

PROFESSIONAL TRAINING REPORT
AT
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology Degree in
Information technology

By
ARTHEESWARI P
(Reg no 38120012)

JAYAROSHINI M
(Reg no 38120032)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI – 600119.

MAY 2022

DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ARTHEESWARI P(Reg. No 38120012) JAYAROSHINI M(Reg. No 38120032)** who carried out the project entitled "**DETECTION OF CAR AND DISTANCE**" under our supervision from MAY 2021 to JUNE 2022.

Internal Guide

Dr.Y.BEVISH JINILA

Head of the Department

Dr. R.SUBHASHINI.,M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **ARTHEESWARI P (Reg. No. 38120012)** and **JAYAROSHINI M(Reg. No 38120032)** hereby declare that the Project Report

entitled “**DETECTION OF CAR AND DISTANCE**” done by me

under the guidance of **Dr.Y.BEVISH JINILA** is

submitted in partial fulfillment of the requirements for the award of

Bachelor of Technology degree in Information Technology.

DATE :

PLACE :

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to Board of Management of SATHYABAMA for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr.T.SASIKALA.,Ph.D.**,Dean, school of computing and **Dr.R.SUBHASHINI.,M.E.,Ph.D.**,Head of the Department of Computing for providing me necessary support and details at the right time during the progressive views.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.V.MARIA ANU.,M.E.,Ph.D.**,for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-Teaching staff members of the Department of **Information Technology** who were helpful in many ways for the completion of my project.

ABSTRACT

Moving car information in traffic video signals contains plenty of important information on road safety. In this paper, we propose an algorithm that extracts and tracks feature points of cars. The information is indispensable for measuring moving car speed, counting the number of cars, monitoring distance between two cars, classifying running directions, and monitoring congestion situation. Utilizing Harris operator for detecting edges and corners, feature points are extracted, followed by block-matching to track the feature points in successive video frames. Many cars can be tracked at the same time automatically since the information is obtained from video sequences. As an example, this paper shows how car speed can be measured. Average speed of the vehicle is 100 km per hour. Image processing methods are a combination of methods of colour region, line selection, canny edge detection, and Hough transform. The result shows this algorithm needed to be add some method that can change the parameters during day and night adaptively. Overall the implementation method in Python Language can successfully detect the road lane with accuracy above 90 percent.

TABLE OF CONTENTS:

CHAPTER NO	TITLE	PAGENO
	ABSTRACT	Vi
	LIST OF FIGURES	viii
	LIST OF ABBREVIATIONS	ix
1.	INTRODUCTION	
	1.1 HISTORY OF PYTHON	1
	1.2 FEATURES OF PYTHON	1
	1.3 ARTIFICIAL INTELLIGENCE	4
	1.4 OUTLINE OF PROJECT	5
2.	AIM AND SCOPE OF THE PRESENT INVESTIGATION	
	2.1 AIM	6
	2.2 SCOPE	6
	2.3 LITERATURE REVIEW	7
3	EXPERIMENTAL OR MATERIALS AND METHODS,ALGORITHMS	
	3.1 MODULES	9
	3.2 METHODOLOGY	9
4.	SYSTEM IMPLEMENTATION	
	4.1 SYSTEM ANALYSIS	11
	4.2 RESULTS	14
5.	CONCLUSION AND FUTURE WORK	
	5.1 CONCLUSION	16
	REFERENCES	17
	APPENDIX	
	A.SOURCE CODE	18

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
4.1	DETECTION THE DISTANCE OF THE VEHICLE	14
4.2	DETECTION THE CAR OR VEHICLE	15

LIST OF SYMBOLS AND ABBREVIATIONS

ABBREVIATION	EXPANSION
HOG	Histogram of Oriented Gradient
PUJ	Public Utility Jeeps
ADAS	Advanced Driver Assistance System
GMM	Gaussian Mixture Model

CHAPTER 1

INTRODUCTION

HISTORY:

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python was conceived in the late 1980s as a successor to the ABC language. Python 2.0, released 2000, introduced features like list comprehensions and a garbage collection system capable of collecting reference cycles. Python 3.0, released 2008, was a major revision of the language that is not completely backward-compatible, and much Python 2 code does not run unmodified on Python 3. Due to concern about the amount of code written for Python 2, support for Python 2.7 (the last release in the 2.x series) was extended to 2020. Language developer Guido van Rossum shouldered sole responsibility for the project until July 2018 but now shares his leadership as a member of a five-person steering council.

Python interpreters are available for many operating systems. A global community of programmers develops and maintains C Python.

FEATURES OF PYTHON1 :

When we say the word 'easy', we mean it in different contexts.

Easy to code:

As we have seen in earlier lessons, Python is very easy to code. Compared to other popular languages like Java and C++, it is easier to code in Python. Anyone can learn python syntax in just a few hours. Though sure,

mastering Python requires learning about all its advanced concepts and packages and modules. That takes time. Thus, it is programmer-friendly.

Easy to read :

Being a high-level language, Python code is quite like English. Looking at it, you can tell what the code is supposed to do. Also, since it is dynamically-typed, it mandates indentation. This aids readability.

Read: Applications of Python Programming Language

Expressive :

First, let's learn about expressiveness. Suppose we have two languages A and B, and all programs that can be made in A can be made in B using local transformations. However, there are some programs that can be made in B, but not in A, using local transformations. Then, B is said to be more expressive than A. Python provides us with a myriad of constructs that help us focus on the solution rather than on the syntax. This is one of the outstanding python features that tells you why you should learn Python.

Free and Open-Source :

Firstly, Python is freely available. You can download it from the following link <https://www.python.org/downloads/>

For insight on how to download and install Python, refer to our tutorial on Python Installation.

Secondly, it is open-source. This means that its source code is available to the public. You can download it, change it, use it, and distribute it. This is called FLOSS(Free/Libre and Open Source Software). As the

Python community, we're all headed toward one goal- an ever-bettering Python.

High- Level :

As we discussed in point 2b, it is a high-level language. This means that as programmers, we don't need to remember the system architecture. Nor do we need to manage the memory. This makes it more programmer-friendly and is 1 of the key python features. Any doubt yet in the features of Python.

Portable :

Let's assume you've written a Python code for your Windows machine.

Now, if you want to run it on a Mac, you don't need to make changes to it for the same. In other words, you can take one code and run it on any machine, there is no need to write different code for different machines. This makes Python a portable language. However, you must avoid any system-dependent features in this case.

Read: Career Opportunities in Python

Interpreted :

If you're any familiar with languages like C++ or Java, you must first compile it, and then run it. But in Python, there is no need to compile it. Internally, its source code is converted into an immediate form called bytecode. So, all you need to do is to run your Python code without worrying about linking to libraries, and a few other things.

By interpreted, we mean the source code is executed line by line, and not all at once. Because of this, it is easier to debug your code. Also, interpreting makes it just slightly slower than Java, but that does not matter compared to the benefits it has to offer.

Any doubt or query yet about the features of python programming language?

If yes, Drop a comment and we will get back to you.

Install Python on Windows

Object-Oriented :

A programming language that can model the real world is said to be object-oriented. It focuses on objects, and combines data and functions. Contrarily, a procedure-oriented language revolves around functions, which are code that can be reused. Python supports both procedure-oriented and object-oriented programming which is one of the key python features. It also supports multiple inheritance, unlike Java. A class is a blueprint for such an object. It is an abstract data type, and holds no values. Read: Syntax in Python

Extensible :

If needed, you can write some of your Python code in other languages like C++. This makes Python an extensible language.

ARTIFICIAL INTELLIGENCE:

It is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs an intelligent software systems. The aim of AI is to improve computer and a functions which are related to human knowledge, for example, reasoning, learning, and problem-solving. The intelligence is intangible. It is composed of

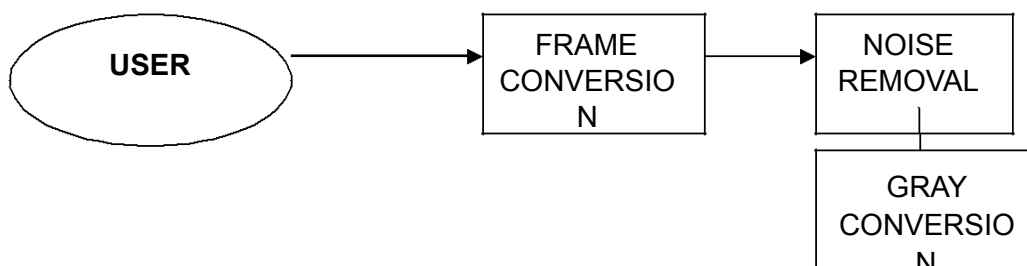
- Reasoning
- Learning
- Problem Solving
- Perception
- Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the artificial intelligence

OUTLINE PROJECT :

In recent years, as the number of cars increases more and more rapidly, road safety is increasingly of a public concern. To address this concern, research on monitoring how cars are running on the roads is mandatory. Possible outcomes of this kind of research will help enhance traffic efficiency by early warning congestion ahead, letting cars slow down, solving various problems such as environmental pollution, etc. One of the outcomes may be to detect the speed of running cars from the video signals. In reality, there is a method of detecting the speed with wireless technology, which is already used in practice. The detection accuracy is very high, but it is necessary to construct a dedicated gate over the road where a wireless sensor is mounted. In addition, when the position of the moving car shifts horizontally a little, the detection accuracy will degrade to a great extent because angle requirement is severe. Furthermore, the speed of more than one moving cars can not be detected at the same time by a single sensor.

If image processing is applied, several cars can be handled simultaneously. With image processing, special carefulness is not needed in installing a video camera. Besides the speed measurement, various information of traffic situation can be presented. In this paper, we propose a method in which extraction of the moving cars along with formation and tracking of the car area is carried out using only the video signals. Feature Points in the car area are extracted by using the Harris operator. Stable detection of Feature Points is realized by application of block matching algorithm, producing good tracking results. Speed of the cars can be detected with considerably high accuracy using



CHAPTER 2

AIM AND SCOPE OF THE PRESENT INVESTIGATION

AIM :

In this work, we implement a real-time vehicle classification and speed estimation system and apply it to videos acquired from traffic cameras installed in highways. In this approach we:

Detect moving vehicles through background- foreground segmentation techniques.

Compare different supervised classifiers (e.g. artificial neural networks) for vehicle classification into categories: (car, motorcycle, van, and bus/truck).

Apply a calibration method to geo reference vehicles using satellite images.

Estimate vehicles speed per class using feature tracking and nearest Neighbours algorithms.

SCOPE:

In the process of selecting the most suitable classification method for this scheme, several supervised algorithms were tested and compared in Python using scikit-learn, a powerful open-source set of machine learning libraries that also includes data mining and data analysis tools. 10-fold cross-validation was used to obtain more accurate performance results, measuring in each experiment:

Training time.

Classification time.

Accuracy on train samples.

Accuracy on test samples.

LITERATURE REVIEW :

Feasibility Study :

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

Economical Feasibility :

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility :

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high

Social Feasibility :

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the

methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 3

EXPERIMENTAL OR MATERIALS AND METHODS,ALGORITHMS

MODULES:

Method 1:

Based on length of trajectory The Feature Point in the car area is used for detecting speed after the car has passed the window. The number of pixels between the start and the end point of the trajectory is counted from the coordinates of the trajectory. After multiplying the distance per pixel by the number of pixels in the trajectory, speed can be calculated, since both the distance and time are already known. The speed measurement results for the two cars are as follows. For the 1st car, there are 83 frames consumed, with Start frame number 64 and End frame number 147. The distance is 32m. Therefore, speed is calculated to be 41 km/h. For the 2nd car, there are 72 frames consumed, with Start frame number 129 and End frame number 201. The distance is 25m. Therefore, speed is calculated to be 38 km/h.

Method 2:

Based on reliable part of trajectory If the trajectory is not a straight line from the start to the end point, the result should be more or less in error. So, we only used the straight part of each trajectory to measure the speed of the two running cars, since the straightness is equal to high reliability. The calculated result of the experiment is as follows. For the 1st car, the entire trajectory is determined as straight since fitting a straight line onto the trajectory produced an error within one pixel difference. Therefore, the result is the same as Method 1.

Error analysis :

Speed of the moving cars is measured for the same dynamic scene by two methods. The results obtained in the measurements are summarized in TABLE I. For the 1st car, both methods show almost the same result, which is very easy.

METHODOLOGY:

The study was divided into four phases namely: Data gathering, classifier training, detection and tracking method, and speed estimation.

Data Gathering :

To be able to collect images and videos for training and testing, a dash cam was used for data collection. The video produced has a resolution of 1280x720 pixels and a frame rate of 30 frames per second. Vehicle images that were collected from the video will be used as training images for the classifiers. The vehicle images were obtained on the roads of the Philippines and were classified into the following: Public Utility Jeepneys (PUJ), Large Vehicles (Trucks and Buses), Motorcycles, and Ordinary Vehicles (Sedans, SUVs, etc.).

Classifier Training :

The collected images from the videos were resized into a 64x64 pixel size. The Histogram of Oriented Gradients (HOG) features of each image were extracted and will be used for classification. The HOG is a computer vision technique that counts the occurrence of gradient orientation in localized portions of an image. It is created by computing the orientation and gradient magnitude at each image sample point in a region around an anchor point. The region is split into NxN sub regions.

CHAPTER 4

SYSTEM IMPLEMENTATION

SYSTEM ANALYSIS:

- Implementation of vehicle detection algorithm on toll road Cipularang as parts of self-driving car system.
- Video image taken using action camera mounted on top of the vehicle, with 1280x720 resolutions.
- Average speed of the vehicle is 100 km per hour. Programming language of image processing using Python 3.
- Image processing method is a combination of methods of object detection, feature intuition, color spaces, and HOG (histogram of oriented gradient). The result shows this algorithm needed to be add some method that can change the parameters during day and night adaptively.
- Because constant parameters can only be used in the same lighting conditions. Overall the implementation method in Python Language can be successfully detecting the vehicle with accuracy above 90 percent.

Disadvantages:

1. If it were to be implemented in a real scenario, vehicle manufacturers would have to be required to implement a similar device which can transmit similar packets.
2. Accuracy is too low.
3. Detecting vehicles in images acquired from a moving platform is a challenging problem.

PROPOSED METHOD:

Vehicle speed from video data consists of processes that are discussed further later in this paper. Each process will do particular work which the result will be used by next process until estimated speed is calculated.

A. Video Input :

Video data is captured from static camera at top position of highway, using geometric equation to calculate perpendicular view (P) and distance between object and camera (D) based on camera height (H), field of view angle of the camera (T_c), and angle of the camera (T_v). Perpendicular view is used to calibrate estimated speed to standard unit, km/h.

B. Pre-processing :

Video data is extracted to frames and each frame is applied preprocessing before frame is being processed by background subtraction. Pre-processing is used to minimize bold shadow that can be detected as solid object which can decrease object detection accuracy.

C. Background Subtraction:

Background subtraction is used to extract foreground image based on background model for each frame. There are many background subtraction methods which have their own advantages and disadvantages. This paper uses Gaussian Mixture Model known as GMM method to do background subtraction process because it is a good algorithm to use for the classification of static postures and non-temporal pattern recognition, more robust than other models and it can handle multi-modal situations.

D. Smoothing :

Smoothing is used to filter or reduce noise produced from background subtraction process. The noise generated from background subtraction is typically in salt and pepper type in which using median filter is good to reduce the noise.

E. Shadow removal:

Background subtraction using GMM can also be used to detect shadow, so to increase object detection accuracy, the shadow should be removed. This

process will map foreground image produced from GMM to binary image which contain only pixel value 1 and 0 for identify moving object and background respectively.

F. Morphology Operation :

In this paper, morphology operation is used to reconstruct image after shadow removal process. Shadow removal process usually creates a gap inside object which if that gap is too large, it can make object will be separated that can affect object detection accuracy.

Advantages:

1. Accuracy is improved.
2. The startling losses both in human lives and finance caused by vehicle accidents.

SYSTEM ARCHITECTURE :

A mathematical procedure is used to recognize a vehicle feature. Principal Component Analysis (PCA) transforms a number of possible correlated variables, this variable will in turn transform in to a small number of uncorrelated variables. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. The most relevant information in the object will extracted to describe an object. This method is used to find the most appropriate computational model of an object.

In the traffic environment, it is important to require the stable and reliable vehicle detections recognized from the moving vehicle images, which is also called Detection of Road Vehicle. Detection of Road Vehicle has significance for future researches in Advanced Driver Assistance System (ADAS) and Automatic Driving System. Since after when a vehicle is detected, other applications can be applied more easily. For the past studies, there have been many methods proposed for image-based vehicle detection which mostly of them follow a two-stage pipeline which have the same similarity as with the visual object detections. One of the key to these issues is the feature extraction and classification of detection images for vehicle rear view. The process of training the neural network requires a great quantity of parameters, which often accompanies by overfitting problems. The

recognition results can achieve a higher degree of accuracy in the training set, but a lower degree of accuracy in the test set.

RESULTS

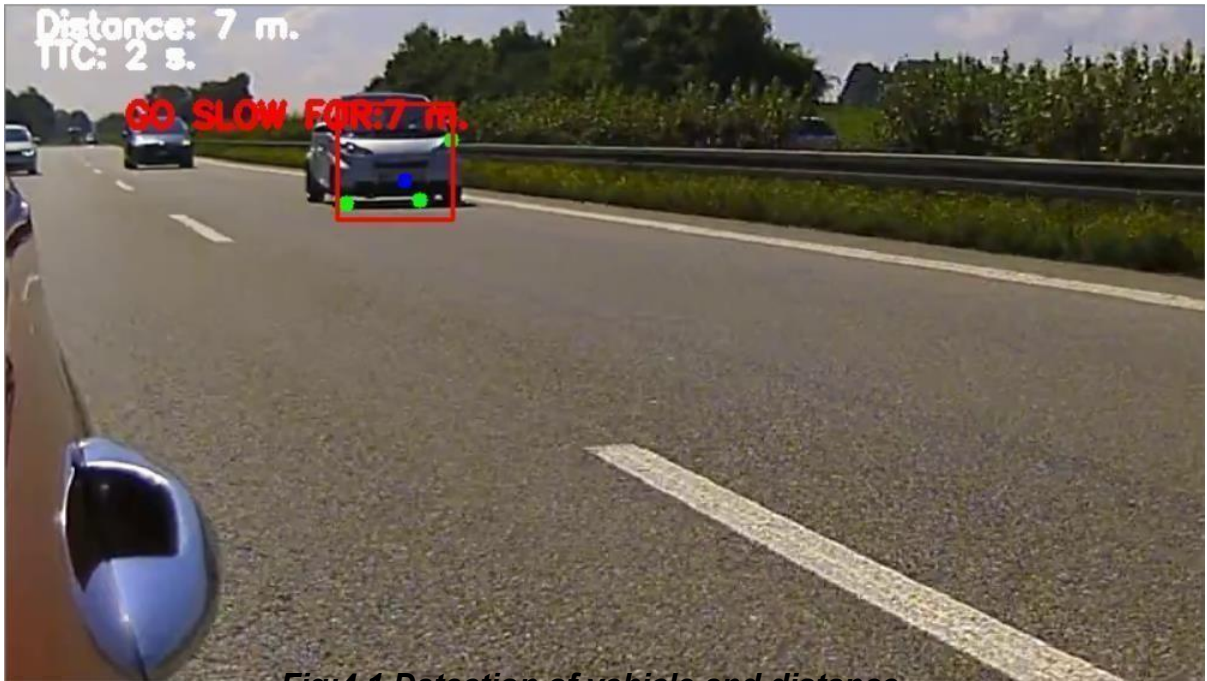


Fig:4.1 Detection of vehicle and distance

In the above figure, it shows the accurate distance of the vehicle and if the vehicle is getting near, it shows the alert message like "GO SLOW".

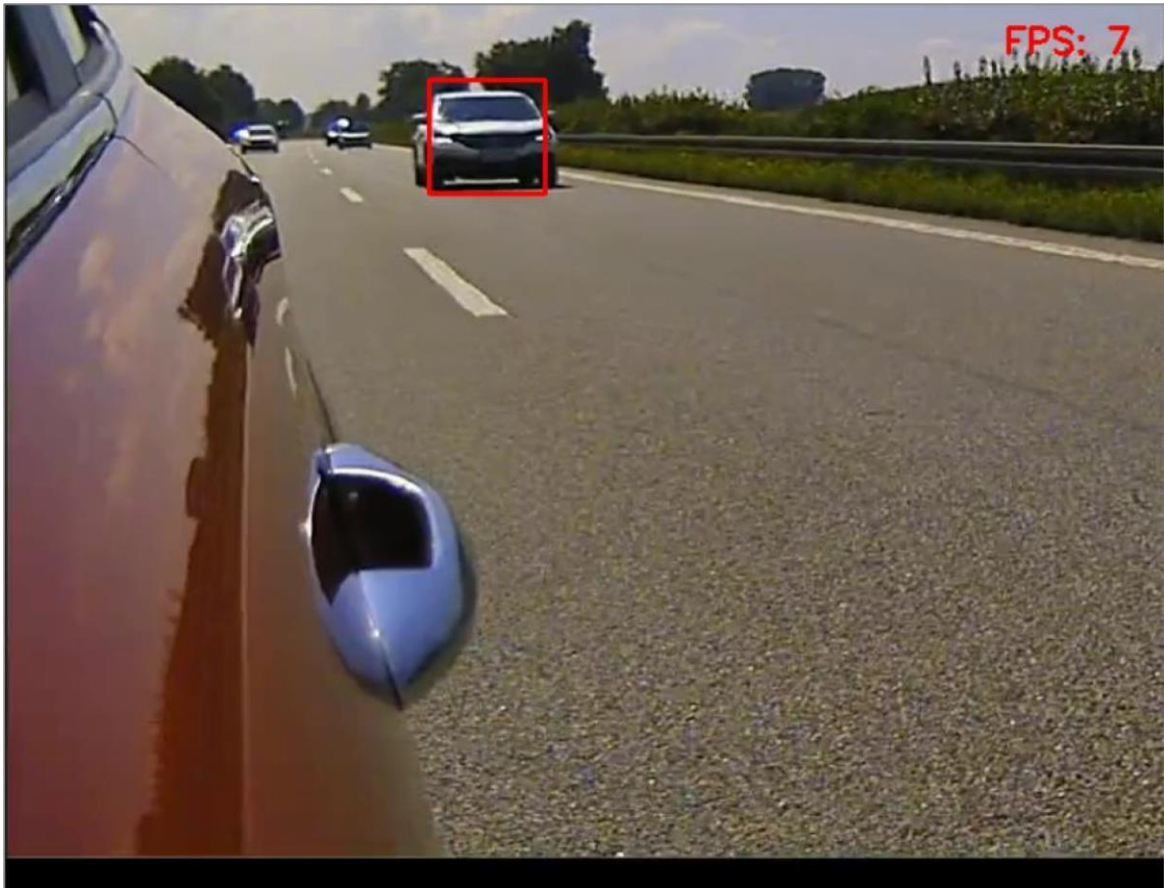


Fig 4.2 Detection of vehicle

CHAPTER 5

CONCLUSION AND FUTURE WORK

CONCLUSION :

There are many reasons to have an detection of car and its speed.It will control the road accidents and knowing advancely about the near cars.And we can estimate the nearer car and that speed.In addition to that it gives a conclusion to drive slowly This is suitable for everyone in this present generation.Which decreases the occura- ance of accidents.The other advantage of this to detect which vechicle is coming an of that speed.This is implemented successfully and is easy to use and also low cost product.

REFERENCES

https://en.wikipedia.org/wiki/History_of_Python

<https://www.python.org/downloads/>

<https://docs.python.org/3/reference/>

<https://www.upgrad.com/machine-learning-and-artificial->

[intelligence/?utm_source=Google&utm_medium=Search&utm_campaign=DV_ML_PGD_GOOGLE_SEARCH_Generic_Metros_All_BM&utm_content=AI_PGD_HighIntent_Exact&utm_term=artificial%20intelligence&gclid=Cj0KCQjwho7rBRDxARIsAJ5nhFqxePj](intelligence/?utm_source=Google&utm_medium=Search&utm_campaign=DV_ML_PGD_GOOGLE_SEARCH_Generic_Metros_All_BM&utm_content=AI_PGD_HighIntent_Exact&utm_term=artificial%20intelligence&gclid=Cj0KCQjwho7rBRDxARIsAJ5nhFqxePjOtSaYYrx11sxjBSuQf4xH0QcUEUrwJSu8L_nHbAu8vhjBgl8aAk3IEALw_wcB)

[OtSaYYrx11sxj](OtSaYYrx11sxjBSuQf4xH0QcUEUrwJSu8L_nHbAu8vhjBgl8aAk3IEALw_wcB)

BSuQf4xH0QcUEUrwJSu8L_nHbAu8vhjBgl8aAk3IEALw_wcB

APPENDIX

SOURCE CODE :

```
import cv2
import numpy
as np import
time

#path to classifier #path
to video file
cascade_src =
'myhaar.xml' video_src
= 'video//prvi.mkv'

#read video

#load classifier from file cap =
cv2.VideoCapture(video_src) car_cascade =
cv2.CascadeClassifier(cascade_src)

fps = 0

#defining borders of
ROI x1 = 0 y1 = 160
v = 720 hc = v - y1

#reading video frame by frame
while True:

#starting time of loop iteration    start_time
```

= time.tim

e() #read

frame

```

ret, img = cap.read() if
(type(img) == type(None)):

    break

#crop frame to get ROI img =
img[y1:y1+hc, x1:x1+720]

#convert to gray scale gray = cv2.cvtColor(img,
cv2.COLOR_BGR2GRAY)

#Detect objects with different dimensions in
frame #output of function is list of rectangles
#parameters: image, scaleFactor, minNeighbors, flags, minSize,
maxSize cars = car_cascade.detectMultiScale(gray, 1.1, 13, 0, (24, 24))

#drawing rectangle around detected object
#defining regions where object will show up
for (x, y, w, h) in cars:

    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 0, 255), 2)

#calculation of frame rate fps =
1.0/(time.time() - start_time)

#showing video with detected object cv2.putText(img, "FPS: " +
str(int(fps)), (620, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255),
2) cv2.imshow('video', img)

#press "esc" key to terminate
if cv2.waitKey(33) == 27:

    break;

```

```
cv2.destroyAllWindows()
cap.release()
```

CAR SPEED

```
import math as
m import
statistics as st
import time
```

```
import cv2 import
matplotlib.pyplot as
plt import numpy as
np
```

```
carCascade =
cv2.CascadeClassifier('myhaar.xml') video =
cv2.VideoCapture('video//prvi.mkv') lk_params =
dict(winSize = (15, 15),maxLevel = 2,criteria =
(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10,
0.03)) velocity = {}
```

```
def find_distance(x1, y1, x2, y2):
d = m.sqrt(m.pow(float(x2) - x1, 2) + m.pow(float(y2) - y1, 2))
return d
def
distance(w1,
w2):
    realWidth = 1.8
    f =
0.0022
    imageWidth =
```

```

800    objectWidth =
(w1 + w2) / 2.0
        sensorWidth =
0.005376

        distanceToObject = (f * realWidth * imageWidth) / (objectWidth *
sensorWidth)
        returndistanceToObject
        # print 'Distance to object: ' + str(distanceToObject) + ' m.'

```

```

def find_center(corners):
    x, y = 0, 0    for
corner in corners:

        y +=
corner[0][1]      x
+= corner[0][0]

center_row = int(1.0 * y / len(corners))
        center_col = int(1.0 * x /
len(corners))    return (center_col,
center_row)

```

```

def timeToCollision(v, d):

```

```

    ttc = (d / v)
        return
    ttc

```

```

#    Speed estimation
def estimate_speed(c1, c2, seconds, w1, w2,
carID):

    n = 0
    distance_pixels = find_distance(c1[0], c1[1], c2[0],
c2[1])    mpp = 1.8 / ((w1 + w2) / 2.0)    meters

```



```

= distance_pixels * mpp    v = meters / (seconds)
    v_kph = v * 3.6

    if carID in velocity.keys():
        i = len(velocity[carID])
        speed = []

        if i < 5:
            velocity[carID].append(v_kph)
        else:
            speed = velocity[carID][-5:]
    avg_speed = np.mean(speed)          # print
    avg_speed      if abs(v_kph - avg_speed) <
2:
velocity[carID].append(v_kph)

                # distance(w1, w2)
                print ('Car ' + str(carID) + ' new speed is: ' +
str(velocity[carID][-1:]))

        else:

                velocity[carID] = [v_kph]
                print ('Car ' + str(carID) + ' GO
SLOW: ')    return v

#    Detectio
n and tracking
def tracker():
    red = (0, 0, 255)
        blue =
(255, 0, 0)
            green = (0,
255, 0)
                frameCount

```

```

er = 0
    currentCarl
D = 0

#    dictionary for locations
previous_location = {}
current_location = {}

# dictionary for
trackers carTracker =
{}

# dictionary for
corners corners1 = {}
corners2 = {}
corners_updat
e = {}
corners_cente
r = {}
old_corners_c
enter = {}
width1 = {}
width2 = {}

speed = {}

                                time1
                                time2
                                while
                                True:

                                # read frame and check it, if it is not frame break

                                rc, image = video.read()
                                if type(image) ==
                                type(None):

```



```

        break
    # start time of iteration
    # crop frame
    # copy cropped frame
    # add 1 to frame counter
    # start = time.time()
    image = image[150:600,
150:950]          resultImage =
image.copy()
    frameCounter = frameCounter +
1

    # create empty list for trackers you need to
    delete carIDtoDelete = []

    # iterate trough dictionary of created trackers and if object leave
frame delete tracker          for carID in carTracker.keys():
    # get position of bounding box          trackedPosition =
carTracker[carID].update(image)          t_x, t_y, t_w, t_h =
trackedPosition[1]

        t_x = int(t_x)
        t_y =
        int(t_y)
        t_w =
        int(t_w)
        t_h =
        int(t_h)

        # x_center = t_x + 0.5 * t_w
        # y_center = t_y + 0.5 * t_h

    # if object leave frame add tracker to delete list if t_x +
t_w >= 750:

```

```

        carIDToDelete.append(carID)
    elif _y >= 570 or t_y <= 0:

        carIDToDelete.append(carID)
    elif _x <= 0:

        carIDToDelete.append(carID)

#    delete all trackers in
    delete list for carID in
        carIDToDelete:

        # print 'Tracker deleted: ' + str(carID) + '!'
        # print 'Current location deleted: ' + str(carID) + '!'
# print 'Previous location deleted: ' + str(carID) + '!'
        # print 'Corners 1 deleted: ' + str(carID) + '!'
        # print 'Corners 2 deleted: ' + str(carID) + '!'
        # print 'Width 1 deleted: ' + str(carID) + '!'
        # print 'Width 2 deleted: ' + str(carID) + '!'
        # print '\n'
    carTracker.pop(carID,      None)
    current_location.pop(carID,
None)
    previous_location.pop(carID, None)
        corners1.pop(carID, None)
        corners2.pop(carID, None)
        width1.pop(carID, None)
        width2.pop(carID, None)
        time1.pop(carID, None)
    time2.pop(carID, None)
        corners_center.pop(carID, None)
        old_corners_center.pop(carID, None)

# try to detect new object in frame in every 10
frames    if not (frameCounter % 10):

        # convert frame to grayscale    # try

```

to detect new object in frame

25

gray =

```

cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cars = carCascade.detectMultiScale(gray, 1.15, 15)

# if object is detected, save it location and calculate center point
of bounding box for (_x, _y, _w, _h) in cars:

    x = int(_x) +
7        y =
int(_y) + 7
    w = int(_w) - 5
    h = int(_h) -
5

    x_bar = x + 0.5 * w
    y_bar = y + 0.5 * h

    matchCarID = None

    # iterate trough tracked
    objects for carID in
carTracker.keys():

        # get object location and calculate center point
of tracked object

        trackedPosition =
carTracker[carID].update(image) t_x, t_y, t_w, t_h = trackedPosition[1]

        t_x = int(t_x)
        t_y =
int(t_y)        t_w =
int(t_w)        t_h =
int(t_h)

```

```

t_x_bar = t_x + 0.5 * t_w
t_y_bar = t_y + 0.5 * t_h

# if condition is true, detected object already have
tracker
if ((t_x <= x_bar <= (t_x + t_w)) and (t_y <= y_bar
<= (t_y + t_h)) and (x <= t_x_bar <= (x + w)) and (y <= t_y_bar <= (y + h))):
    matchCarID = carID

# if detected object don't have tracker yet, create new tracker and add
it to tracker dictionary

# save object location for speed
estimation            if matchCarID is None:
                        bbox = (x, y, w, h)
                        if bbox[0] + bbox[2] < 400 and bbox[1] < 100 and
bbox[0] > 70:
                            tracker =
cv2.TrackerMedianFlow_create()
                            tracker.init(image, bbox)
                            carTracker[currentCarID] = tracker
                            previous_location[currentCarID] =
                            bbox ROI = gray[y:y + h, x:x + h]
                            corners1[currentCarID] =
cv2.goodFeaturesToTrack(ROI, 10, 0.25, 5) corners1[currentCarID][:, 0,
                                0] += x
                                corners1[currentCarID][:, 0,
1] += y for i in corners1[currentCarID]:
                                    x, y = i.ravel()
                                    cv2.circle(resultImage, (x, y), 5,
green, thickness = -1)

#--

```

```

width1[currentCarID] = bbox[2]
old_corners_center[currentCarID] =
find_center(corners1[currentCarID])

#--
time1[currentCarID] = time.time()
currentCarID = currentCarID + 1

# in every frame iterate
# through trackers for carID in
# carTracker.keys():

# get position of object
trackedPosition = carTracker[carID].update(image)
t_x, t_y, t_w, t_h = trackedPosition[1]

t_x = int(t_x)
t_y =
int(t_y)
t_w = int(t_w)
t_h = int(t_h)

t_x_bar = t_x + 0.5 * t_w
t_y_bar = t_y + 0.5 * t_h
bbox = (t_x, t_y, t_w, t_h)
width2[carID] = bbox[2]

if len(corners1[carID]):
    ret, frame = video.read()
    if type(frame) ==
type(None):

        break
    frame = frame[150:600, 150:950]
    gray2 = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    corners2[carID], st, err = cv2.calcOpticalFlowPyrLK(gray,
gray2, corners1[carID], None, **lk_params)

```

```

        corners_center[carID] = find_center(corners2[carID])
    # print(corners_center[carID])
    cv2.circle(resultImage,
corners_center[carID], 5, blue, thickness = -1)
    for corner in
corners2[carID]:
cv2.circle(resultImage, (corner[0][0],
corner[0][1]), 5, green, -1)

    corners1[carID] = corners2[carID].copy()
    time2[carID] = time.time()
    gray = gray2.copy()

    # save location for speed estimation
    # draw new rectangle in frame
current_location[carID] = bbox

    cv2.rectangle(resultImage, (t_x, t_y), (t_x + t_w, t_y + t_h),
red,
2)

    #--
    # iterate trough
    locations for i in
corners_center.keys():
    sec = time2[i] - time1[i]

    if old_corners_center[i] != corners_center[i] and sec >=
0.01 and sec < 0.1:

        v = estimate_speed(old_corners_center[i],
corners_center[i], sec, width1[i], width2[i], i)
        print (v)
        d = distance(width1[i], width2[i])
        cv2.putText(resultImage, 'Distance: ' + str(int(d)) + ' m.',
(20, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (255, 255, 255),
3) cv2.putText(resultImage, 'TTC: ' +
str(int(timeToCollision(v, d))) + ' s.', (20, 40),
cv2.FONT_HERSHEY_SIMPLEX, 0.75,

```

```

(255, 255, 255), 3)
cv2.putText(resultImage, 'GO SLOW FOR:' + str(int(d)) +
'm.', (80, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0,
255), 3)
old_corners_center[i] =
corners_center[i]
time1[i] = time2[i]
# if len(width1):
width1[i] =
width2[i]

#--
# show results
# wait for esc to
terminate if cv2.waitKey(33)
== 27:
break
cv2.imshow('image', resultImage)

# close all open
cv2.destroyAllWindows()

if __name__ == '__main__':
tracker()

```