-3/config_file.sumocfg' loaded.   13   x:2264.10, y:6471.81   lat:23.128316, lon:72.544187
```

Figure 3.2.1 Random traffic generation in SUMO simulator

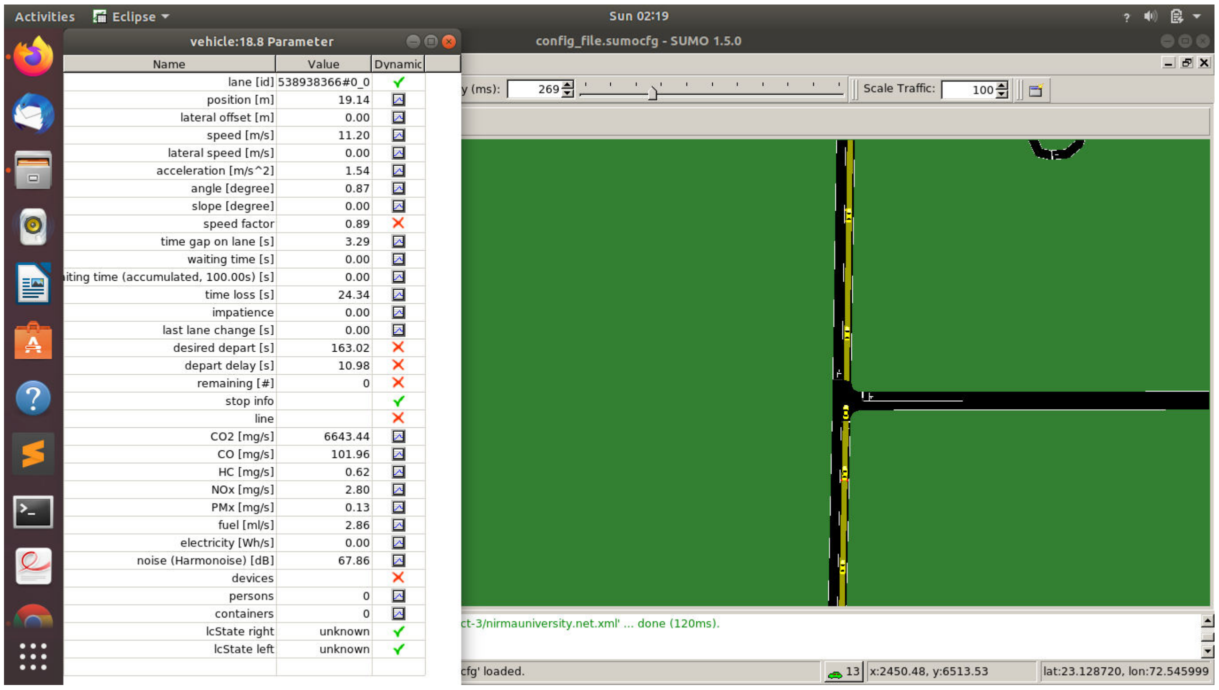
Here, in the above image, a random number of traffic is generated in form of yellow vehicles and is moving from its source to destination which is already set.



```
Loading net-file from '/home/karan/Documents/SUMO_demo/Mini-Project-3/nirmauniversity.net.xml' ... done (120ms).
Loading done.
Simulation started with time: 0.00
'/home/karan/Documents/SUMO_demo/Mini-Project-3/config_file.sumocfg' loaded.   13   x:2450.48, y:6513.53   lat:23.128720, lon:72.545999
```

Figure 3.2.2 SUMO simulator's Functions

In the above image, one car is selected and we can see the best path to destination, and also we can see all possible paths available for a car to reach its destination and many other specifications. In the below image, various parameters like speed, position, etc. for the selected car which can be interpreted using SUMO simulator are shown.



Name	Value	Dynamic
lane [id]	538938366#0_0	✓
position [m]	19.14	📈
lateral offset [m]	0.00	📈
speed [m/s]	11.20	📈
lateral speed [m/s]	0.00	📈
acceleration [m/s^2]	1.54	📈
angle [degree]	0.87	📈
slope [degree]	0.00	📈
speed factor	0.89	✗
time gap on lane [s]	3.29	📈
waiting time [s]	0.00	📈
iting time (accumulated, 100.00s) [s]	0.00	📈
time loss [s]	24.34	📈
impatience	0.00	📈
last lane change [s]	0.00	📈
desired depart [s]	163.02	✗
depart delay [s]	10.98	✗
remaining [#]	0	✗
stop info		✓
line		✗
CO2 [mg/s]	6643.44	📈
CO [mg/s]	101.96	📈
HC [mg/s]	0.62	📈
NOx [mg/s]	2.80	📈
PMx [mg/s]	0.13	📈
fuel [ml/s]	2.86	📈
electricity [Wh/s]	0.00	📈
noise (Harmonoise) [dB]	67.86	📈
devices		✗
persons	0	📈
containers	0	📈
lcState right	unknown	✓
lcState left	unknown	✓

y (ms): 269

Scale Traffic: 100

```
ct-3/nirmauniversity.net.xml' ... done (120ms).
```

```
cfg' loaded.    13    x:2450.48, y:6513.53    lat:23.128720, lon:72.545999
```

Figure 3.2.3: Various parameters calculated for a selected vehicle

Chapter 4
COLOR RECOGNITION & IDENTIFICATION

4.1 Proposed Algorithm

This project focuses on color classifying by K-Nearest Neighbors Machine Learning Classifier which is trained by R, G, B Color Histogram. It can classify and divide into different colors like White, Black, Red, Green, Blue, Orange, Yellow and Violet. By this method it is possible to classify into more different colors according to the work on training data or by considering other features as color moments or correlogram, as this techniques are far better than color histogram when we use it for content-based image retrieval and thus also helps to increase the accuracy in terms of color combination.

In this project we have classified the colors by using K-NN Machine learning classifier algorithm, This classifier is trained the image by only 3 color Histogram values that are R, G, B Color Histogram values. The general work flow is given at the below.

There are 2 main phenomena to understand basic Object Detection/Recognition Systems of Computer Vision and Machine Learning.

1.) Feature Extraction

It generally describes some relevant information which is present in pattern.How to represent the some interesting points we found to compare them with other interesting points (features) in the image.

2.) Classification

An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. “classifier” generally trained the input data for understanding, which are related to a class. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.

For this project:

1.) Feature Extraction = Color Histogram

Color Histogram is a representation of the distribution of colors in an image. For digital images, it will represent the total number of pixels having fixed range of colors whether they belong to R, G, B segment or any combination of it, that span the image's color space, the set of all possible colors.

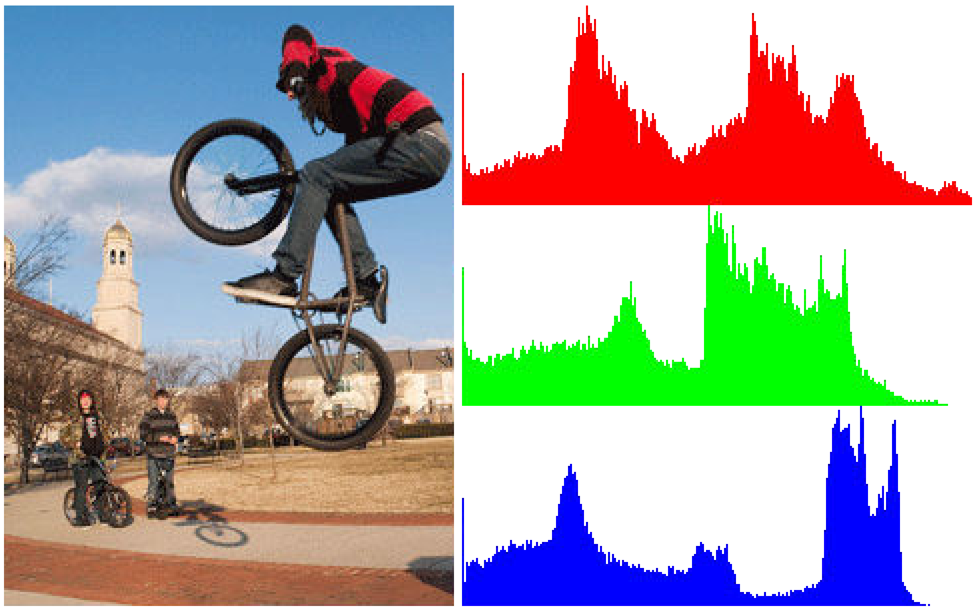


Figure 4.1.1 shows the image with the combination of r,g,b

2.) Classification = K-Nearest Neighbors Algorithm

K nearest neighbors is a simple as well as supervised learning algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). It assumes that all the similar or same things are near to each other or related to each other. For eg: "Birds of a feather flock together."

So KNN we can say that it generally captures the similarity in terms of proximity(distances) with some mathematics. The only dis-advantage of this algorithm is it gets much slower with the increase in the number of independent variables.

Thus KNN has been used in statistical estimation and pattern recognition

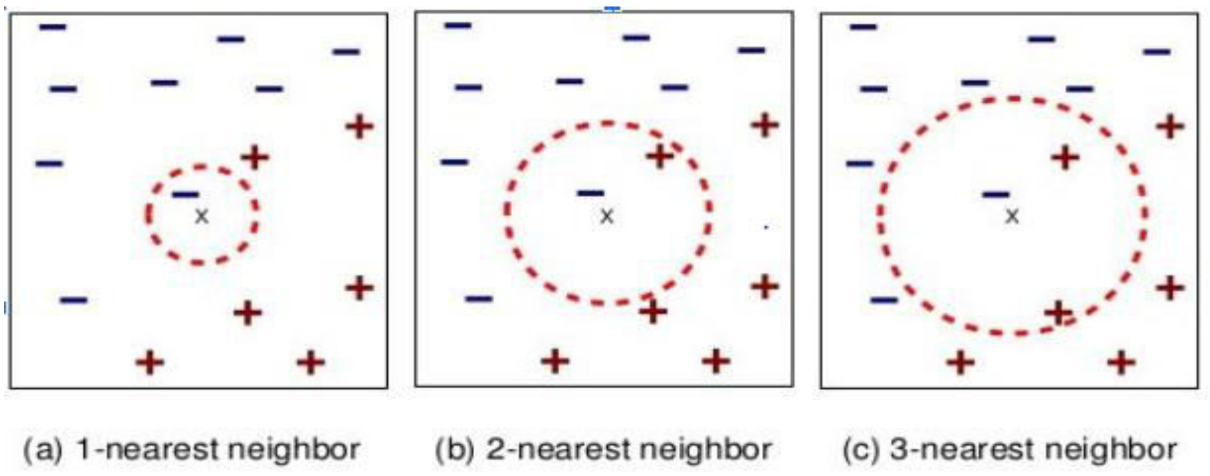


Figure 4.1.2 K - nearest neighbour of a record x are data points that have the k- smallest distance to x.

Implementation
OpenCV was used for color histogram calculations and knn classifiers. NumPy was used for matrix/n-dimensional array calculations. The program was developed on Python in the Linux environment.

Following are the color recognition python files:

- *color_classification_webcam.py*: test class to perform real-time color recognition from webcam stream.
- *color_classification_image.py*: test class to perform color recognition on a single image.
- *feature_extraction.py*: feature extraction operation class
- *knn_classifier.py*: knn classification class

1.) Explanation of "feature_extraction.py" python file
We got the RGB color histogram of images by this Python class. For example, a plot of RGB color histogram for one of the red images is given at the below.

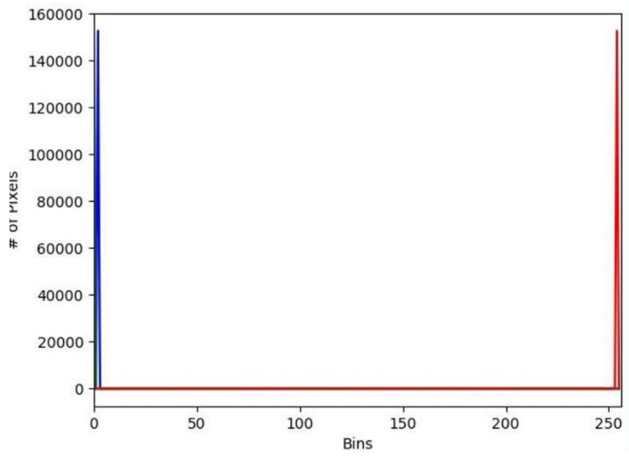


Figure 4.1.3 It shows the domination of R,G,B values with combination of [254,0,2]

We decided to use the bin number of histogram which has the peak value of pixel count for R, G and B as a feature so I can get the dominant R, G and B values to create feature vectors for training. For example, the dominant R, G and B values of the red image which is given at above is [254, 0, 2].

Thus we get the dominant R, G, B values by using Color Histogram for each training image then we labelled them because KNN classifier is a supervised learner and we deploy these feature vectors in the csv file.

2.) Explanation of “knn_classifier.py”
This class provides these main calculations;

1. Fetching training data
2. Fetching test image features
3. Calculating euclidean distance
4. Getting k nearest neighbors
5. Prediction of color
6. Returning the prediction is true or false

“color_classification_webcam.py” is the main class of our program, it provides;

1. Calling “feature_extraction.py” to create training data by feature extraction
2. Calling “knn_classifier.py” for classification

.

4.2 Object Counting Using Mask R-CNN

The method, called Mask R-CNN, extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition .

It is a kind of the separator which generally separates the images as well as videos and outputs it in the form of bounding box,classes and masks.

Mask R-CNN is simple to train and adds only a small overhead to Faster R-CNN, running at 5 fps.

Moreover, Mask R-CNN is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework. We show top results in all three tracks of the COCO suite of challenges, including instance segmentation, bounding-box object detection, and person keypoint detection.

Architecture of Mask RCNN structure

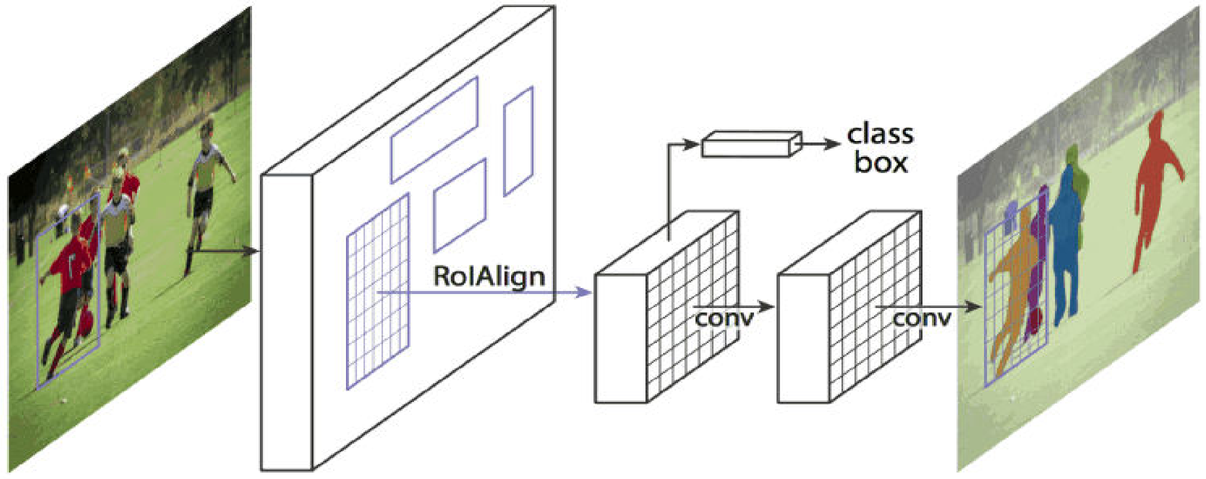


Figure 4.2.1 It shows the architecture of R-CNN

Illustration of Mask RCNN structure

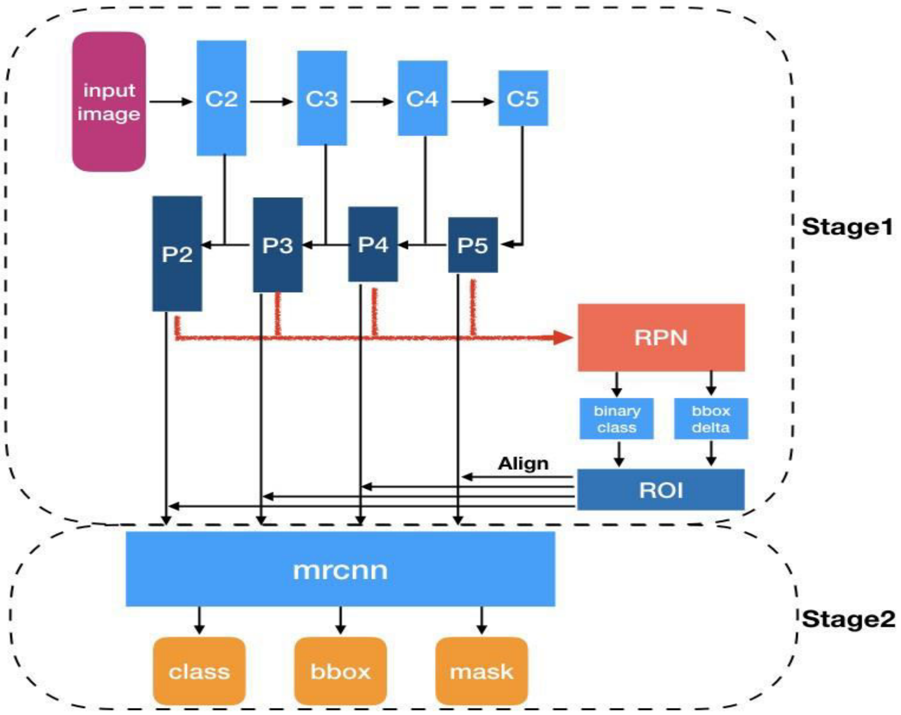


Figure 4.2.2 IIstrutrate about Mask R-CNN structure

Installation

1. Clone this repository
2. Install dependencies

```
pip3 install -r requirements.txt
```

1. Run setup from the repository root directory

```
python3 setup.py install
```

1. Download pre-trained COCO weights (mask_rcnn_coco.h5) from the releases Page.
1. To train or test on MS COCO install pycocotools from one of these repos. They are forks of the original pycocotools with fixes for Python3 and Windows (the official repo doesn't seem to be active anymore)

Converting XML to TFRecord

To convert XML files to TFRecord, first It need to be converted to CSV. below is the code that converts the XML files to CSV files.
Once the XML files have been converted to CSV files, TFRecords can be generated using a python. Using Tensorflow TFRecords is a convenient way to get your data into your machine learning pipeline.

Commands for generating the tfrecord for the training data as well as testing data::

1. **python generate_tfrecord.py — csv_input=data/train_labels.csv — output_path=data/train.record**
2. **python generate_tfrecord.py — csv_input=data/test_labels.csv — output_path=data/test.record**

To get our Object detector we can either use a pre-trained model and then use transfer learning to learn a new object, or we could learn new objects entirely from scratch. The benefit of running the model on a training data set is to get the output much quicker, easy to handle and require less data. For this reason, we're going to be doing transfer learning here. TensorFlow has quite a few pre-trained models with checkpoint files available, along with configuration files.

For this task We have used Faster R-CNN Inception.

Commands to be fire on Terminal:

```
python3 train.py --logtostderr --train_dir=training/--
pipeline_config_path=training/faster_rcnn_inception_v2_coco_2018_01_28.config
```

Testing Object Detector

To test how good is our model, for that we import an inference graph. In the 'models/object_detection' directory, there is a script that does this for us: 'export_inference_graph.py'.

To run this, it just needs to pass in the checkpoint and your pipeline config. Your checkpoint files should be in the 'training' directory. For example:

Following is the command to run;

```
python3 export_inference_graph.py
    --input_type image_tensor
    --pipeline_config_path training/ssd_inception_v2_coco_2017_11_17.config
    --trained_checkpoint_prefix training/model.ckpt-7051
    --output_directory logos_inference_graph
```

4.3 Implementing k-NN for image classification

The kNN also refer as k-Nearest Neighbour classifier is one of the most simple machine learning/image classification algorithms.

Inside this algorithm it calculates the distance between feature vectors or the query,as well as all the examples in the data much like building an image search engine — After this as an output we have the labels which are mapped with each image to help us to predict and return an actual category for the image.

Simply put, the k-NN algorithm classifies unknown data points by finding the most common class among the k-closest examples [11]. Each data point in the k closest examples casts a vote and will show finally the category with the most votes wins!

Chapter 5 Object Detection and Classification

5.1 Tensorflow Object Counting

General Capabilities of The TensorFlow Object Counting API ARE SHOWN BELOW:

1. Detect just the targeted objects
2. Count just the targeted objects
3. Predict color of the targeted objects
4. Predict speed of the targeted objects
5. Predict speed of all the objects
6. Print out the detection-counting result in a .csv file as an analysis report
7. Save and store detected objects as new images under detected_object folder.
8. Save detection and counting results as a new video or show detection and counting results in real time
9. Process images or videos depending on your requirements

In this project object detection and tracking is restricted for one car. Find the location of an object if it appears in an image. So our the detection part as well as classification part of the real object capture in the frame Mask R-CNN is used as a deep learning technique.

Mask R-CNN has three outputs for each candidate object, a class label, bounding box offset and an object mask. Here COCO [13] dataset is used for large-scale object detection, segmentation, and captioning. COCO API available on the Internet. COCO have several features like it has 2,500,000 instances and 91 object

5.2 Classification Results

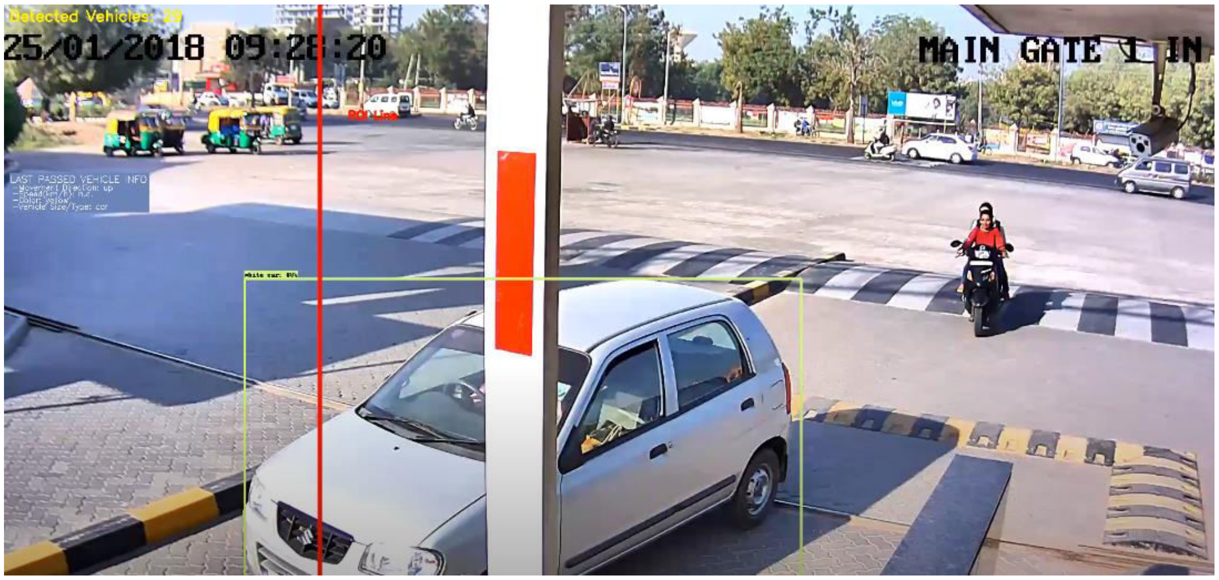


Figure 5.2.1 Car Passing through ROI is being detected

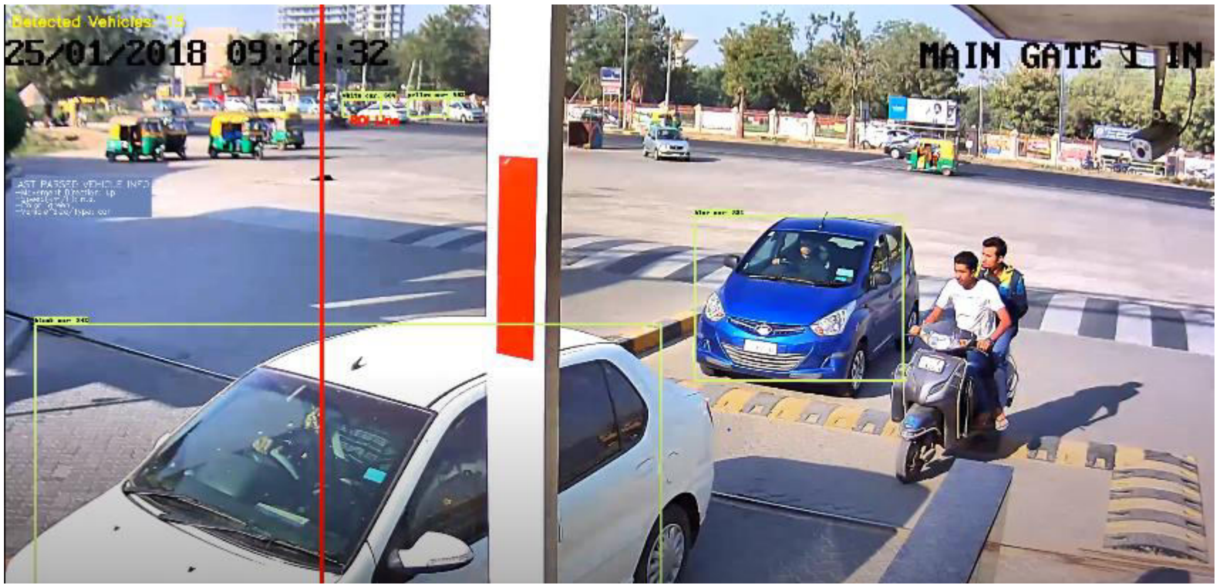


Figure 5.2.2 Detect the multiple Vehicle in one frame

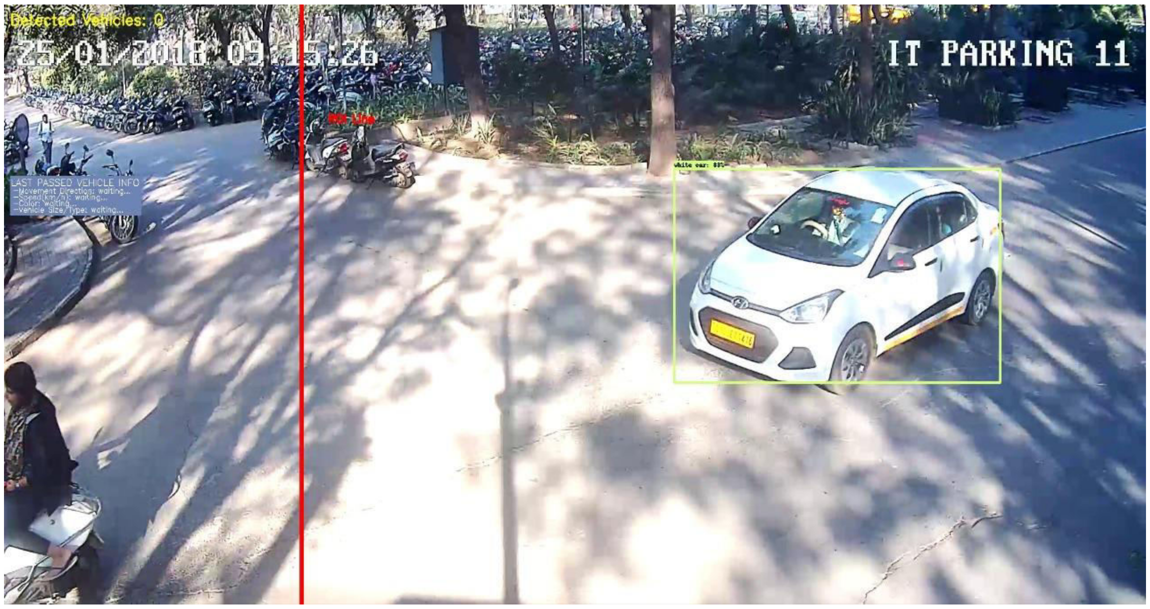


Figure 5.2.3 shows the detail of the speed when passes line

5.3 Speed Detection & its Results

Speed detection is based on contour, motion detection, and background subtraction. Python and OpenCV used to perform this task.With help of contour points ROI was fixed. If the object passes through ROI then its speed has been calculated. According to vehicle density, different scenarios have been taken. In this scenario, it also detects the speed of person and objects if it detects in the video because here a person is considered as an object and it detects the motion of the person and according to that calculate the speed. Speed detection code was taken from the github [12]. Speed was calculated using contour and motion detection.

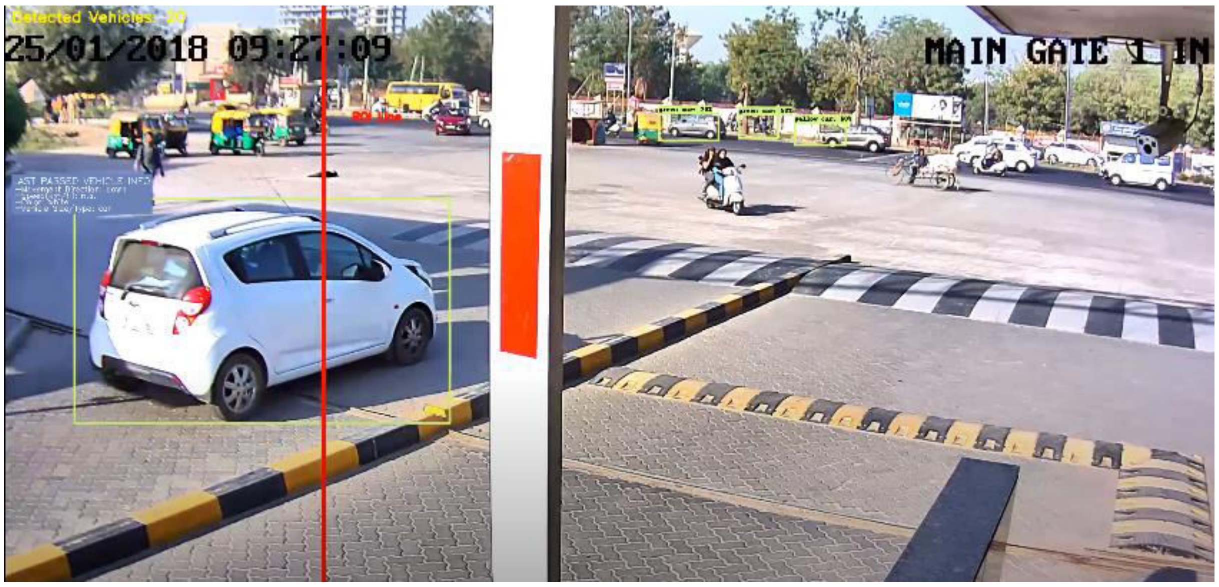
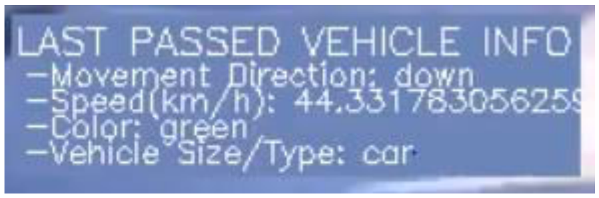


Figure 5.3.1: Speed Detected Vehicle Image as well as number of detected vehicle are increases



LAST PASSED VEHICLE INFO
-Movement Direction: down
-Speed(km/h): 44.33178305625
-Color: green
-Vehicle Size/Type: car

Figure 5.3.2 Detailed About Passes Vehicle Info

Chapter 6 Conclusion and Summary

6.1 Summary

In our project we perform Vehicle Identification as well as Classification of vehicles by video surveillance by using R-CNN Mask as well as KNN machine learning model. We have performed vehicle identification using CCTV cameras used for Identification of vehicle, measure the speed of a vehicle and to find exactly which route a vehicle used. These goals provide both theoretical and practical ways. So for the theoretical approach, we used a SUMO simulator to estimate the vehicle location inside Nirma University. Simulation gives the vehicle parameters like vehicle id, speed, position, location etc. information. So using these vehicle parameters one can identify the vehicle. For the practical approach Mask R-CNN and KNN algorithm is used to classify and identify the vehicles in terms of their color, the direction of the vehicle, speed, for this we have used COCO API.

6.2 Conclusion

The final result of the Simulation shows the identification as well as classification of the vehicle in which one can identify the vehicle with all information like vehicle is at which location at which time. With the help of SUMO simulation with theoretically approach it has been tried out to find the vehicle whereabouts. Simulation is performed for various scenarios as per various geographical location and from the results, a behavior of the vehicle could be known in the context of speed, position, lane ID etc. from the observations gathered. From this knowledge distributed camera networks were employed in our detecting areas for the recognition of vehicles. Practical results also show the identification of vehicles in which the project restricted for one car and 2 cameras are considered, the one at front gate of the Institution and other at the parking area. With the help of Mask R-CNN technique object detection and classification was done. Classification results show the vehicle type like whether it is a car, bus, or motorcycle etc. At different cameras car shows from different angles and different speeds. So one can identify the vehicle after comparing the data from the two resulting cameras . And from Nirma university map, optimal distance database, detection, classification and speed of the vehicle one can also predict that if the vehicle appears at point X at time P then it appears at point Y at time Q in the second camera and it appears at point Z at time R in the third camera.

REFERENCES

1. B. Ashwini, B. Deepashree, B. N. Yuvaraju and P. S. Venugopala, "Identification of vehicles in traffic video," 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES), Paralakhemundi, 2016, pp. 588-593, doi: 10.1109/SCOPES.2016.7955507.
2. Chih-Yang Lin, Cheng-Hao Yeh and Chia-Hung Yeh, "Real-time vehicle color identification for surveillance videos," 2014 International Conference on Electronics, Communications and Computers (CONIELECOMP), Cholula, 2014, pp. 59-64, doi: 10.1109/CONIELECOMP.2014.6808568.
3. L. Herle and P. Sharma, "Vehicle detection and identification in an unconstrained environment," 2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE), Bhopal, 2017, pp. 66-69, doi: 10.1109/RISE.2017.8378126.
4. Y. Iwasaki, M. Misumi, and T. Nakamiya, "Robust vehicle detection under various environmental conditions using infrared thermal images and its application to road traffic flow monitoring," Proceedings in ARSA-Advanced Research in Scientific Areas, no. 1, 2012.
5. K. He, G. Gkioxari, P. Doll´ar, and R. Girshick, "Mask r-cnn," in Computer Vision (ICCV), 2017 IEEE International Conference on, pp. 2980–2988, IEEE, 2017.
6. T. Wan, H. Lu, Q. Lu and N. Luo, "Classification of High-Resolution Remote-Sensing Image Using OpenStreetMap Information," in IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 12, pp. 2305-2309, Dec. 2017, doi: 10.1109/LGRS.2017.2762466.
7. S. Haddouch, H. Hachimi and N. Hmina, "Modeling the flow of road traffic with the SUMO simulator," 2018 4th International Conference on Optimization and Applications (ICOA), Mohammedia, 2018, pp. 1-5, doi: 10.1109/ICOA.2018.8370580.
8. J. George, L. Mary and Riyas K S, "Vehicle detection and classification from acoustic signal using ANN and KNN," 2013 International Conference on Control Communication and Computing (ICCC), Thiruvananthapuram, 2013, pp. 436-439, doi: 10.1109/ICCC.2013.6731694.
9. Okfalisa, I. Gazalba, Mustakim and N. G. I. Reza, "Comparative analysis of k-nearest neighbor and modified k-nearest neighbor algorithm for data classification," 2017 2nd International conferences on Information Technology, Information Systems and Electrical Engineering (ICITISEE), Yogyakarta, 2017, pp. 294-298, doi: 10.1109/ICITISEE.2017.8285514.
10. S. Bouaich, M. A. Mahraz, J. Riffı and H. Tairi, "Vehicle counting system in real-time," 2018 International Conference on Intelligent Systems and Computer Vision (ISCV), Fez, 2018, pp. 1-4, doi: 10.1109/ISACV.2018.8354033.
11. X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan, "Proposal-free network for instance-level object segmentation," arXiv preprint arXiv:1509.02636, 2015.
12. @ONLINE{vdtct,
    author = "Ahmet Özlü",
    title = "Vehicle Detection, Tracking and Counting by TensorFlow",
    year   = "2018",
    url   = "https://github.com/ahmetozlu/vehicle_counting_tensorflow"

}
13. T.-Y. L. tylin, “COCO API - Dataset @ http://cocodataset.org/.” https://github.com/cocodataset/cocoapi, 2018.