

# Vehicle detection, tracking and behavior extraction (V-DTBe) - Vehicle Detection and Speed Estimation

## 1 Data set Analysis

The video that I worked on is attached in my github repository. It has been downloaded from this link : [https://gtvault-my.sharepoint.com/personal/cpps3\\_gatech\\_edu/Documents/Vehicle%20Behavior%20Monitoringslrid=61f9a59e%2Dc0d7%2D7000%2D65f6%2Ddb8cb41ca1e7&id=%2Fpersonal%2Fcpps3%5Fgatech%5Fedu%2FDocuments%2FVehicle%20Behavior%20Monitoring](https://gtvault-my.sharepoint.com/personal/cpps3_gatech_edu/Documents/Vehicle%20Behavior%20Monitoringslrid=61f9a59e%2Dc0d7%2D7000%2D65f6%2Ddb8cb41ca1e7&id=%2Fpersonal%2Fcpps3%5Fgatech%5Fedu%2FDocuments%2FVehicle%20Behavior%20Monitoring)

The downloaded zip folder is then extracted to our system. It consists of :

- checkerboard.png which is added in photoshop for the matlab code
- final.mp4 which is the for which speed is calculated
- Questionnaire.pptx which helps us to understand the challenge and it's complications
- transformation.m file which is the Matlab code provided to us for reference

The input video has two lanes , the left lane for which vehicles are coming towards us and the right lane for which, vehicles are going away from us. We have processed and calculated speed of vehicles for the left lane and not the other because a separate camera will be provided for that lane in the other direction.

## 2 Implementation Logic

1. Capture the video
2. Capture individual frames
3. Convert each individual frame to Grayscale
4. Apply Gaussian Filter to eliminate noise
5. Find difference ROI image from two consecutive frames
6. Apply Binary Thresholding and then Dilate it to increase the quality
7. Find contours
8. Eliminate contours who have area less than a certain minimum value , so as to assure only vehicles are selected and not any other unwanted things
9. Draw bounding rectangles around the selected contours
10. Find x,y,w and h of the bounding rectangle and it's centroid as well and draw a white circle on the centroid

11. Find final no of vehicles detected per frame difference
12. If the no of vehicles in the next frame is greater than the number of vehicles in the previous frame, then increase counter by one
13. Whenever a vehicle is detected, keep displaying the counter on the screen of no of vehicles detected at that particular frame only
14. Keep displaying the current date and time on the screen
15. Select tracking points from the image
16. I have selected the four corner points of the rectangle and the centroid as they are the most important ones in the tracking of the vehicle.
17. Keep adding their values to their respective arrays
18. Find corresponding displacements by the formula, displacement =  $\sqrt{(x1-x2)^2 + (y1-y2)^2}$
19. Take average of these distances to get the average distance
20. Get the average speed by dividing the average distance by time between two frames
21. Time = 1000/fps in msec which is equal to 33.33
22. Neglect distances which are greater than 20 because these are erroneous displacement vectors( they can be sudden changes while detecting different vehicles which may lead to errors, so they must be neglected)
23. Corresponding to distance of 20, speed =  $(20/(1000/30)) = 0.66$
24. Thus, neglect speed which is less than 0.66
25. Calculate mean of this speed
26. This is the average speed in image plane in pixel units
27. Convert this speed to speed in image plane in metric units by multiplying by the resolution(width\*height) in pixels and dividing by the camera pixel size which is 572 mm in general on an average for a general camera
28. Convert this speed to speed in object plane in m/msec by multiplying with the scaling factor
29. Scaling factor =  $1+(d/f)$
30. d is the distance from camera and it is selected as 10m on an average and f is the focal length of the camera which is 5.9mm on an average
31. Multiply this image by 1000 and divide by (1000\*3600) to get the speed in the object plane in km/hr

### 3 Calculations and Results

The actual speed from the vehicle was calculated by considering the following basic information of the traffic rules :

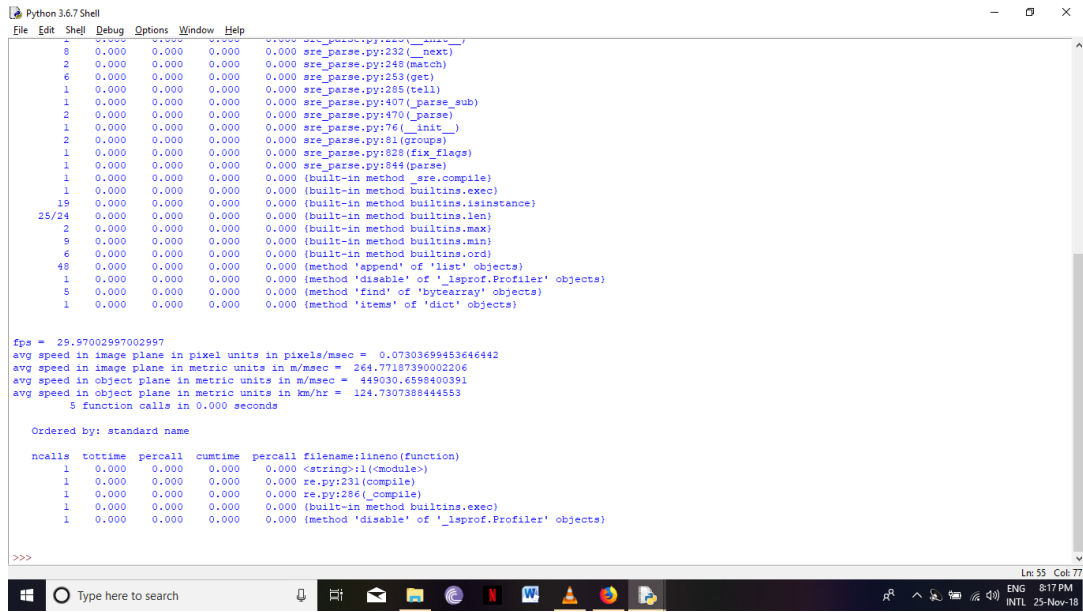
1. Each white line is 10ft long
2. Distance between two white lines is 30 ft
3. Also, width of the road is 12 ft

We considered the start and end points of a car in the camera's field of view. It covered 23 white lines in a duration of 8 sec

Thus, actual distance =  $23 \times 10 + 22 \times 30 = 890 \text{ ft} = 271.272 \text{ m}$

Thus, actual speed =  $271.272 / 8 = 33.909 \text{ m/sec} = 122.0724 \text{ km/hr}$ .

#### 3.1 Outputs obtained :



```
Python 3.6.7 Shell
File Edit Shell Debug Options Window Help
1 0.000 0.000 0.000 0.000 sre_parse.py:122(__new__)
2 0.000 0.000 0.000 0.000 sre_parse.py:232(_next)
2 0.000 0.000 0.000 0.000 sre_parse.py:248(match)
6 0.000 0.000 0.000 0.000 sre_parse.py:253(get)
1 0.000 0.000 0.000 0.000 sre_parse.py:285(tell)
1 0.000 0.000 0.000 0.000 sre_parse.py:407(_parse_sub)
2 0.000 0.000 0.000 0.000 sre_parse.py:470(_parse)
1 0.000 0.000 0.000 0.000 sre_parse.py:76(__init__)
2 0.000 0.000 0.000 0.000 sre_parse.py:81(groups)
1 0.000 0.000 0.000 0.000 sre_parse.py:828(fix_flags)
1 0.000 0.000 0.000 0.000 sre_parse.py:844(parse)
1 0.000 0.000 0.000 0.000 (built-in method sre.compile)
1 0.000 0.000 0.000 0.000 (built-in method builtins.exec)
19 0.000 0.000 0.000 0.000 (built-in method builtins.isinstance)
25/24 0.000 0.000 0.000 0.000 (built-in method builtins.len)
2 0.000 0.000 0.000 0.000 (built-in method builtins.max)
9 0.000 0.000 0.000 0.000 (built-in method builtins.min)
6 0.000 0.000 0.000 0.000 (built-in method builtins.ord)
48 0.000 0.000 0.000 0.000 (method 'append' of 'list' objects)
1 0.000 0.000 0.000 0.000 (method 'disable' of '_laprof.Profiler' objects)
5 0.000 0.000 0.000 0.000 (method 'find' of 'bytearray' objects)
1 0.000 0.000 0.000 0.000 (method 'items' of 'dict' objects)

fpe = 29.97002997002997
avg speed in image plane in pixel units in pixels/msec = 0.07303699453646442
avg speed in image plane in metric units in m/msec = 264.77187390002206
avg speed in object plane in metric units in m/msec = 449030.6598400391
avg speed in object plane in metric units in km/hr = 124.7307388444553
5 function calls in 0.000 seconds

Ordered by: standard name
ncalls  tottime  percall  cumtime  percall filename:lineno(function)
1 0.000 0.000 0.000 0.000 <string>:1(<module>)
1 0.000 0.000 0.000 0.000 re.py:231(compile)
1 0.000 0.000 0.000 0.000 re.py:286(compile)
1 0.000 0.000 0.000 0.000 (built-in method builtins.exec)
1 0.000 0.000 0.000 0.000 (method 'disable' of '_laprof.Profiler' objects)

>>>
```

Figure 1: Output showing the speed obtained in object plane for first 4 vehicles in km/hr

$$\text{Error} = (124.7307 - 122.0724) / 122.0724 = 0.02$$

Thus, 98 percentage accuracy is obtained by the code which speaks of the high working accuracy of the code.



Figure 2: Output showing a car being detected successfully

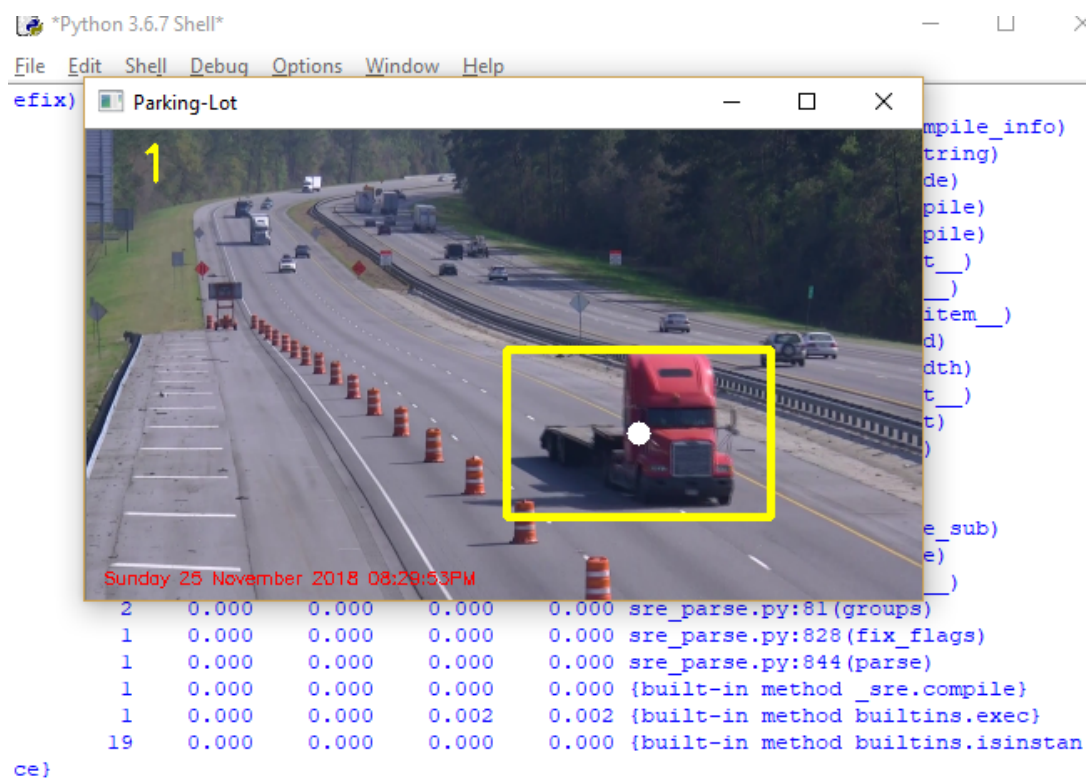


Figure 3: Output showing a heavy vehicle being detected successfully

## 4 Conclusion

Thus, vehicles are detected successfully and speed estimation is being made with an efficiency of 98 percentage. This could be further increased if a wider data set is

selected and different method like optical flow and motion history image could be used to detect the motion vectors.

The github repository link for the code is : <https://github.com/shubhamsidhwa/Vehicle-detection-tracking-behavior-extraction-V-DTBe-Vehicle-Detection-and-Speed-Estimation>

Thus, this was my research for the above task for speed estimation. I will like to work more on this task and such related tasks if given an opportunity to work as a Research Assistant in your lab. Thanks a lot for these tasks as they helped me to learn a lot !