Real time Vehicle Speed and Lane Detection

Project Group : xkcd
Mohan Kagita
Computer Science
UNSW
Sydney,Australia
z5124393@ad.unsw.edu.au

***Abstract*—Detecting the speed of the vehicle in real time environments**

I. Background

Speed detection has traditionally been achieved via directly measuring the distance from the camera to the vehicle in order to calibrate the speed of the vehicle across frame [**?**]. For speed cameras specifically, they use Doppler radar technology [**?**] will measures the changes of microwaves reflected from vehicles in order to obtain the speed.

II. Definition of Project Goals

My goal in this project have been defined as follows:

- Speed Detection: Detecting the speed of the identified vehicles (km/hr) in a video sequence.

III. Scope of Project

My contribution to the project centres on detecting speeds of the identified vehicles in real time video environment, by using trained vehicle classifier to spot vehicles and subsequently place a bounding box over any vehicles detected in the frame and then calculating the speed of the vehicle using the relative pixel co-ordinates of the vehicles detected in subsequent frames divided by the focal length relative to the vehicles position in the video.

IV. Individual Contributions

Since the project had 3 core modules, the tasks were split into three and assigned to the team as follows:

- Vehicle Identification - Surya Avinash Avala
- Speed Detection - Mohan Kagita
- Lane Detection - Kaan Apaydin

V. Problem Decomposition

The project was decomposed into the following sub-tasks:

A. Vehicle Detection

In this section, Vehicles/vehicle boundaries (primarily Cars) were identified automatically using Haar Cascade Classifier specifically trained to detect cars. This involved training and optimizing the model to be specific to vehicle classification.

B. Speed Detection

This part mainly focused on detecting the speed of the vehicle in real time by keeping track of the relative pixel positions in every single frame and identifying the changes in positions of the cars detected by using pre-trained vehicle classifier and converting them into real object parameters.To generalize the direction in which the vehicle is progressing subjecting to change in pixel values, Manhattan distance is being calculated between the centroids of the vehicles in subsequent frames and resulting pixel value is converted from image plane to object plane.

C. Unidentified Vehicle Removal

This involves finding out the vehicles that are not identified in any specific frame and keep updating the information to see if the same vehicle is being detected in subsequent frames and update the speeds accordingly to avoid any fluctuations in speeds.

VI. Hardware & Software Requirements

During the implementation of my part I have used various software packages and open source tools available.They are Scikit video for reading videos and opencv3 to perfom few video processing techniques and numpy package to use an efficient array to store the tracked values.All these packages are being used as part of python programming language.Detailed instructions on installing the mentioned packages are given in readme file of group report.

VII. Implementation

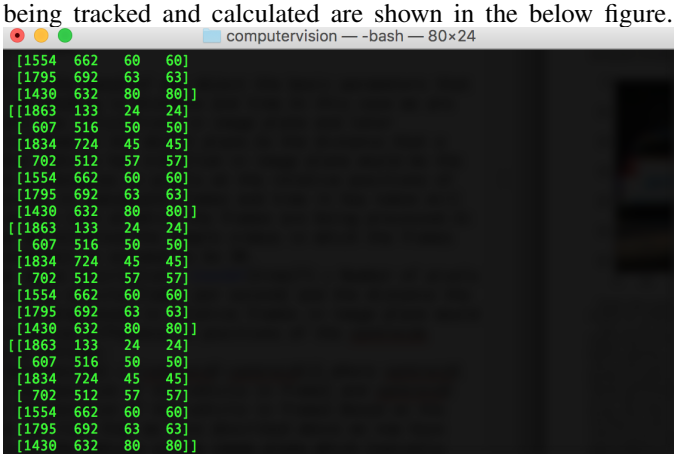
In my approach to solving the problem mentioned in the problem decomposition, traditional computer vision processing techniques and Classifiers based on feature matching algorithms are used in order to develop our prototype.

A. Design & Algorithms

Vehicles are detected using a Haar Cascade Classifier, which is being trained by with a data set from Stanford which containes about thousands of images to extract all possible features of the cars and use them to detect in real time environments

Once the vehicle is detected using the above trained Classifier, the subsequent part that needed to be solved was the

speed of the vehicle. In order to achieve this, the video was read using Scikit video package and all video frames with vehicle being detected and bounding boxes drawn around with the help of the Cascade Classifier, were output to a folder. Subsequently, the video frames were iterated over one by one and their coordinates or absolute pixel values of the positions of the detected vehicles were obtained, using the HOG Classifier in order to track the ROI of the vehicle/s present in every frame. The ROI or the coordinates of all the cars that being tracked and calculated are shown in the below figure.



Once the regions of interest of all the vehicles in the frames were obtained, the midpoints of the bounding boxes surrounding the vehicles were obtained, which is held as the ”position” of the vehicle which will be key for future frames. This process was repeated for all vehicles to create a list of midpoint values which were updated every frame.

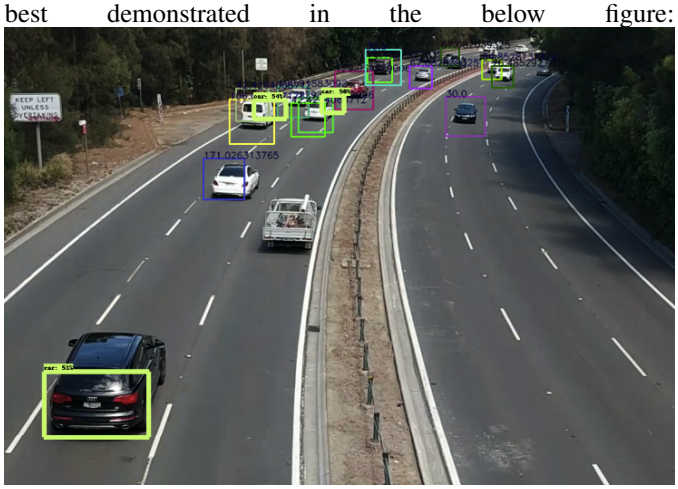
In order to find out the speed of any object the basic parameters that will be needed are distance and time. In this project, we instead are making all calculations in the image plane and then later projecting them in the object plane. In this case, the distance a particular vehicle has travelled in the image plane would be the relative differences in pixels at various vehicle positions in subsequent frames. Time would be the rate at which the frames were being processed, which we assumed to be 30 frames per second for the purposes of this project.

From the above description, **Time = Number of pixels progressed by vehicle/frames per second** and **Distance = sqrt(mid1*mid1+mid2*mid2)** where mid1 and mid2 are the positions of the vehicle in frame 1 and 2 respectively. If the difference is less than 10 pixels, then the car is assumed to be stationary and is neglected when updating values.

Once the speed of the vehicle in the image plane is calculated, we then convert the pixel values obtained into km/sec which is representative of the object plane, using the distance of the object from the camera in the given formula **exact distance to object (mm) = focal length (mm) * real height of the object (mm) * image height (pixels)/object height (pixels) * sensor height (mm)** The calculated values are displayed in every frame is shown in beloew figure.

However, the trained classifier is unable to detect all the cars in every single frame and thus the obtained speed values are not always accurate and subject to deviate from original value depending on the following assumptions on the parameters. The height of the object in object plane is assumed to be 4meters in general approximating to median values for all cars which is not the same in every case. The distance between the camera and the road is approximated which is not accurate due to practical constraints and hence approximated.

The detected cars and their speeds are best demonstrated in the below figure:



One scenario faced in this project is that the classifier was able to detect the car in one frame and then unable to detect the vehicle several subsequent frames before finally detecting the vehicle once more. This obviously led to erroneous speed measurements as the vehicle is seen travelling one position from one frame to another position in the n’th frame as a transition between two frames. This was rectified by introducing a variable check to determine whether the vehicle which discarded speed readings based on the difference between frames.

B. Implementation Issues

There were a few implementation issues when it came to developing the project. Firstly, training the Classifier with the data-sets available took quite some time which interfered in achieving deadlines for completing tasks and prolonged testing.

In order to obtain accurate speed detection readings, the focal length parameter is manually changed for each video to reflect different approximate distances between vehicles and the camera, which is obviously prone to inaccuracies.Also

the classifier is not able to detect cars in every single frame which is needed to obtain more accurate speeds in object plane.

Furthermore, Collision detection has become a huge challenge due to inefficiency of the classifier detecting cars in every single frame which made the task complicated and couldnt finish within the deadlines

VIII. Design Justifications, Testing & Results

In order to test our prototype,samples of videos were collected across different locations around Sydney to capture vehicles with varying speeds and backgrounds. The testing phase allowed us to fix few bugs in our implementation which needs improvements in efficiency of the Classifier to detect vehicles in every detected frame and work on removing false positives and subtracting stationary objects etc.. one such case where false positives occur is demonstrated in below figure.

The inefficiency of the classifier not detecting vehicles in every frame can be best demonstrated in below figure in which the car is not detected.

Drawing a comparison between predicted vs obtained values,we are successful in most of the cases except for the few cases mentioned in the Implementation issues which are limited by many real time constraints such as fps of camera,height of vehicles,focal length etc.

IX. Relating to Previous Experiences

In order to successfully complete my project,the computer vision techniques that I have learned are used in major part of the project.Supporting this to put my theoretical aspects into working prototype programming language I have learned in COMP9021 is the backbone for this project.Apart from this technical aspects all my previous experiences gained over the course of time are useful in order to communicate with my group mates and push the project with good understanding and coordination to make it a successful project

X. Conclusion

In conclusion, although there were few issues with false positives and the inability to detect cars, the prototype was able to detect speeds of most of the vehicles with reasonably high accuracy during the lab demonstration.