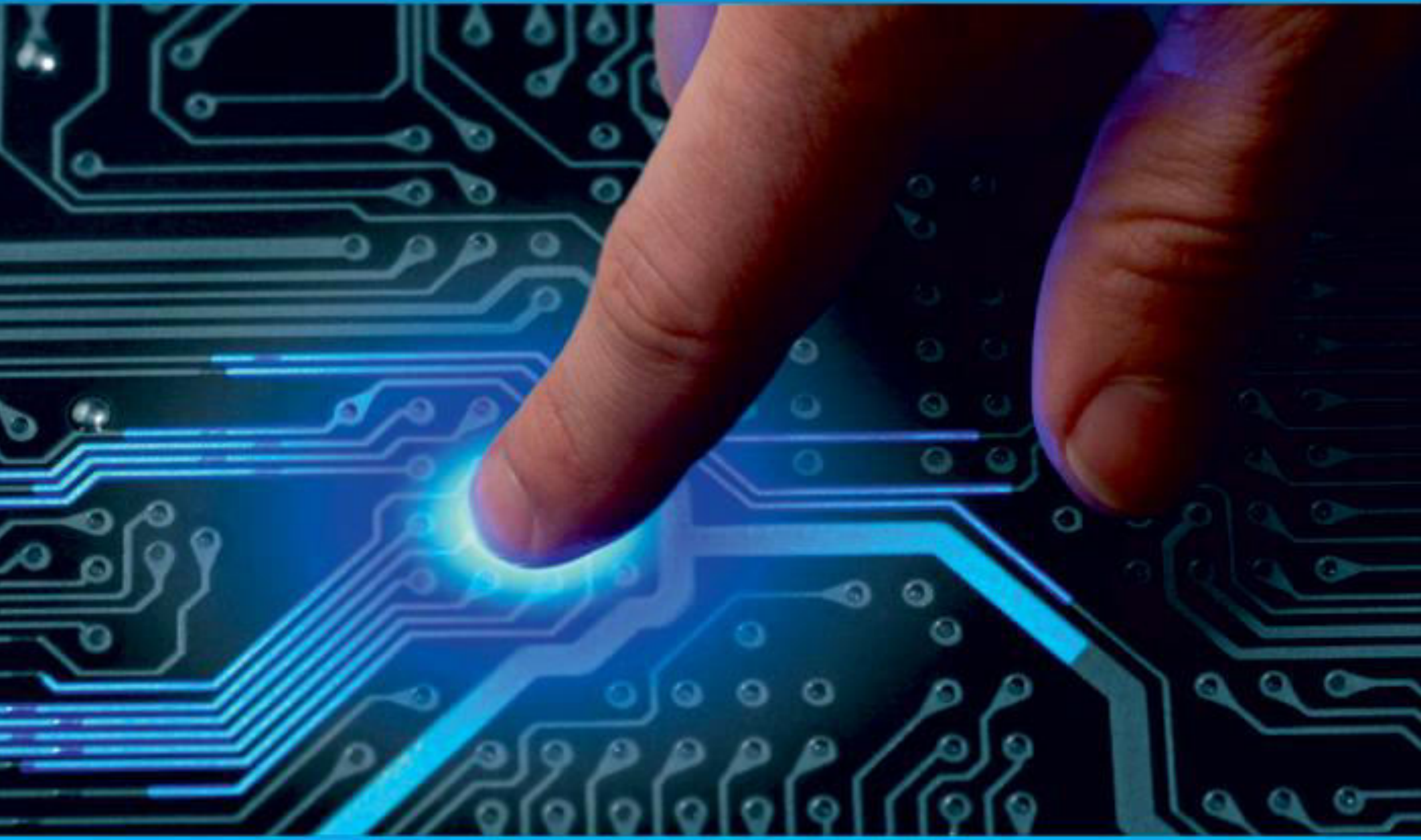


INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 7, July 2022

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007076 |

Video-Based Moving Vehicle Detection and Speed Estimation System using Machine Learning

Adviya Saba, Ayesha Siddiqua, Bharath R, Darshan S, HAMSA AS

UG Student, Dept. of Computer Science Engineering. ATME College of Engineering, Mysore, Karnataka, India

Assistant Professor, Dept. of Computer Science Engineering. ATME College of Engineering, Mysore, Karnataka, India

ABSTRACT: In the recent years we can see there is a vast increase in the number of vehicles all around the globe. Along with the increase in number of vehicles increases the number of accidents. Therefore, it is important to limit the speed of the vehicles at certain zones or areas. Radar speed measurement tools are commonly used for this purpose which can be inaccurate in certain cases such as in sensing smaller vehicles with weaker echoes. Also, it is difficult for these tools to detect vehicles changing in speeds too often or fast. Therefore, there is a need for a better technique to detect the speed of the moving vehicles. Then using expensive sensors such as radars, the vehicles video streaming could be used for this purpose. The video stream of the moving vehicle is given as an input, then it is passed through the filter for detecting its speed. Where Based on the speed, individual vehicle speed is detected and if the vehicle is over speeding then the warning is notified on the number so that more accident can be reduced.

KEYWORDS: DBSCAN and Guasian Algorithms, Back propagation Algorithms using Open CV python in Machine Learning.

I. INTRODUCTION

Vehicle speed detection in a complex environment is a hard task, since people interact with each other, form groups and may move in unexpected ways. This requires a robust method, which copes with the different motions, without being affected by occlusions and changes of environment features. To overcome changes in the environment monitored by the system, we have to design a robust background model that can deal with slow illumination changes like light changes between day and night, fast illumination changes like clouds blocking the sun. There are several successful vision systems for people detection and tracking. These systems use human features like head or body shape, leg symmetry analysis and statistical models which restrict them to human figures. They also need large number of pixels on target due to the shape based nature of the model which leads to miss identification of small vehicles. The main idea is that a simple form of skeletonization which only extracts the broad internal motion features of a target can be employed to analyze its motion. The background is the image which contains the non-moving vehicles in a video. Obtaining a background model is done in two steps: first, the background initialization, where we obtain the background image from a specific time of the video sequence. Second, the background maintenance, where the background is updated due to the changes that may occur in the real scene.

DISADVANTAGES OF EXISTING SYSTEM:

In these smart transportation and urban surveillance applications, cameras/imaging sensors are commonly installed to automatically detect and identify potential humans through automated motion detection methods. Usually, such automated motion detection methods demand high- complexity image/data processing technologies and algorithms. Hence, the design of low- complexity automated motion detection algorithms becomes an important topic in urban surveillance systems.

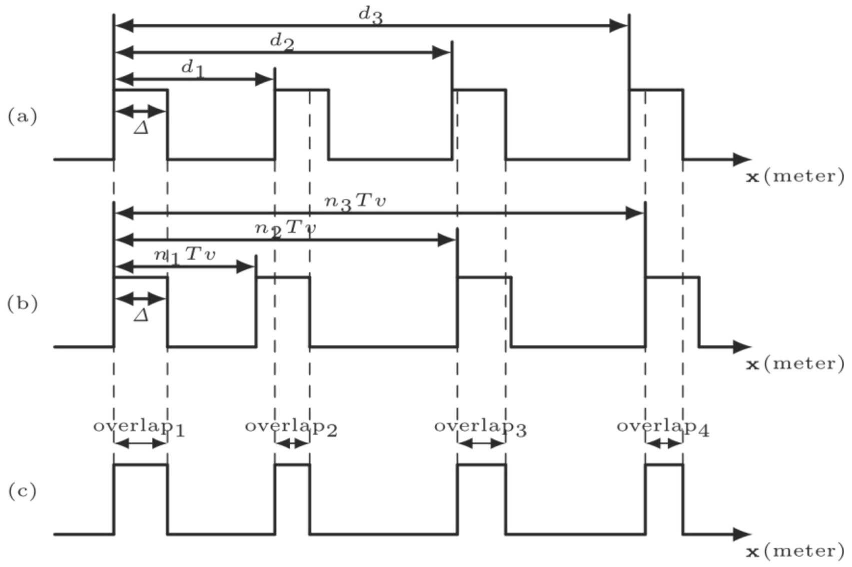
This process can be easily implemented in any urban surveillance systems in smart cities to pick out comparably important areas from the images captured by any camera/imaging sensors in urban environments, which will be a useful method not only to rapidly detect the important information but also to reduce the large data volume required to be stored because only those selected important (smaller) data will be stored as compared to huge raw data captured/generated from any cameras/imaging sensors 24 hours per day, 7 days a week and 365 days per year.

PROPOSED SYSTEM:

In this project, we propose a novel, simple and unified method to search the motions by filtering out the human and/or images rapidly from the digital camera imaging sensors. We design a simple filter to effectively detect either human or humans, motivated by the observation that in the most cases the motion is the highest energy frequency part of an image. This process can be easily implemented in any urban surveillance systems in smart cities to pick out comparably important areas from the images captured by any camera/imaging sensors in urban environments, which will be a useful method not only to rapidly detect the important information but also to reduce the large data volume required to be stored because only those selected important (smaller) data will be stored as compared to huge raw data captured/generated from any cameras/imaging sensors 24 hours per day, 7 days a week and 365 days per year

ADVANTAGES OF PROPOSED SYSTEM:

- Speed estimation model Intrusion detection is a method used to determine when an object crosses a virtual line and enters a region of interest. For video-based observations, the object’s position is acquired at discrete locations due to the temporal sampling rate. The sampling interval between the detected discrete locations is equal to the camera’s sampling time. Thus, the intrusion cannot be detected exactly at the line but instead within a detection distance (m). The detection distance is related directly to the camera’s sampling time T (s) and the hypothetical speed of the vehicle v (m/s), $\Delta = Tv$. (1) In other words, the detection distance is directly proportional to the vehicle’s speed and inversely proportional to the camera’s frame rate. The initial position (starting point) x1 (m) where the vehicle appears after the first intrusion line is assumed to be random. In the case of several intrusion lines, for successful detection, the vehicle should fall within the detection distances associated with each consecutive intrusion line. In this paper, a model is proposed for calculating a pdf of a passing vehicle’s speed based on a movement pattern vector. Hence, M intrusion lines ($M \geq 2$) should be placed on the receiving video frames of a stationary camera such that the relative distances between them in the real world along the road are known. Accordingly, the intruding vehicle is detected as soon as it crosses each line at a specific frame index. The main inputs of the model are the movement pattern vector n = [n0, . . ., nM−1] ∈ N and the distance vector of the intrusion lines d = [d0, . . ., dM−1] ∈ R+ 0 , which are defined as, nm = fm+1 − f1, (2) dm = lm+1 − l1, (3) where m ∈ {0, . . ., M − 1}, M is the number of intrusion lines, fm+1 is the video frame index number at which detection occurs, and lm+1 (m) is the position of the intrusion line in the real- world coordinate system (see Fig. 2). As mentioned earlier, given the relative placements of the intrusion lines in the distance vector d and the camera frame rate 1/T, a vehicle can appear at any point after the first intrusion line within . within which a vehicle traveling at a hypothetical speed should appear in order



•

a. The intrusion lines l_m , (b) The hypothetical vehicle's positions x_m within the maximum detection distance for a hypothetical speed v , (c) One illustration of vehicle's positions and the associated n_m s for a hypothetical speed v .

b. I **Fig. 1.** The common region g of all the distance overlaps based on the scenario in Fig. 3 associated with the probability of the hypothetical speed v .

In addition, there should be a common region between all the distance overlaps themselves (see Fig. 4) since the set of possible positions are generated by repetition according to the movement pattern vector, the camera frame rate and the vehicle's hypothetical constant speed. In other words, the detection distances at intrusion lines are not, in fact, continuous but rather discrete positions at which the vehicle can appear after each line provided the movement pattern vector ***n*** is satisfied. Therefore, the distance overlaps are valid if the intervals between them do not violate the movement pattern vector.

The common region of all distance overlaps is found via

$$g(v|\boldsymbol{n},\boldsymbol{d}) = \max\big(0, \min_m(d_m + (1 - n_m)Tv) - \max_m (d_m - n_m T v)\big), \qquad (4)$$

where $m \in \{0, 1, . . . , M - 1\}$, and $g(v \mid \boldsymbol{n}, \boldsymbol{d})$ is the common region of the distance overlaps. The pdf is obtained with

$$h_V(v \mid n, d) = \frac{g(v \mid n, d)}{\int_{v_{lower}}^{v_{upper}} g(v \mid n, d)\, dv}, \qquad (5)$$

where v_{lower} and v_{upper} are the lower and upper bounds, respectively, of the hypothetical speeds with $g(v \mid \boldsymbol{n}, \boldsymbol{d})$ greater than zero, and $h_V(v \mid \boldsymbol{n}, \boldsymbol{d})$ is the pdf for the stochastic speed V. Accordingly, the expected speed can be determined as

$$\mu_v = E[V] = \int_{v_{lower}}^{v_{upper}} v\, h_v(v \mid n, d)\, dv, \qquad (6)$$

where $E[V]$ is the expected speed of the vehicle given the movement pattern vector ***n*** and the intrusion lines distance vector ***d*** .

To sum up, the mathematical model that has been introduced is able to provide the pdf of the vehicles speed based on the input parameters. The input parameters are the camera sampling time T , the intrusion lines distance vector ***d*** and the movement pattern vector ***n*** . The learning phase for the system is based mainly on T and ***d*** , which are computed only

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007076 |

once in the beginning. The remaining parameter ***n*** is computed upon the passage of each vehicle in real-time; therefore, the model computes the speed of the vehicle instantly.

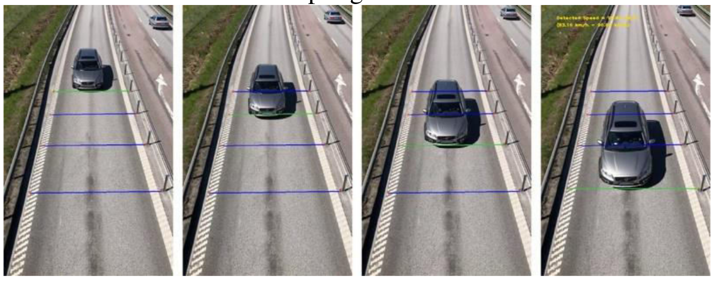
EXPERIMENTAL RESULTS AND DISCUSSION

This system focuses on employing multiple intrusion lines although the evaluation of the parameters under these circumstances can be quite complicated. When there are more than two intrusion lines, the system's error e would be reduced since the number of measurements would increase. The error e is considered as the absolute difference between the actual and expected speeds for each measurement. However, intrusion lines cannot be added indefinitely. The primary restriction is that the minimum distance between two intrusion lines should be greater than $Tv_{\max}$. Further restrictions include the geometrical properties of the scene, the size of the vehicles and the camera's resolution.

In this experiment, we used approximately two hours of data collected by cameras monitoring a major highway with normal to heavy traffic. The data contains information on trucks, trailers, cars and buses passing by at various speeds. The recordings were acquired simultaneously by one handheld camera and one smartphone with frame rates of $1/T$ = 50 fps and $1/T$ = 30 fps, respectively, in order to investigate the proposed model. The size of the input frame was 960 × 540 pixels. The distance between the first and last intrusion lines was 8.97 m, which was in accordance with the cameras' resolutions and the detection scene. Four intrusion lines were implemented, and each intrusion line was aligned to a guardrail pole as a reference point, see Fig. 1. The distance vector of the intrusion lines was then defined as d = [0 , 2 . 87 , 5 . 95 , 8 . 97] (m). The proposed system was realized on a common notebook platform and was able to operate in real time.

For every passing vehicle, a movement pattern vector was generated that indicated the frame number differences ***n*** between the four intrusion lines. According to the model derived in Section 2 . A, a speed pdf was associated with the movement pattern vector of each vehicle. The pdf provided a range of possible speeds (lower and upper bounds) and also the probability of each speed to occur given the movement pattern vector ***n*** . In order to evaluate the proposed method more accurately, we drove through the four intrusion lines with a GPS-equipped car and compared the detected speeds with the ones obtained via the GPS. Several runs were made at different constant speeds, and the motion data extracted from the vehicle included the time, location and speed, see . The speeds extracted from the navigator device were considered to be ground truth for our measurements since the error range of a GPS used under the same atmospheric conditions is very low .

As explained earlier, a higher frame rate provides a higher sampling rate, generating a smaller . It was also ob- served during the experiment that the relation between the sampling rate and the measurement error for various speeds .



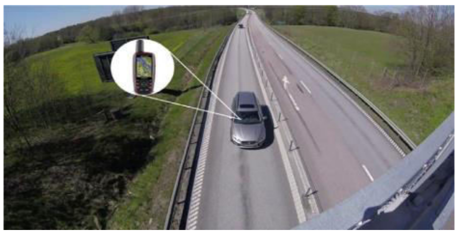


FIG.2 DETECTION OF CARS

II.IMPLEMENTATION

2.1Vehicle Detection import cv2 import dlib import time import threading import math carCascade = cv2.CascadeClassifier('myhaar.xml') video = cv2.VideoCapture('cars.mp4') WIDTH = 1280 HEIGHT = 720 def estimateSpeed(location1, location2): d_pixels = math.sqrt(math.pow(location2[0] - location1[0], 2) + math.pow(location2[1] - location1[1], 2)) # ppm = location2[2] / carWidht ppm = 8.8 d_meters = d_pixels / ppm #print("d_pixels=" + str(d_pixels), "d_meters=" + str(d_meters)) fps = 18 speed = d_meters * fps * 3.6 return speed def trackMultipleObjects(): rectangleColor = (0, 255, 0) frameCounter = 0 currentCarID = 0 fps = 0 carTracker = {} carNumbers = {} carLocation1 = {} carLocation2 = {} speed = [None] * 1000 5.2 Write output to video file out = cv2.VideoWriter('outpy.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10, (WIDTH,HEIGHT)) while True: start_time = time.time() rc, image = video.read() if type(image) == type(None): break image = cv2.resize(image, (WIDTH, HEIGHT)) resultImage = image.copy() frameCounter = frameCounter + 1 carIDtoDelete = [] for carID in carTracker.keys(): trackingQuality = carTracker[carID].update(image) if trackingQuality < 7: carIDtoDelete.append(carID) for carID in carIDtoDelete: print ('Removing carID ' + str(carID) + ' from list of trackers.') print ('Removing carID ' + str(carID) + ' previous location.') print ('Removing carID ' + str(carID) + ' current location.') carTracker.pop(carID, None) carLocation1.pop(carID, NonecarLocation2.pop(carID, None) if not (frameCounter % 10): gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY) cars = carCascade.detectMultiScale(gray, 1.1, 13, 18, (24, 24)) for (_x, _y, _w, _h) in cars: x = int(_x) y = int(_y) w = int(_w) h = int(_h) x_bar = x + 0.5 * w y_bar = y + 0.5 * h matchCarID = None for carID in carTracker.keys(): trackedPosition = carTracker[carID].get_position() t_x = int(trackedPosition.left()) t_y = int(trackedPosition.top()) t_w = int(trackedPosition.width()) t_h = int(trackedPosition.height()) t_x_bar = t_x + 0.5 * t_w t_y_bar = t_y + 0.5 * t_h 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 matchCarID is None: print ('Creating new tracker ' + str(currentCarID)) tracker = dlib.correlation_tracker() tracker.start_track(image, dlib.rectangle(x, y, x + w, y + h)) carTracker[currentCarID] = tracker carLocation1[currentCarID] = [x, y, w, h] currentCarID = currentCarID + 1 #cv2.line(resultImage,(0,480),(1280,480),(255,0,0),5) for carID in carTracker.keys(): trackedPosition = carTracker[carID].get_position() t_x = int(trackedPosition.left()) t_y = int(trackedPosition.top()) t_w = int(trackedPosition.width()) t_h = int(trackedPosition.height()) cv2.rectangle(resultImage, (t_x, t_y), (t_x + t_w, t_y + t_h), rectangleColor, 4).

Table 1

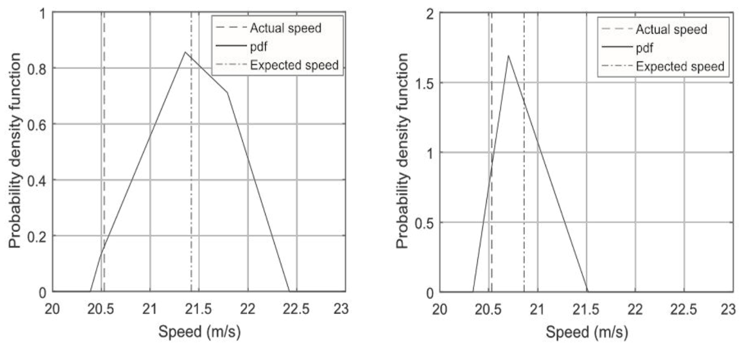
The performance evaluation of the proposed method at a frame rate of $\frac{1}{T}$ = 30 fps with d = [0 , 2 . 87 , 5 . 95 , 8 . 97] (m).

No.	Actual speed (m/s)	Movement pattern vector	Detected speed (m/s) ($v_{\text{lower}} - v$ upper) μ_v	Error rate (%)
1	20.5	n		1.46
2	26.2	n	(20.3 – 21.5) 20.8 (23.1 – 26.9) 25.2 (19.2 22.3) 20.6	3.82
3	20.0	n	–	3.00
4	25.3	n = [0 , 4 , 7 , 11]	(23 . 1 − 26 . 9) 25.2	0.39

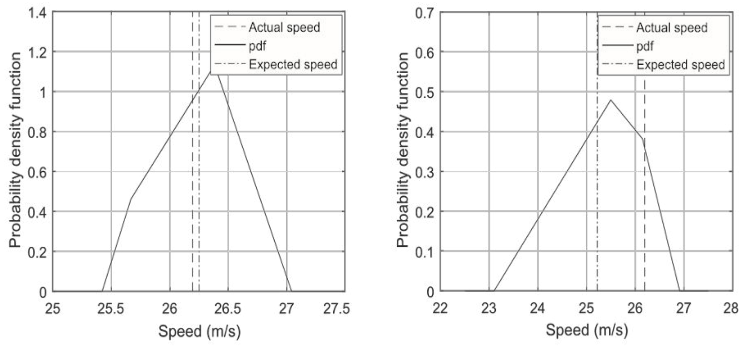
Table 2

The performance evaluation of the proposed method at a frame rate of $\frac{1}{T}$ = 50 fps with d = [0 , 2 . 87 , 5 . 95 , 8 . 97] (m).

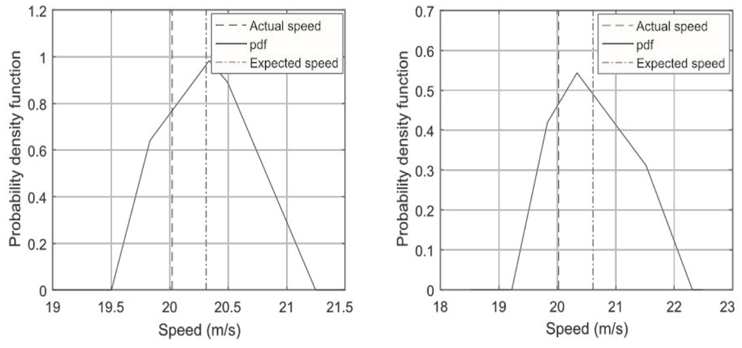
No.	Actual speed (m/s)	Movement pattern vector	Detected speed (m/s) ($v_{\text{lower}} - v$ upper) μ_v	Error rate (%)
1	20.5	n = [0 , 7 , 14 , 21] n = [0 , 6 , 12 , 17]	(20 . 4 − 22 . 4) 21.4	4.39
2	26.2		27 . 0) 26.2	0.00
3	20.0	n		1.50
4	25.3	= [0, 7, 15, 22] n = [0, 6, 12, 18]	(19.5 – 21.2) 2((23.6 – 26.4) 2!	1.18


Probability density function
Speed (m/s)
Actual speed
pdf
Expected speed
0
0.2
0.4
0.6
0.8
1
20
20.5
21
21.5
22
22.5
23
0
0.5
1
1.5
2

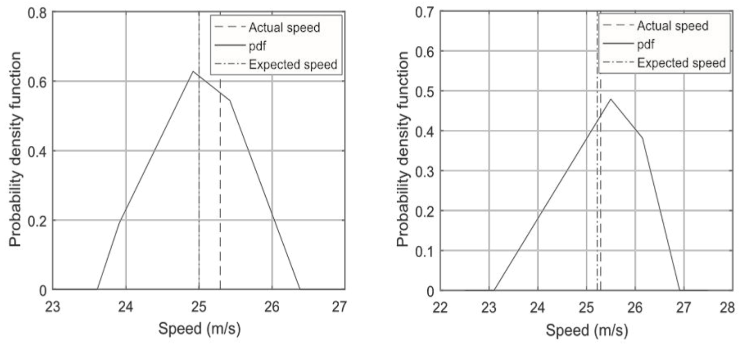
(a)


Probability density function
Speed (m/s)
Actual speed
pdf
Expected speed
0
0.2
0.4
0.6
0.8
1
1.2
1.4
25
25.5
26
26.5
27
27.5
0.1
0.3
0.5
0.7
22
23
24
25
26
27
28

(b)


Probability density function
Speed (m/s)
Actual speed
pdf
Expected speed
0
0.2
0.4
0.6
0.8
1
1.2
19
19.5
20
20.5
21
21.5
0.1
0.3
0.5
0.7
18
22
23

(c)


Probability density function
Speed (m/s)
Actual speed
pdf
Expected speed
0
0.2
0.4
0.6
0.8
23
24
25
26
27
0.1
0.3
0.5
0.7
22
28

(d)

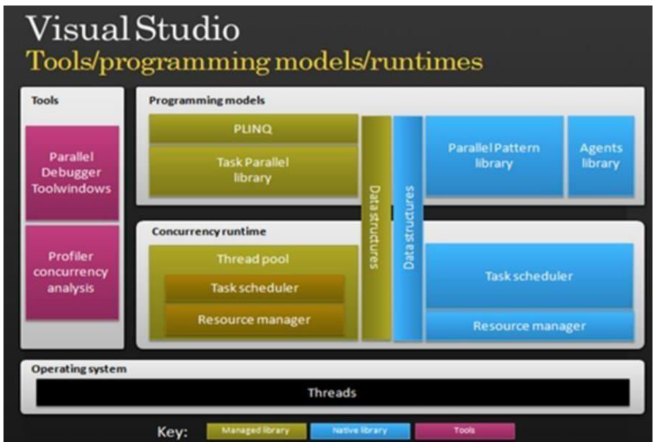


Fig1.1 Visual Studio Model

The visual studi0 which includes the b0th web based applicati0ns and web server applicati0ns which will be devel0ped 0n the client and the server side. The visual studi0 mainly c0nsist 0f the .Net applicati0ns to build easily and to implement it in a understandable way. The Micr0s0ft has given visual studi0 as a free of cost for the users and also other commercial versi0ns al0ng with the visual studi0 versions are available for free in the Microsoft. It supports for the many 0ther languages like ruby rails, python are available language service will be installed individually. In the past the java script and css are supported by this visual studio application.

III.METHODOLOGY

I. OVERVIEW 0F .NET FRAME W0RK. IT IS A NEW M0DEL F0R C0MPUTING AND DEPL0YMENT 0F APPLICATI0NS. .NET MICR0S0FT VERSI0N S0FTWARE AS A SERVICE IN WHICH THE DEVEL0PMENT CAN WE BUILD. IT IS N0T A SINGLE APPLICATI0N IT IS ACTUALLY A C0LLECTI0N 0F DIFFERENT TECHN0L0GIES. THE .NET LANGUAGE USES IN THE LANGUAGES LIKE VISUAL BASIC, C, C++, ETC. S0 T0 RUN THIS APPLICATI0N THE MS VISUAL STUDI0 2010 IS ACTUAL SUITABLE. IT ASSURES THE SAFETY 0F THE C0DE THAT IS EXECUTING. IT IS THE NATIVE .NET APPLICATI0N DEVEL0PMENT. IT HAS A DIFFERENT TECHN0L0GY WHICH INCLUDES;- THE .NET LANGUAGE. ASP .NET LANGUAGE. WIND0WS F0RM APPLICATI0N. VISUAL STUDI0 .NET. 0BJECT C0DE IS ST0RED AND EXECUTED IN 0BJECT 0RIENTED C0NSISTENTLY. IT IS A SAFE EXECUTE 0F C0DE AND IT INCLUDE THE C0DE CREATED BY THE 0THER UNKN0WN 0R THIRD PARTY. IT ALS0 USES A WEB BASED APPLICATI0NS. IT ELIMINATES THE PR0BLEMS PERF0RMANCE 0F INTERRUPTED ENVIR0NMENTS.

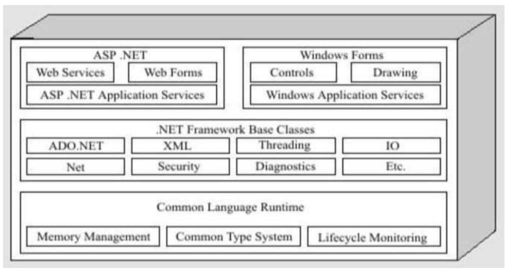


Fig 1.2 .NET Framew0rk 0vervie

MSSQL is an back end it st0res all the c0mp0nents 0f fr0untend data fr0m data fetch fr0m the fr0nt end and using primary meth0d t0 retrieve data and st0re The applicati0n is mainly t0 c0llect the data inf0rmati0n 0n the empl0yee w0rks and pr0ject in 0ur g0verning process.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165

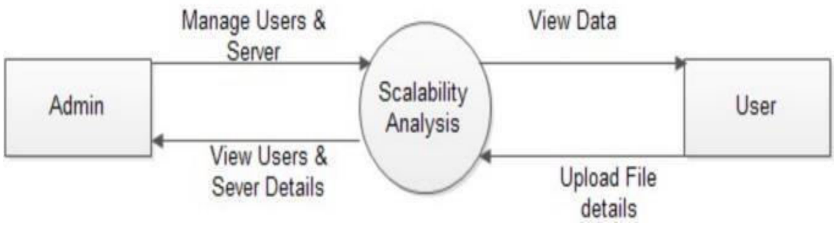
|| Volume 10, Issue 7, July 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1007076 |

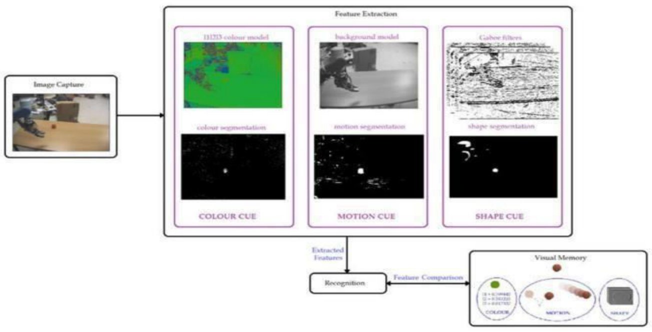
LITERATURE SURVEY

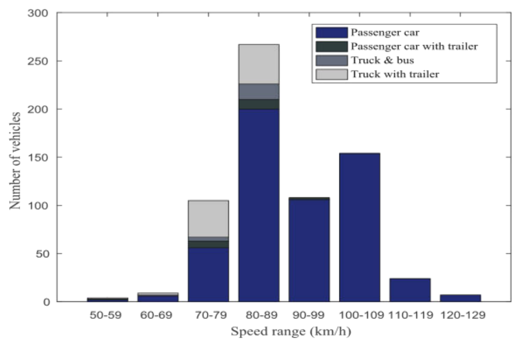
II. [1] Vehicle Speed Detection using Python Author : MD MUBEEN KHAN, K SRINIVAS Year : 2020 This paper targets to predict the speed of a vehicle with respect to the data from a recorded video source. Serving as the hypothesis, the paper portrays the various important procedures such as unequivocal Gaussian blend, models, DBSCAN, Kalman channel, Optical stream. [2] Vehicle Detection, Tracking and Speed Measurement for Traffic Regulation Author : Pulli Harsha Samhitha, Allu Naga Jyothi, Ramana Vesapogu, Manasa Mannem ,S. Sri Harsha Year : 2019 In this paper we present a concise image processing method in developing traffic surveillance systems. The proposed model implements enhanced preprocessing, background Subtraction,

III. **Context Diagram** show a system, often software-based, as a whole and its inputs and outputs from/to external factors. Context Diagrams represent all external entities that may interact with a system. Such a diagram pictures the system at the center, with no details of its interior structure, surrounded by all its interacting systems, environments and activities. The objective of the context diagram is to focus attention on external factors and events that should be considered in developing a complete set of systems requirements and constraints.



Architectural Design





IV. CONCLUSION

The presented video-based speed measurement model provides the probability density function of a passing vehicle's speed based on its movement pattern vector. In this work, the proposed method was implemented using four intrusion lines, and the input frames were captured simultaneously by an off-the-shelf handheld camera and a smartphone device with frame rates of 50 fps and 30 fps, respectively. According to the experimental results, the average error rates of the speed measurement system at 50 fps and 30 fps were 1.77% and 2.17%, respectively, for vehicles travelling in the range of 70 km/h– 100 km/h. The error rate decreased noticeably with an increase in the frame rate, as expected. In addition, the actual speeds of the vehicles were within the range of the estimated pdfs, increasing the confidence in the model. It was also observed that the ranges of obtained pdfs often decreased with the increase in the camera frame rate. Furthermore, the proposed model was employed to measure the speed of more than 670 vehicles, and the results were compared with the speed limits with respect to the vehicles categories passing through the measurement site.

V. FUTURE ENHANCEMENT

1. Later on we will build the power of our calculations by taking care of such mistakes powerfully. 2. In expansion, because of this execution corruption blunder, we will consider SLA arrangement in Cloud figuring conditions to improve the vigor. 3. We will likewise include diverse sort of administrations and other valuing systems, for example, spot valuing to expand the benefit of specialist organization. In addition, to research the knowledge based confirmation control and booking for boosting a SaaS providers benefit is one of our future bearings for improving our algorithms time multifaceted nature.

REFERENCES

[1] Vinay D R, N Lohitesh Kumar, " Vechile Tracking Using Background Subtraction Algorithm" in International Journal of Engineering Research and General Science Volume 3, Issue 1, JanuaryFebruary, 2015. [2] Rohan K. Naik, " A Robust Background Subtraction Technique for Vechile Detection" in the International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 2, February 2015. [3] Niranjil Kumar A and Suresh kumar, "Background Subtraction in Dynamic Environment based on Modified Adaptive GMM with TTD for Moving Vechile Detection" in the J Electr Eng Technol.2015;10(1): 372-378 http://dx.doi.org/10.5370/JEET.2015. [4] Asaad AMA, Syed IA (2009). "Object identification in video images using morphological background estimation scheme" Chapter, 22: 279-288. [5] Bailo G, Bariani M, Ijas P, Raggio M (2005). "Background estimation with Gaussian distribution for image segmentation, a fast approach," Proc. IEEE Intl. workshop on Measurement Systems for Homeland Security, Contraband Detection and Personal safety, pp. 2-5. [6] Ferrier NJ, Rowe SM, Blake A (1994). Real--time traffic monitoring. In WACV94, pp. 81—88. Fumio Y, Wen L, Thuy TV (2008). "Vehicle Extraction And Speed Detection From Digital Aerial Images" IEEE International Geosciences and Remote Sensing Symposium, pp. 1134-1137. [7] Huei-Yung L, Kun-Jhih L (2004). "Motion Blur Removal and Its Application to Vehicle Speed Detection", The IEEE International Conference on Image Processing (ICIP 2004), pp. 3407-3410. Rad et al. 2563 [8] Jianping W, Zhaobin L,