"INDIAN ROAD SCENE ANALYSIS FOR DRIVER ASSISTANCE"

MINI PROJECT REPORT

submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

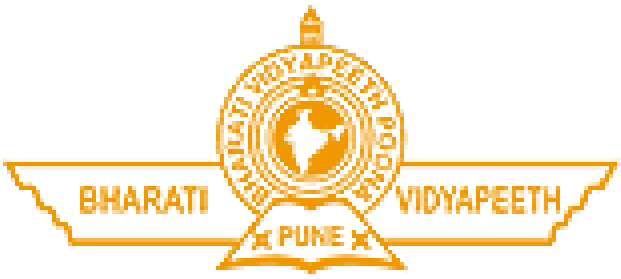
ELECTRONICS & COMMUNICATION ENGINEERING

TEAM

CHETAN CHAWLA
ISHANI JANVEJA
DIVYANSH MALHOTRA
HARSHIL BANSAL

MENTOR

MR. ABHISHEK GAGNEJA



DEPARTMENT OF ELECTRONICS COMMUNICATION ENGINEERING
BHARATI VIDYAPEETH'S COLLEGE OF ENGINEERING
(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI)
NEW DELHI - 110063

APRIL 2018

ABSTRACT

Driver assistance systems have progressed rapidly in recent years to increase road safety by assisting drivers with lane departure alerts or automatic braking during potential collisions. Developing such an assistance system for Indian roads presents a massive task owing to the prevailing chaotic traffic conditions, especially in India. Therefore, the key goal of our work is to develop a driver assistance system for which we apply road scene analysis techniques and vehicle-to-cloud communication to avoid vehicle-to-pedestrian collision and vehicle-to-vehicle collision, respectively. Our work is divided into two phases - (i) Prototyping a driver assistance system to prevent vehicle-vehicle and vehicle-pedestrian collision, and (ii) Introducing a pedestrian detection benchmark dataset consisting of videos collected on roads in New Delhi, India. While prototype-building serves as a task to understand the deceptive issues involved in building a driver assistance system, the second part aims at getting the ground ready for solving problems associated with pedestrian detection in heterogeneous scenarios.

Contents

List of Figures

Chapter 1

Introduction

1.1 Project Motivation

Road safety in India has always been a matter of concern especially in the recent years after Brasilia Declaration on Road Safety,as a signatory to which India resolved to reduce the number of road accidents and fatalities by 50% by 2020 [1]. Statistics as mentioned in the report "Road Accidents in India 2015" by Shri Nitin Gadkari suggest that over 200,000 people in India lost their lives in road accidents in 2015. The report also states that 49% of accidents occur at intersection points. A major number of fatalities are caused due to over speeding vehicles hitting pedestrians and cyclists.

After an analysis of the prevailing situation, we concluded that a smart system capable of warning the drivers prior to predicted potential collisions or alerting a distracted driver about pedestrian in front of the vehicle could save nearly 100,000 lives every year.

With an aim to improve road safety and to help India realize the vision of bringing down the count of accidents to half, we designed a framework for a collaborative driver assistance system that takes camera feed and GPS sensor data as its inputs to assist the drivers. Since the prototype developed uses native computer vision algorithms for pedestrian detection, the accuracies obtained were not suitable for a driver assistance system.

This drawback not only motivated us to work on deep learning algorithms for object detection but also led us to discover the lack of a benchmark dataset for Indian road scenarios. Therefore, we present a pedestrian detection dataset which includes hours of video recording shot on the roads of New Delhi, India using a front-facing camera mounted in a vehicle. In order to annotate the data, we also build a customized video annotation tool (labelVDOS) to suit our future requirements for building a multi-class, multi-object tracking dataset.

1.2 Overview

PHASE-I

With the primary aim to tackle vehicle-to-vehicle and vehicle-to-pedestrian collisions, we designed the framework of a collaborative driver assistance system "DRIZY:DRIve eaSY". The application of this system can be broadened to solve more road safety issues using the same basic architecture which consists of two standalone modules :

1. **DRIZY - VISION** : Responsible for handling all computer vision related processes. This module has currently been prototyped using the Raspberry Pi 3 Model B to alert driver of impending collisions with pedestrians.

2. **DRIZY - COM** : Establishes communication between vehicles and cloud. In order to prevent vehicle collisions at intersections, we have prototyped this module to predict and alert vehicles in collision trajectory. This module has been implemented using a smart-phone application

PHASE-II

The next phase of our project focused on laying the ground work for improving accuracies of the pedestrian detection algorithm and for solving the problem of pedestrian trajectory estimation. Both these tasks required deep learning models to be trained and tuned specifically for the chaotic Indian road scenarios. However, since the existent dataset like KITTI[2], Caltech[3] and Inria have all been recorded on western roads, they do not contain the unique problems associated with Indian road scenarios. Therefore, we took up the challenge to introduce a pedestrian detection dataset prepared on the roads of New Delhi, India.

The preparation of the dataset required hours of manual labour to annotate pedestrians in each frame of collected videos. Therefore, to simplify the task, we built a toolbox (labelVDOS) specifically designed to deal with video annotations for multi-class, multi-object tracking. The details of both the the toolbox and the dataset have been enumerate in this report.

Chapter 2

Driver Assistance System

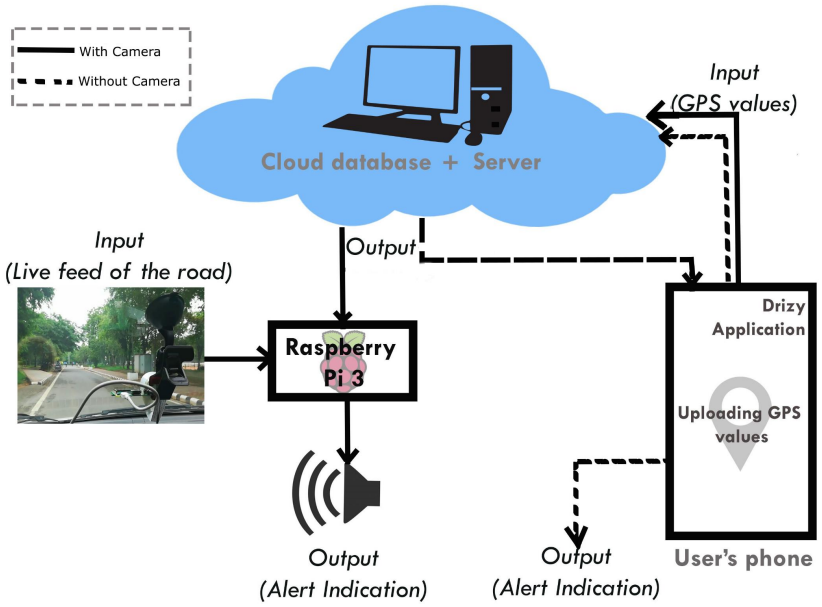


Figure 2.1: System Architecture with and without camera

The system consists of two modules - DRIZY-VISION and DRIZY-COM.

2.1 Drizy - VISION

Drizy Vision is responsible for all driver assistance features using computer vision algorithms.

Components

1. **Raspberry Pi 3 Model B**
   This embedded platform is responsible for executing computer vision algorithms that obtain input frames from camera feed to detect and warn drivers of obstacles ahead. Specifications

   - CPU: 4x ARM Cortex-A53, 1.2GHz
   - GPU: Broadcom VideoCore IV

- RAM: 1GB LPDDR2 (900 MHz)

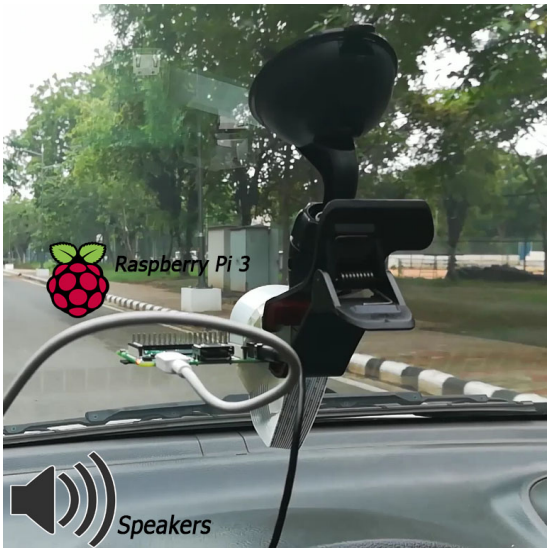


Figure 2.2: Raspberry Pi and camera mounted in the car during test drives

2. **Raspberry Pi Camera V2**
   We used Raspberry Pi Camera V2 which is an 8MP Sony IMX219 image sensor featuring a fixed focus lens is capable of 3280 x 2464 pixel static images, and also supports 1080p at 30fps, 720p at 60fps, and 640x480p at 90fps video. The Camera Serial Interface(CSI) connects the camera directly to the GPU while the CPU remains available for processing the compute-intensive computer vision algorithms.

3. **OpenCV + Python**
   OpenCV is an open source library of programming functions designed for computational efficiency and with a strong focus on real-time applications. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. We use OpenCV with python bindings to code computer vision algorithms.

4. **Buzzer**
   We make use of a 5V piezoelectric buzzer connected to pi's GPIO pins to sound alerts. Vehicle's speakers connected to pi's the audio jack can also be used for audio outputs.

The module has been deployed using Raspberry Pi 3 Model B and Raspberry Pi Camera Module V2. This not only makes a cost effective solution but also speeds up the acquisition of frames from the camera due to CSI connectivity to GPU directly. Drizy Vision has been prototyped for pedestrian-to-vehicle collision avoidance using HOG+SVM [4] for pedestrian detection that has been implemented using OpenCV and Python2.

The HOG (**H**istogram of **O**riented **G**radients) person detector uses a sliding detection window, 64 pixels wide by 128 pixels tall, which is moved around the image at a predefined step-size. At each position of the detector window, a HOG descriptor is computed. This descriptor is then shown to the trained SVM (**S**upport **V**ector **M**achine), which classifies it as either a pedestrian or not a pedestrian.

The state of art object detection techniques are based on deep learning algorithms like faster R-CNN[5](**R**egions with **C**onvolutional **N**eural **N**etworks) and its derivatives like YOLO[6]. However, the real challenge involves running them real-time on embedded platforms. Therefore, as an acceptable trade-off of 0.8 and 9 (Figure 2.5) between precision and frames processed per second, respectively, we chose HOG+SVM based pedestrian detector. It has been found that an FPPI (false positives per image) value of 0.5 is acceptable for such driver assistance systems. Figure 2.5 shows that our HOG+SVM pedestrian detector lies well below this threshold at fairly high recall rates.

Figure 2.3: Selecting Region of Interest from the Camera view prevents algorithm from detecting all the pedestrians, hence increasing its speed. (b) Detecting pedestrians in ROI using HOG + Opencv

The sliding window algorithm employed for pedestrian detection using HOG+SVM further requires optimisation for bring the fps count to match real-time. Therefore, we reduce the number of sliding windows by selecting region of interest based on known geometric constraints. We achieve further improvement in fps by increasing the step-size of the detection window from 4 pixels to 8 pixels in both x and y direction with nearly no compromise in accuracy. Thus, the processing time for each frame is reduced by a factor of 7x (from 1.4 fps to 9.8 fps). Next, we use multi-threading to parallelize the capture of recent camera frames and pedestrian detection for multiple frames. With the optimised detection techniques, Drizy - Vision is capable of detecting pedestrians within the range of 20 metres.

2.2 Drizy - COM

Drizy-com uses vehicle-to-vehicle communication over network and GPS based localization to provide real time alerts for any V2V collision at an accident blackspot.

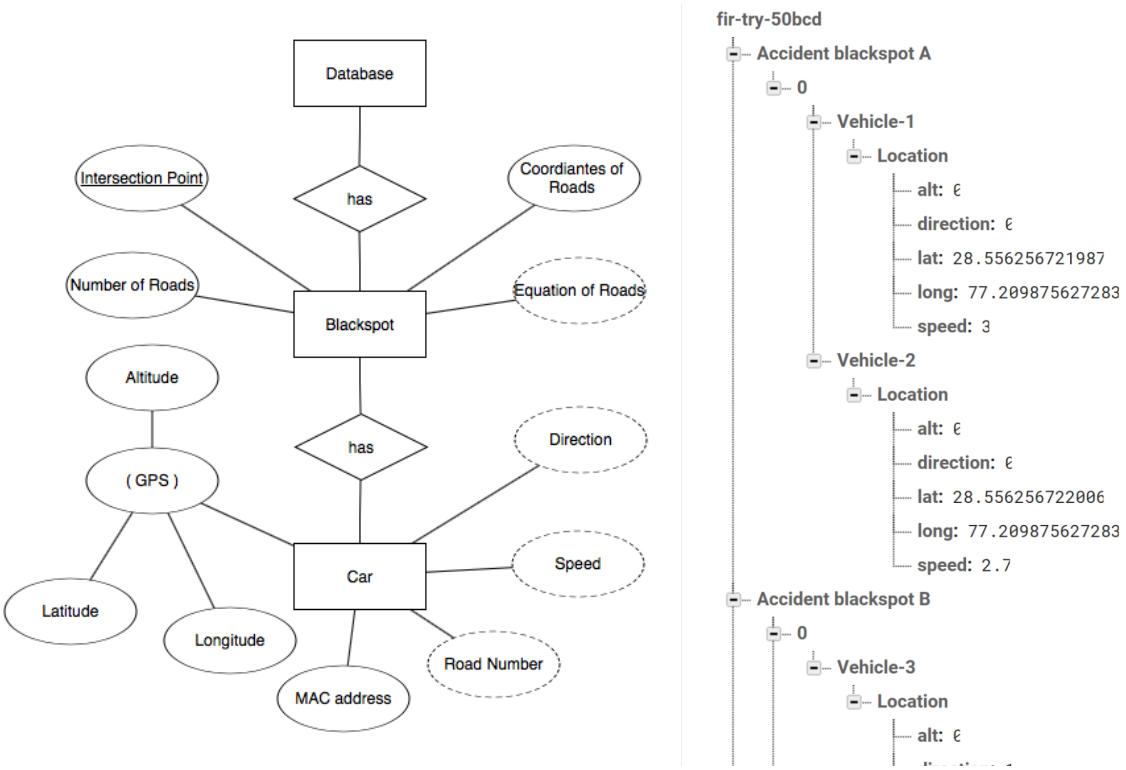


Figure 2.4: Entity relationship diagram of Drizy-Com Database and its firebase implementation

Components

1. **Smartphone Application**
   The application uses GPS sensors of the smartphone to calculate latitude, longitude, altitude and speed of vehicle. Using the latitude, longitude and altitude, it also determines the accident blackspot and road number associated with the blackspot on which vehicle is travelling. This is done by comparing dynamic GPS data of vehicle with a pre-defined electronic map of accident prone blackspots known as electronic horizon. The smartphone apllication is also responsible for sounding alerts via smartphone speakers.

2. **Cloud Database**
   The database is maintained in a tree structure in which the parent nodes represent accident blackspots, child nodes represent road numbers and end points represent vehicles and their parameters(GPS coordinates, direction of navigation and speed). The cloud database used here is Google Firebase for its real

time applications.

3. **Cloud Server**
   The cloud server processes data stored in the cloud database to generate assistance warnings when drivers surpass unacceptable risk thresholds. It is responsible for running all the algorithms efficiently with minimum latency. Drizy-Com is currently protoyped on AWS EC2 UBUNTU 16.04 free tier instance.

Drizy - COM has been prototyped for vehicle-to-vehicle collision avoidance alerts at 2 accident blackspots at IIT Delhi. The driver is required to turn on the Drizy smartphone application and turn on the volume. The app starts gathering GPS values from the satellites and infers its attribute like blackspot number, road number etc. The estimate of a potential collision is made on the cloud using a linear predictive algorithm.

Each vehicle entering a specific accident blackspot uploads its GPS derived attributes via the app to the cloud. The cloud maintains a hierarchical structure of the vehicles and their attributes, as shown in Figure 2.4. These attributes include speed and direction of navigation(away or toward the accident blackspot) besides the vehicle's latitude, longitude and altitude. The speed and direction of navigation are determined by studying the change in GPS coordinates of the vehicle. The cloud server refers the cloud database to predict potential collisions in a blackspot and alerts the vehicles that could be on probable collision trajectory. The alerts are sounded on the smartphone itself.

Algorithm 1 discusses how the server predicts impending collisions and calculates time for each vehicle to reach the intersection point. If the difference in estimated time of any two vehicles to approach the intersection is less than a threshold set to 5 secs, then the cloud sends an alert to vehicles of interest. The alerts are received on the application itself and preset alert tones are generated.

The algorithm adapts its threshold according to the speed of the vehicle. For example, if the vehicle is travelling too fast, then the threshold is doubled to 10 secs, so that driver has sufficient time to apply brakes. This reduces the frequency of alerts thus avoiding redundant or false alerts.

We used Google Firebase for real-time cloud database maintenance and AWS EC2 UBUNTU 16.04 free tier instance for cloud computing.

Algorithm 1 Server Code Algorithm

```
procedure SERVER CODE ALGORITHM(.)
    For each pair of vehicles in Accident Black Spot
    Get vehicle's GPS, speed, road number from Cloud
    if road1 != road2 then
        if direction of both vehicles is toward intersection then
            Find the distance of two vehicles from intersection using Haversine formula [7].
            Predict times t1 and t2 to reach the intersection using linear predictive model
            if |t1 − t2| < threshold then
                Send collision alert to vehicles.
            end if
        end if
    end if
end procedure
```

2.3 Evaluation

2.3.1 DRIZY - VISION

This section addresses the question of how well Drizy can detect pedestrians keeping sufficient reaction time for the driver.

Accuracy

Accuracy of a pedestrian detector is generally determined on a frame-by-frame basis. However, Drizy-VISION, a warning system installed in a car, is not affected if all the pedestrians are detected or correctly localized in the frame, instead it validate if an alert is generated when pedestrians walk in front of the car or not.

Using this method, various parameters were evaluated on a video dataset collected on Indian roads with 5500 positives(pedestrians) and 4500 negatives (non-pedestrian/background). All the algorithms were benchmarked on Raspberry Pi 3. Figure 2.5 shows that state of the art techniques like faster RCNN and YOLO offer high precision rates (Equation 2.1) of 98 to 99 percent with an fps of 0.8 to 1 fps. HOG+SVM classifier (proposed system) provides a speedup 10x over deep learning at a precision of 80 percent.

Figure 2 shows that HOG+SVM classifier performs better over HAAR with a gain of 0.4 percent recall (Equation 2.2) at 0.1 FPPI (false positives per image).

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.1}$$

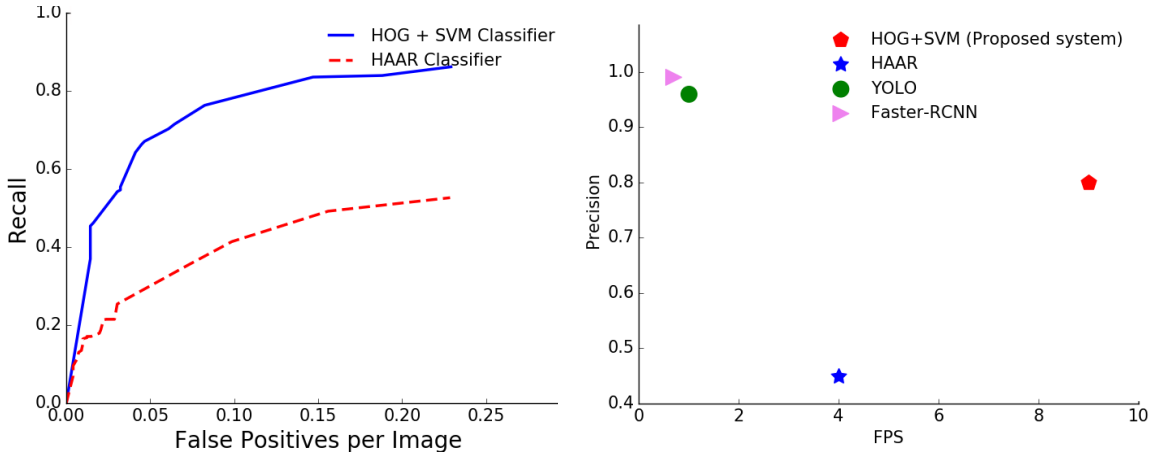


Figure 2.5: Left: Recall vs FPPI for pedestrian classifiers at varying thresholds. Right: Precision vs frames processed per second for various detection techniques.

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{2.2}$$

Reaction Time

An alert system should not only detect pedestrians accurately, but also give sufficient to the driver to apply brakes. Here we calculate average reaction time available to the driver from the time of alert to the time of collision with pedestrian.

The pedestrian detection module is capable of successful detection upto a range of 18 to 23m at varied lighting conditions.A set of experiment was carried out at IIT Delhi campus roads with 50 test drives in a car. The car was maintained at constant speed of 20, 30, 40 and 50km/hr for 10 test drives each. For every drive the time taken to reach the pedestrian after alert reception by Drizy-VISION was noted down. Figure this this this shows that the system warns the driver 4-5 sec prior to approaching a pedestrian ahead at 10-40 km/hr speed limit.

2.3.2 Drizy - COM

This section evaluates Drizy-COM on the basis of the following parameters:
(i) available time for driver to apply brake before collision.
(ii) frequency of alerts generated by DRIZY. We compare DRIZY with an "Always On" system that gives alerts to every vehicle in the accident blackspot.
(iii) accuracy of localization of vehicles.

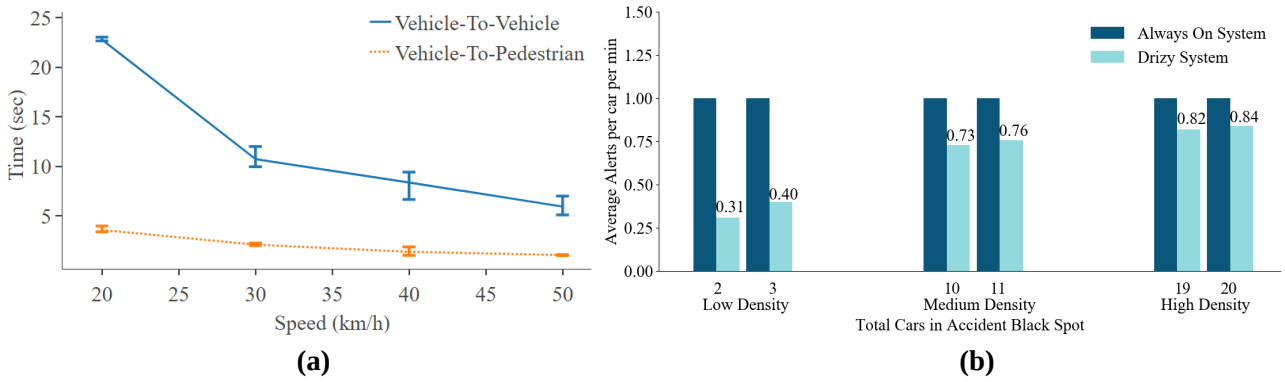


Figure 2.6: (a) Available Reaction time before collision with pedestrians and vehicles. It shows maximum and minimum values of time obtained during test-runs. (b) Comparative analysis of a system that gives alerts to every vehicle in the accident prone area with DRIZY that predicts collisions before giving alerts. The values were obtained using simulation and spawning of vehicles with random attributes in an accident blackspot.

Reaction Time

We simulate the same experiment as in Drizy-VISION to analyze reaction time for Drizy-COM at road intersections. We take 50 test drives with 2 vehicles approaching a road intersection from two different roads at varying speed. We analyze the available reaction time for the driver after the collision alert and collision prediction behaviour of the DRIZY app. Figure 2.6 (a) shows that for speeds limits between 20-25 kmph the system generates alerts with available reaction time >20s and for speed limits reaching 40-45 kmph, it is found to be close to 8s.

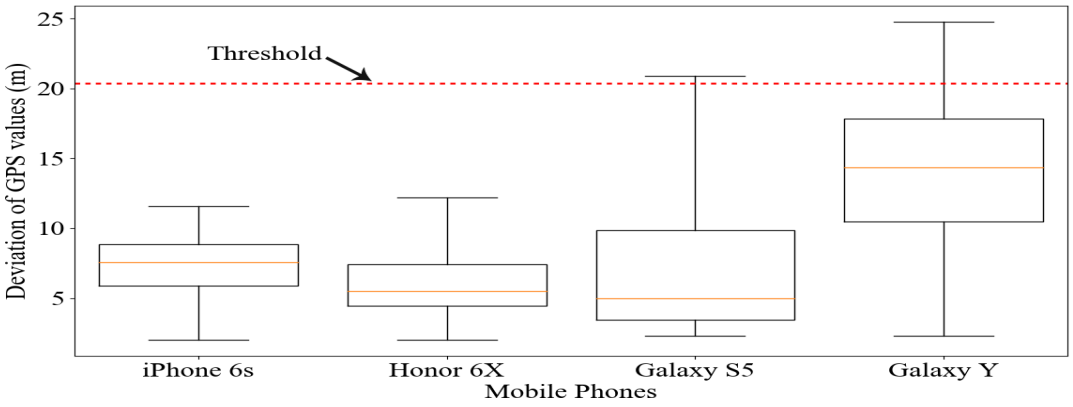


Figure 2.7: Box plot showing deviation in GPS vaues for different mobile phones

Comparison with "Always On System"

In order to study the frequency of collision alerts in different traffic densities, we simulate an experiment. We generate synthetic data with uniform distribution for 'N'vehicles which includes vehicle distance from intersection ranging between 0-400m, road number from 0-4, vehicle speed in range 1-36kmph and direction of motion of vehicle. This data is then accessed by the server to predict collisions. Figure

2.6 (b) shows the comparison of DRIZY with an Always On system at low, medium and high density traffic levels. The proposed system reduces the redundant warnings by 80%, 50% and 10% in low, medium and high traffic density, respectively.

Network Attributes

Since the assistance feature is dependent on the network speed, we show the accuracy of GPS data and latencies associated with upload, download and server processing over 4G and WiFi networks. Figure 2.7 shows a box plot of GPS errors experimented on different mobile phones. The threshold line shows the maximum tolerable limit in deviation that does not interfere in assistance system. It can be observed that mobile phones launched after 2014 are within the tolerable limit.

Table 1 shows the latencies of both 4G and WiFi netwoks for different processes. It can be observed that the throughput of the system is not dependent on the type of network or bandwidth of network, it just requires a network connectivity. This behaviour of Drizy-COM is due to the small packet size of the vehicle data that is being uploaded.

	Time (milliseconds)	
Process	**4G**	**Wifi**
Upload	258	229
Download	746	678
Server processing	30	30

Table 2.1: Average latency of Drizy-COM for different types of networks.

Chapter 3

Creating A Pedestrian Detection Video Dataset On Indian Roads

3.1 Introduction

In order to improve the pedestrian detection accuracies as obtained in the Drizy-VISION prototype, we realise that porting the system to use deep learning algorithms is a necessity since these algorithms stand at the forefront when it comes to object detection. With deep learning algorithms becoming a mainstay for vision problems, the need of advanced datasets is a necessity since it is a known fact that deep learning models are data hungry. This means that training these models for pedestrian detection requires a huge pedestrian detection dataset which includes hours annotated video footage of the road scene. All the conventionally available datasets for pedestrian detection (eg. Caltech), however, are recorded in western countries where traffic and pedestrian behaviour is much less chaotic in comparison to countries like India. Hence, we present a pedestrian detection dataset prepared on the roads of New Delhi, India.

Further, since creating such a large scale dataset requires hours of manual labour to annotate thousands of frames, we also propose a video annotation tool to minimize the effort required in creating annotations for multi-class, multi-object tracking in chaotic, object dense, occlusion heavy videos. Although the pedestrian dataset does not make use of the full capabilities of this powerful tool, the features that allow creation of a multi-class object tracking dataset are necessary from the perspective of the future extension of our work and are also discussed in detail in the following section.

3.2 Dataset Collection

As we discussed above that a deep learning algorithm works as good as the data it is fed to train upon. Hence, we propose to use an Indian dataset for training a deep learning model for pedestrian detection to improve the accuracies associated with

Figure 3.1: WheelWitness mounted Dash Cam

Drizy-VISION module. The main need of dedicated Indian dataset arises due to these reasons:

- High possibility of partial or full occlusion due to heavy traffic
- Random movement of Pedestrians across the roads
- No separate lanes of pedestrians i.e. cyclists, pedestrians, motorcyclists, cars, heavy vehicles, all run on same roads.
- Majority of roads are one-way street and thus pedestrians comes from both the sides.

3.2.1 Hardware Used

The dataset was collected by mounting a WheelWitness HD Pro Dash Cam which has the following features:

- Maximum Video Resolution: Super HD 1296P 2304 X 1296 30 FPS
- WDR (Wide Dynamic Range) for best night vision
- 170 Degress, Extra Wide Angle Lens

It has a Ambarella A7 Processor which allows us to record data on the SD card mounted with recording details and real-time camera feed displayed on the 3.0” TFT Display. The recording starts as soon as the camera is plugged in to the 12V standard Cigar/Charger point in any car and allows the user to change any setting using the provided push buttons on it.

3.2.2 Setup

We collected approximately 10 hours of video at 25 fps and resolution 1280 x 720 taken from WheelWitness HD PRO Dash Cam using Ford ECO Sports Car, driving through regular traic in urban and semi urban streets and roads of New Delhi in dierent lightning conditions over daily commune. The camera was setup on the rear-view mirror of the car giving a central perspective of the road from the vehicle.

3.2.3 Recording Procedure

The car was driven normally throughout the recordings. Different drivers were appointed to prevent any driving style bias that might be added otherwise. The video was captured across Paschim Vihar, Rajouri Garden, Punjabi Bagh, Rohini, Noida, driving through DND flyway, Gurugram, Dwarka, Rohtak and some local areas of South Delhi. The video Resolution was of 1280 x 720 pixels which was cropped to 1280 x 300p to remove the sky and lower part of the car (dash-board), to avoid unnecessary computation. The unstabilized instances which occured due to unstable roads were discarded.

3.2.4 Annotation Procedure

The dataset was annotated using the tool LabelVDOS discussed later in the document. Inspite of the extended abilities of the Video Annotation tool, we have used only single class labelling of pedestrians for detection purposes, including the occlusion label. This setting compares to that of Caltech Dataset and KITTI Benchmark Suite, which some of the most comprehensive dataset for pedestran detection.

A total of 12539 frames (approx 80 minutes) of data annotation has been completed. In these frames approximately 27000 labelled Bounding Boxes(BBs) of pedestrians were done. Each bounding box(BB) has given two types of labels, pedestrian(23786 BBs) or pedestriano(3182 BBs) denoting unoccluded pedestrians or occluded pedestrians respectively. Since our dataset is still evolving and we are benchmarking it, Indian dataset is not publicly available till date.

We split Indian dataset into 3 parts, one training and two testing dataset in the ratio of approximately 70%, 13% and 17%. Two testing dataset are created to comment on effect of illumination on results as Test2 dataset have better illumination than Test1.

Table 3.1 Summarizes the whole dataset
Table 3.2 Specifies the comparison between Caltech dataset, KITTI Benchmark

Suite Dataset and Indian (collected dataset).

Table 3.1 - Dataset Specification

No.	Training	Test1	Test2	Total
Frames	8781	1551	2027	12359
Non empty Frames	8164	1401	1927	11492
BBs of Pedestrians	19343	3449	4176	26968
Unoccluded	17065	2964	3757	23786
Unoccluded pedestrian per frame	2.09	2.12	1.95	2.07
Unoccluded pedestrian per second	52.26	52.89	48.74	51.74

Table 3.2 - Dataset Comparison

No.	Caltech	KITTI	Our
Frames	61439	7481	12539
Non empty Frames	61439	1779	11492
BBs of Pedestrians	153234	4487	26968
Unoccluded	93613	4487	23786
Unoccluded pedestrian per frame	1.52	Not video	2.07
Unoccluded pedestrian per second	38.0	Not video	51.74

3.3 labelVDOS : Video Annotation Toolbox

When we look at the existing open source video annotation tools such as ViTBaT [8], VATIC [9], ViPER [10] and iVAT[11], we find that none of them is suitable due to the complexity and effort in usage, and the high chances of error. It is with this motivation that we create our own tool to label Very Dense Object Sequences (aptly named labelVDOS) with the following novel capabilities:

- The ability to leverage interpolation techniques to only have to correct the bounding box for an object once every few frames, rather than draw one for every frame
- The ability to simultaneously keep track of multiple objects without mixing IDs even in dense scenarios
- The ability to annotate frame-level phenomenon, making it useful even for tasks such as activity recognition
- The ability to work on any platform using just Python

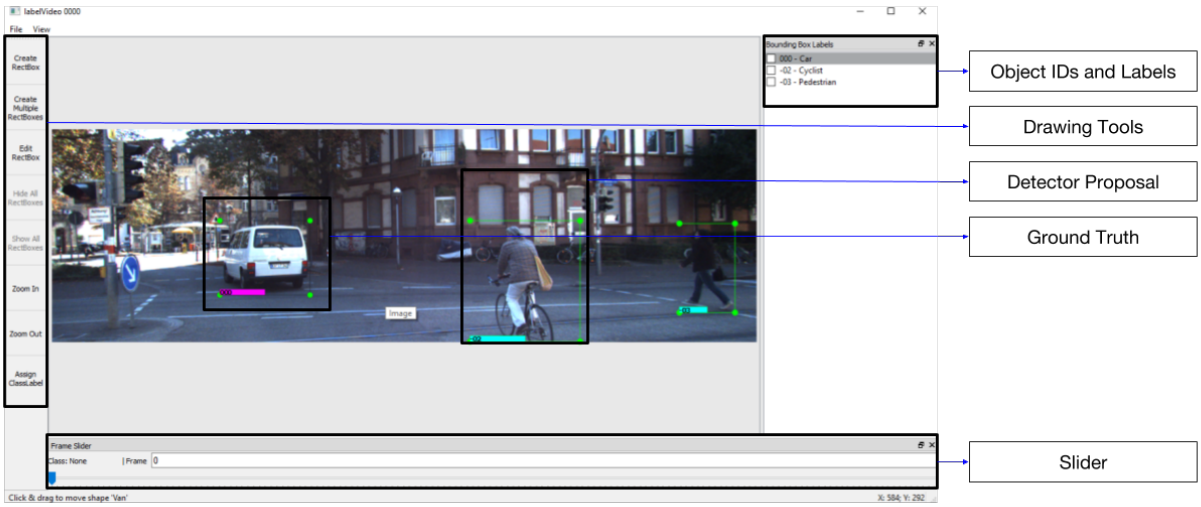


Figure 3.2: The User Interface of Our Proposed Tool

The proposed tool, labelVDOS, extends the image annotation tool labelImg[12] to the particular use-case of creating annotations for object detection as well as tracking in videos. Using an interpolation algorithm, is preferred over an object tracker due to the immense likelihood of ID switches in object dense scenarios which can increase the chances of annotation error. The overall user interface is presented in Figure 3.2 and we now discuss the challenge each component of the tool attempts to solve and its implementation.

3.3.1 Object Propagation

In order to ensure that the right-IDs are assigned to the right objects across all frames, we design a Propagation Engine. This Engine ensures that the annotator is able to simultaneously keep track of multiple objects of a wide variety of classes without mixing IDs, even in scenarios that have a high object density. This is accomplished is by ensuring that the bounding boxes that are created on a frame, either by drawing or by accepting proposals, show up on all further frames until the user designates that the object has exited the video.

The likely location of the bounding box in all future frames is computed by the Propagation Engine using a constant velocity motion model. This allows for users to merely have to correct the positions of bounding boxes in future frames rather than to have to redraw and re-assign their IDs on each frame.

3.3.2 Object Interpolation

The next challenge that increases the effort of annotators is marking the correct location for all objects in each frame. In order to work around this issue, we design an Interpolation Engine that takes in the bounding boxes of an object in two frames and

interpolates all intermediate bounding boxes. We currently use linear interpolation for its simplicity.

Suppose that $bbox_A$ defines the bounding box in frame A, $bbox_B$ in frame B ($A < B$) and let a bounding box be represented as:

$$bbox = [x_{min}, y_{min}, x_{max}, y_{max}] \tag{3.1}$$

Then, for a frame t ($A < t < B$), we can interpolate the bounding boxes using the following equation:

$$bbox_t = \frac{t}{B-A-1} * (bbox_B - bbox_A) + bbox_A \tag{3.2}$$

In this way, we enable annotators to have to only correct the bounding boxes generated by the Propagation Engine once every few frames, rather than to have to do so on each and every frame.

3.3.3 User Interface

Our tool was designed in Python, focusing primarily on creating a simple interface that would allow users to draw bounding boxes and to have to assign an ID to an object only once. In this way, it would not be difficult for users to keep track of multiple IDs in dense scenarios.

We keep the UI to be minimalistic and intuitive, in an attempt to minimize the time it takes for the user to create and annotate a bounding box. Moreover, in order to minimize the chances of error involved in marking labels, we allow for the dataset creators to predefine the classes and have the annotators select from a fixed menu of choices. Some of the other key features, apart from the intelligent components like the Propagation and Interpolation Engines include:

- The ability to show and hide individual bounding boxes easily, preventing the canvas from becoming especially cluttered
- The ability to zoom in and out, which ensures that all relevant objects can be easily selected irrespective of their size
- The ability to create multiple bounding boxes without having to click on the draw tool repeatedly

Although our tool has been designed for object tracking in object dense scenarios, it can also be extended to behavior level annotations, as well as handle the frame level annotations required in creating datasets for activity recognition and video captioning.

Conclusion and Future Scope

We started this project with the aim to solve the social problem of road safety in India. It took us no time to realize that it's a long way down the road for driver assistance systems here. To fill these gaps fast, different companies must join hands to create a collaborative system just like it happened a decade ago to create maps. We contributed to this field by implementing Drizy, which is based on the concept of connected cars and computer vision.

In order to achieve desired accuracies in detecting pedestrians in video input from a dashboard camera as in Drizy-VISION, we took up the challenge of compiling a Pedestrian Detection Video Dataset recorded on Indian roads to train deep learning models. Realising that the existent video annotation toolboxes did not fill in our needs, we also launched our very own customised toolbox - labelVDOS.

The future prospects of our project include determining and improving the accuracies of existent object detection models like Faster RCNN by fine tuning them on the Indian Pedestrian Detection Dataset. We also wish to extend our work for Pedestrian Trajectory Estimation.

References

[1] "Shri Nitin Gadkari releases the report “Road Accidents in India 2015”: reiterates commitment to reduce the number of road accidents and fatalities by 50 per cent by 2020", Pib.nic.in, 2017, url = `http://pib.nic.in/newsite/PrintRelease.aspx?relid=146093`.

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[3] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):743–761, 2012.

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR 2005*.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS 2015*.

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR 2016*.

[7] C Carl Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

[8] Tewodros A Biresaw, Tahir Nawaz, James Ferryman, and Anthony I Dell. Vitbat: Video tracking and behavior annotation tool. In *Advanced Video and Signal Based Surveillance (AVSS), 2016 13th IEEE International Conference on*, pages 295–301. IEEE, 2016.

[9] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.

[10] David Doermann and David Mihalcik. Tools and techniques for video performance evaluation. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 167–170. IEEE, 2000.

[11] Simone Bianco, Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. An interactive tool for manual, semi-automatic and automatic video annotation. *Computer Vision and Image Understanding*, 131:88–99, 2015.

[12] Tzutalin. Labelimg, 2015.