In recent times, there has been a drastic change in people’s lifestyles and with an increase in incomes and lower cost of automobiles there is a huge increment in the number of cars on the roads which has led to traffic and commotion. The manual efforts to keep people from breaking traffic rules such as the speed limit are not enough. There is not enough police and man force available to track the traffic and vehicles on roads and check them for speed control. Hence, we require technologically advanced speed calculators installed that effectively detect cars on the road and calculate their speeds.

To implement the above idea two basic requirements, need to be met which are the effective detection of the cars on roads and their velocity measurement. For this purpose, we can use OpenCV software which uses the Haar cascade to train our machine to detect the object, in this case the car.

we have developed a Haar cascade to detect cars on the roads, whose velocities are then measured using a python script. The real-time application of this project proves to be much useful as it is easy to implement, fast to process and efficient with low cost development. Also, the tool might be useful to apply in simulation tools to measure velocities of cars. This can be further developed to identify all kinds of vehicles as well as to check anyone who breaks a traffic light.

The improvements in the project can be done by creating a bigger haar cascade since bigger the haar cascade developed, more the number of vehicles that can be detected on the roads. Better search algorithms can allow a faster search and better detection of these vehicles for better efficiency.

This paper is to develop an algorithm to calculate the speed of the object(vehicle) detected. We have implemented the algorithm using

Python Script.

The complete implementation uses two basic processes: -

1. Car detection using Haar cascades in OpenCV

2. Measurement of velocity of detected cars using python script.

**Car Detection:**

Object Location utilizing Haar highlight based course classifiers is a compelling item discovery strategy that uses a machine learning based approach where a course capacity is prepared from a considerable measure of positive and negative pictures. It is then used to recognize protests in different pictures.

• Initially, the calculation needs a considerable measure of positive (pictures of autos) and negative (pictures without autos) to prepare the classifier. At that point, we have to concentrate highlights from it. For this, haar highlights appeared in beneath picture are utilized. They are much the same as our convolutional part. Each component is a solitary esteem acquired by subtracting total of pixels under white rectangle from aggregate of pixels under dark rectangle.



Now every single conceivable size and areas of every part is utilized to ascertain a lot of components. (Simply envision what amount of calculation it needs? Indeed, even a 24x24 window comes about more than 160000

components). For each component computation, we have to discover whole of pixels under white and dark rectangles. To tackle this, they presented the necessary pictures.

• Now, we apply each component on all the preparation pictures. For each component, it finds the best limit which will characterize the countenances to positive and negative. Be that as it may, clearly, there will be blunders or misclassifications. We select the elements with least mistake rate, which implies they are the elements that best orders the auto and non-auto pictures.

• So now you take a picture. Take each 24x24 window. Apply 6000 elements to it. Check on the off chance that it is auto or not.

**Speed Calculation**

• Once a car is detected, using the cascadeClassifier() function on the haar cascade developed.

• Now the time is started which was initialized to 0.

• Using the ratio in the image for each cm travelled by the detected image and real-time distance in meters, the actual distance covered by the car is calculated.

• As soon as the car reaches the center of the detection window whose distance is already known to us the time is stopped.

• Now the actual distance calculated is divided by the time calculated and velocity is obtained.

• This velocity and the distance of the camera in feet from the car (i.e. the height of camera above the car) is printed on the output screen.

For this use multiple object detection algorithms could have been used but the algorithm of developing the Haar cascade and its implementation proves to be the best since it is the least time consuming, most efficient and highly reliable.

Execution:

python speedDetect.py

