RAMAIAH
Institute of Technology

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

RAMAIAH INSTITUTE OF TECHNOLOGY
(AUTONOMOUS INSTITUTE AFFILIATED TO VTU)
M. S. R. I. T. POST, BANGALORE – 560054
2022-2023

Mini Project Report

Computer Vision based Traffic Monitoring and Analysing from On-Road Videos

For the subject Computer Vision (ISE555)

Submitted By:

1. VIVEK B V-1MS20IS135
2. SRI VATSA V-1MS20IS116
3. SUMANTH B S-1MS20IS119
4. SHREYAS S-1MS20IS111

Submitted To:

Dr.Megha P Arakeri
Associate Professor,
Dept. of ISE, RIT

INDEX

ABSTRACT

A modern and practical traffic system requires intensive traffic monitoring and analysis. However, it is a very difficult task because the traffic situation is dynamic, making it very difficult to maintain the traffic in the conventional way. It is also necessary for large, crowded cities to design a smart traffic system. In this article, we suggest a vision-based traffic monitoring system that will aid in judicious traffic system maintenance. We also produce a traffic analysis for a specific time period, which is useful for creating a practical and smart traffic system for a crowded city. To identify vehicles in a video using the suggested method, we use an Adaboost classifier based on Haar features.

Introduction

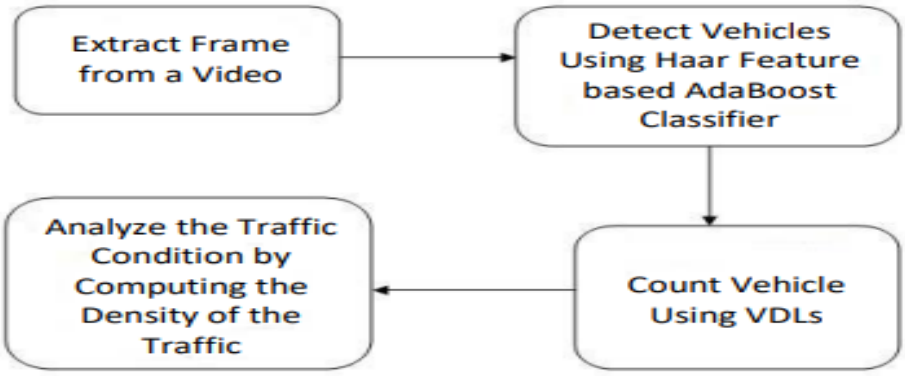
In the developed world over the past ten years, the field of vision-based traffic surveillance systems has seen a surge in interest. Given that it has some important applications, this is a possible area for research. For the purpose of obtaining accurate information on traffic flow and spotting traffic events, it is important to digitally process and analyze these videos in real-time. The least crowded routes and travel time estimates can be computed, and traffic density in major arteries can be estimated. By counting the number of vehicles that use the roads, this information can be obtained. The authority can use this to measure rush hour traffic on a specific road at a specific time and alleviate the issue.

To be put into practice, the suggested method requires two key components:

1. Locating and counting vehicles.
2. Generating the data required for traffic maintenance.

Methodology

Our approach is suggested for real-time applications. Therefore, both detection, counting, and analysis must have faster execution times. For real-time traffic surveillance, we suggest a quicker, less expensive model that is also simple to use. The approaches and techniques we employ to identify and count vehicles in a video and then produce traffic data will be covered in the following subsections.



Our approach is suggested for real-time applications. Therefore, both detection, counting, and analysis must have faster execution times. For real-time traffic surveillance, we suggest a quicker, less expensive model that is also simple to use. The approaches and techniques we employ to identify and count vehicles in a video and then produce traffic data will be covered in the following subsections.

Implementing the suggested method requires two key steps: I Locating and counting vehicles, ii) Producing the data required to

keep the flow going. For vehicle detection, numerous methods have been developed.

a) Recognition

i. Camera placement: During the initial stages of detection, we must choose an appropriate location to set up the camera. We suggest using our method to watch videos captured by a stationary camera.

ii. Feature Extraction: Because we plan to use our method in real time, we must choose a feature that can be quickly computed. The proposed method selects features that are similar to the haar for vehicle detection.

The Viola-Jones object detection framework is used for the detection. The input image is covered by a window that is the target size, and the Haarlike feature is calculated for each area of the image.

The use of cameras for vehicle detection and classification has increased because t hey are more affordable than radar or lidar systems.

Despite a significant increase in computational power, classifying and detecting v ehicles is a difficult task.

The dynamic environment of the road is the issue.

One cannot predict the state of the road.

This task is challenging because there may be numerous pedestrians and human-made infrastructures.

Additionally, there are background changes, optical illusions, and a diversity of ve hicles.

Implementation

Detection.py

```
import cv2

# capture video/ video path
cap = cv2.VideoCapture('cars.mp4')
#viola jones algorithmn starts

#use trained cars XML classifiers
car_cascade = cv2.CascadeClassifier('haarcascade_cars.xml')


#read until video is completed
while True:
    #capture frame by frame
    ret, frame = cap.read()
    #convert video into gray scale of each frames
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #detect cars in the video
    cars = car_cascade.detectMultiScale(gray, 1.1, 3)
    #cv2.im_write(cars)

    #to draw a rectangle in each cars
    for (x,y,w,h) in cars:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
        cv2.imshow('video', frame)
        crop_img = frame[y:y+h,x:x+w]

     #press Q on keyboard to exit
    if cv2.waitKey(1000) & 0xFF == ord('q'):
        break


#release the video-capture object
cap.release()
```

```
#close all the frames
cv2.destroyAllWindows()
```

Count.py

```
import cv2
import numpy as np

# capturing or reading video
#cap = cv2.VideoCapture(0)
cap = cv2.VideoCapture('video.mp4')


# adjusting frame rate
fps = cap.set(cv2.CAP_PROP_FPS,0.25)


# minimum contour width
min_contour_width=40  #40

# minimum contour height
min_contour_height=40  #40
offset=10   #10
line_height=550  #550
line_height2=450  #550

matches =[]
cars=0

# defining a function
def get_centroid(x, y, w, h):

    x1 = int(w / 2)
    y1 = int(h / 2)
```

```
    cx = x + x1
    cy = y + y1
    return cx,cy
    return [cx, cy]


cap.set(3,5920)
cap.set(4,5080)

if cap.isOpened():
    ret,frame1 = cap.read()
else:
    ret = False
ret,frame1 = cap.read()
ret,frame2 = cap.read()

while ret:
    d = cv2.absdiff(frame1,frame2)
    grey = cv2.cvtColor(d,cv2.COLOR_BGR2GRAY)

    blur = cv2.GaussianBlur(grey,(5,5),0)

    ret , th = cv2.threshold(blur,20,255,cv2.THRESH_BINARY)
    dilated = cv2.dilate(th,np.ones((3,3)))
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (2, 2))

    # Fill any small holes
    closing = cv2.morphologyEx(dilated, cv2.MORPH_CLOSE, kernel)
    contours,h =
cv2.findContours(closing,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
    for(i,c) in enumerate(contours):
        (x,y,w,h) = cv2.boundingRect(c)
        contour_valid = (w >= min_contour_width) and (
                h >= min_contour_height)

        if not contour_valid:
            continue
        cv2.rectangle(frame1,(x-10,y-10),(x+w+10,y+h+10),(255,0,0),2)

        cv2.line(frame1, (0, line_height), (1200, line_height), (0,255,0), 2)
        cv2.line(frame1, (0, line_height2), (1200, line_height2), (0,255,0), 2)
```

```
        centroid = get_centroid(x, y, w, h)
        matches.append(centroid)
        cv2.circle(frame1,centroid, 5, (0,255,0), -1)
        cx,cy= get_centroid(x, y, w, h)
        for (x,y) in matches:
            if (line_height + offset) > y > (line_height - offset):
                cars=cars+1
                matches.remove((x,y))
                print(cars)

                print(cars/3)

    cv2.putText(frame1, "Total Vehicles Detected: " + str(cars), (10, 90),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                (0, 170, 0), 2)

    cv2.putText(frame1, "density: " + str(cars/3), (30, 130),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                (0, 170, 0), 2)


    #cv2.drawContours(frame1,contours,-1,(0,0,255),2)



    cv2.imshow("OUTPUT" , frame1)
    #cv2.imshow("Difference" , th)
    if cv2.waitKey(14) == 5000000:
        break
    frame1 = frame2
    ret , frame2 = cap.read()

    target = 5
counter = 0
while ret:
  if counter == target:
    ret, frame2 = cap.read()
    # display and stuff
    counter = 0
  else:
    ret = cap.grab()
```

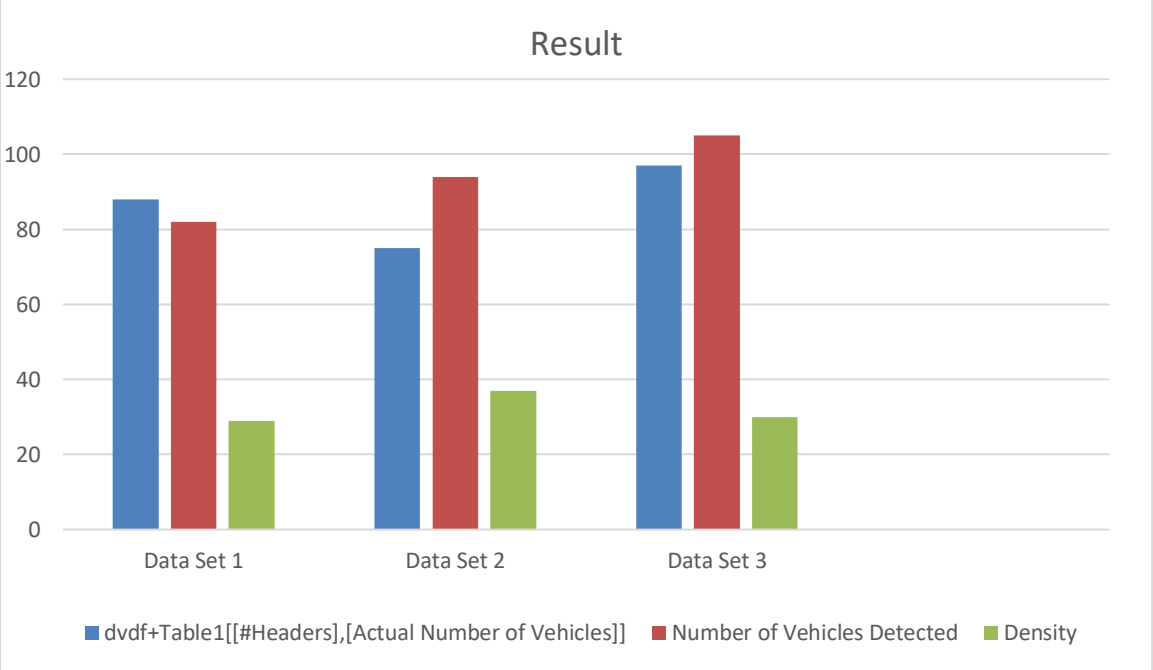
```
    counter += 1

#print(matches)
cv2.destroyAllWindows()
cap.release()
```

Results

Multiple experiments have been done to generate the result of the method. All the experiments have been done on different videos.
After counting the vehicles for a particular time , the proposed method provides the information about the density. Experiments have been carried out to generate the result of the proposed method. Without considering occlusion.

Data Set	Detection Accuracy	False Positive Rate	Duration (Secs)
Data Set 1	92	1	
Data Set 2	91	2	
Data Set 3	92	1	

Data Set	Actual Number of vehicles	Number of vehicles detected	Density
Data Set 1	**88**	**82**	**29**
Data Set 2	**75**	**94**	**37**
Data Set 3	**97**	**105**	**32**



Output images

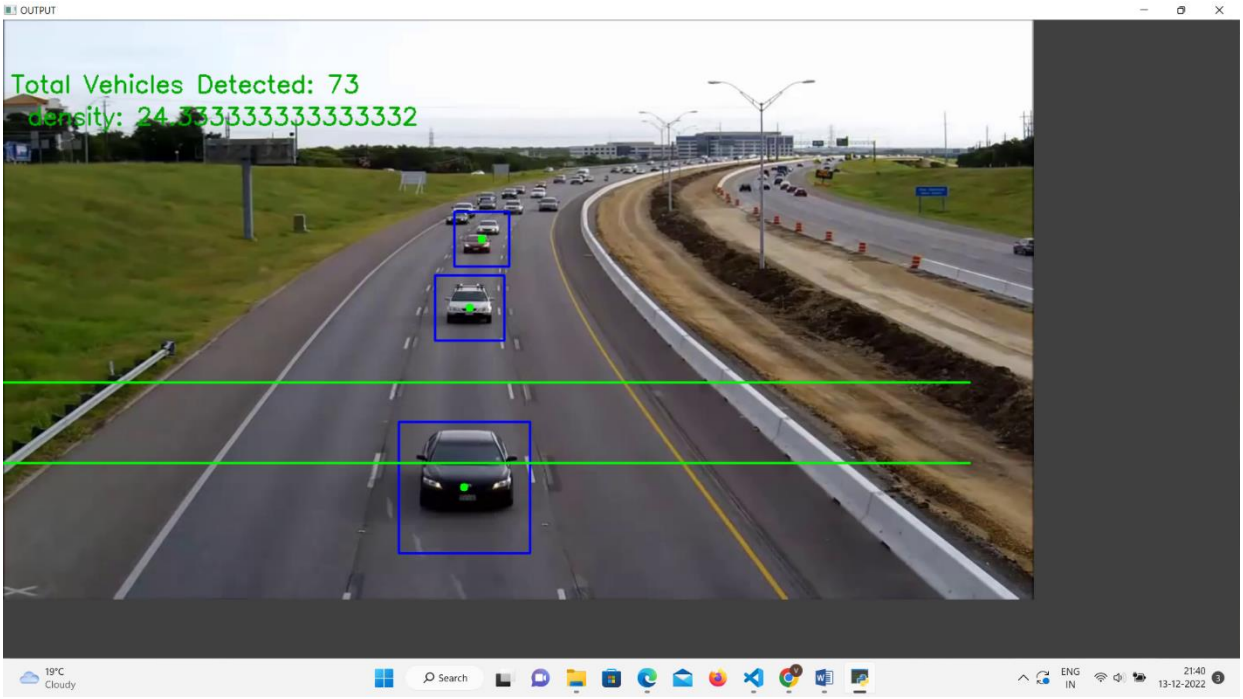


Fig 1

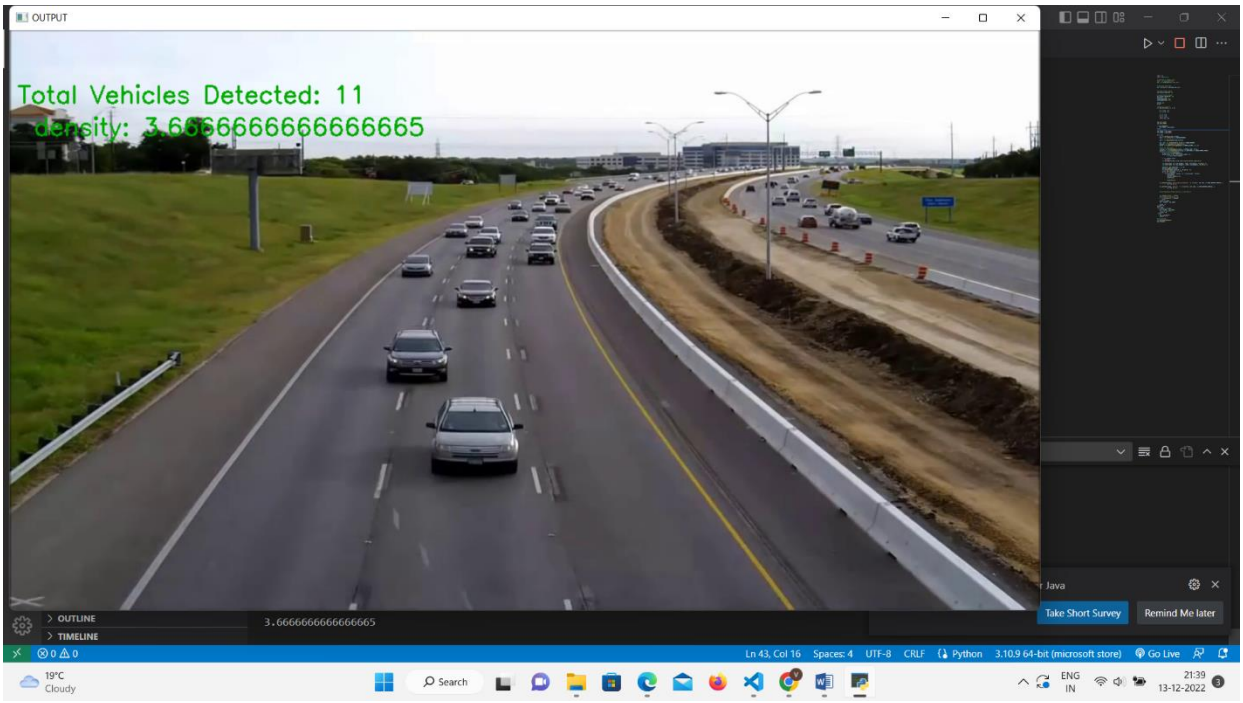


Fig 2

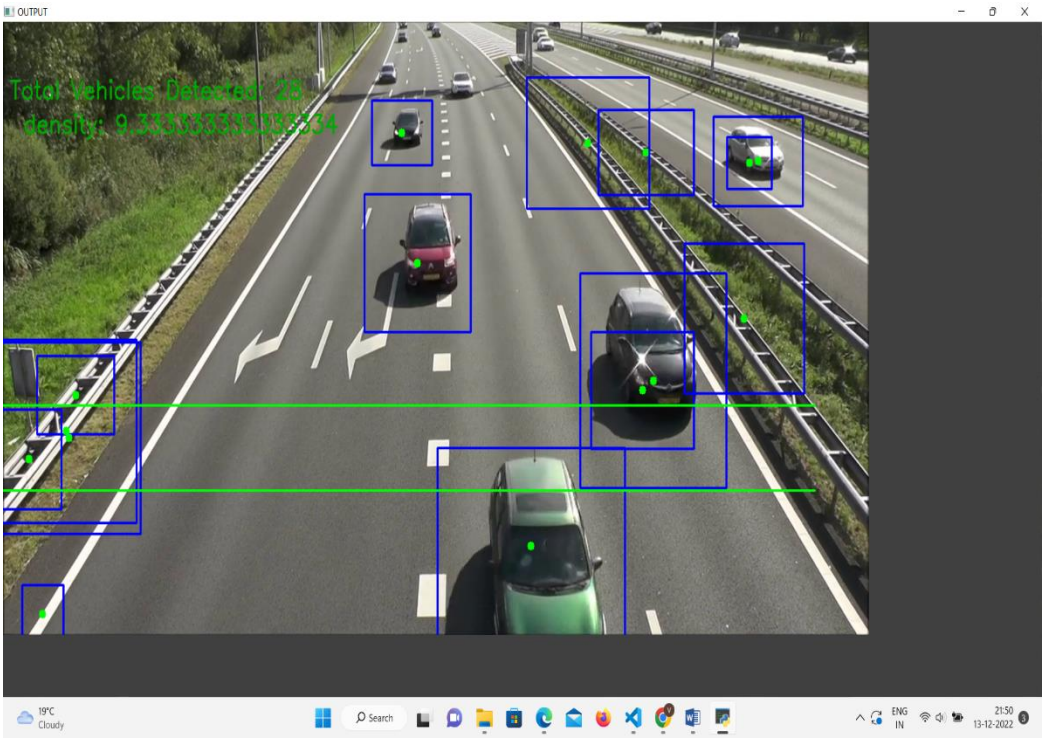

OUTPUT
Total Vehicles Detected: 28
density: 9.333333333333334
19°C
Cloudy
Search
ENG
IN
21:50
13-12-2022

Fig 3

Project schedule

Sl No	Process	Date
1	Topic Selection	28-10-2022
2	Idea Presentation	14-11-2022
3	Code Implementation	29-11-2022
4	Report Making	11-12-2022
5	Report Submission	15-12-2022

Conclusion

The suggested approach can offer a solution because it is affordable and simple to use.The suggested method makes use of an Adaboost classifier based on Haarlike features, which is quicker to compute and offers very good detection accuracy. To count the vehicles, two virtual detection lines (VDL) are used.The two VDL's difference is calculated in a way that reduces the possibility of missing or counting twice.Although the results are encouraging, the algorithm still requires more changes because the method's accuracy suffers when there is a lot of traffic.

References

- https://computerresearch.org/index.php/computer/article/view/1836/3-Computer-Vision-based-Traffic_htmlRecognizing human emotion using computer vision
- https://www.mygreatlearning.com/blog/viola-jones-algorithm/The dataset the used is Facial Expressions Recognition-2013(FER-2013)
- https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html
- https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html