Vehicular Trajectory Detection and Traffic Analysis

Project report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Bachelor of Technology

in

Electronics and Electrical Communication Engineering

by

Sreyan Biswas

(19EC39036)

Under the supervision of

Assistant Professor Adway Mitra

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology Kharagpur

Spring Semester, 2022-23

April 28, 2023

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: April 28, 2023 (Sreyan Biswas)
Place: Kharagpur (19EC39036)

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

KHARAGPUR - 721302, INDIA

CERTIFICATE

This is to certify that the project report entitled "**Vehicular Trajectory Detection and Traffic Analysis** " submitted by **Sreyan Biswas** (Roll No. 19EC39036) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Bachelor of Technology in Electronics and Electrical Communication Engineering is a record of bona fide work carried out by him under my supervision and guidance during Spring Semester, 2022-23.

Assistant Professor Adway Mitra
Department of Electronics and Electrical
Communication Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: April 28, 2023

Place: Kharagpur

Abstract

Name of the student: **Sreyan Biswas** Roll No: **19EC39036**

Degree for which submitted: **Bachelor of Technology**

Department: **Department of Electronics and Electrical Communication Engineering**

Thesis title: **Vehicular Trajectory Detection and Traffic Analysis**

Thesis supervisor: **Assistant Professor Adway Mitra**

Month and year of thesis submission: **April 28, 2023**

In the realm of highway management, intelligent vehicle detection and counting are becoming more and more crucial. However, because vehicles come in a variety of sizes, it might be difficult to detect them, which has an impact on how accurately counts of vehicles are made. To address this issue, this work looks at a vision-based vehicle detection and velocity estimation methodology. Several highway surveillance videos based on different scenes are used to verify the proposed methods.

The use of deep Convolutional Neural Networks (CNNs) has achieved amazing success in the field of vehicle object detection. CNNs have a strong ability to learn image features and can perform multiple related tasks, such as classification and bounding box regression. The experimental results show that using deep learning methods can provide higher detection accuracy with multiple objects, especially for the trajectory tracking of the detected vehicles.

Acknowledgements

I place on record, my sincere thanks to Assitant Prof. Adway Mitra, supervisor for this project, for his efficacious advise, perpetual and prolific encouragement and constructive criticism during the course of the work and in the preparation of this project report. I would like to express my sincere gratitude to all the staff of this department for their help rendered in completion of this project-work. Last but not the least I would like to extend my gratitude to all the professors in my department who had been a teacher in the courses that I have attended during the pursual of my academic degree.

Contents

Chapter 1

Introduction

For the development of Advanced Driver Assistance Systems and traffic studies, a realistic data base is essential. Traffic cameras at metropolitan crossroads offer a mechanism to effectively gather data regarding complex traffic behaviour, with road users' trajectories being especially important.

Over the past decade, there has been a significant growth in the use of video surveillance equipment and an increase in processing power, which has made it possible to develop more advanced systems for vehicle tracking. These systems are important for a variety of applications, including automatic video monitoring, traffic control, and abnormal event detection.

1.1 Motivation

The following factors are the project's main drivers:

- **Traffic monitoring**: Traffic video monitoring can provide real-time information about traffic flow, congestion, and accidents. This information can be used by traffic management centers to adjust traffic signals, reroute traffic, or dispatch emergency services.
- **Vehicle speed enforcement**: Traffic video monitoring can be used to enforce speed limits on roads and highways. By estimating the speed of vehicles using

video footage, authorities can identify drivers who are exceeding the speed limit and issue fines or other penalties.

- **Safety analysis**: Video monitoring can also be used to analyze traffic patterns and identify potential safety risks. By studying video footage of intersections or highways, researchers can identify dangerous driving behaviors, such as running red lights or making unsafe lane changes.
- **Autonomous vehicles**: Video monitoring and vehicle speed estimation are also important for the development of autonomous vehicles. These technologies can be used to detect and track other vehicles on the road, estimate their speed and trajectory, and help the autonomous vehicle navigate through traffic safely.
- **Traffic research**: Traffic video monitoring and vehicle speed estimation can be used for traffic research purposes, such as studying traffic patterns and identifying areas of congestion. This information can be used to design better road networks and transportation systems.
- **Advanced Driver Assistance Systems (ADAS)** rely on accurate and reliable data to function effectively. These systems, which are becoming increasingly common in modern vehicles, use a combination of sensors, cameras, and other technologies to monitor the vehicle's surroundings and provide real-time feedback to the driver or even take control of the vehicle in certain situations. In order to develop and test these systems, it is essential to have access to a realistic data basis that accurately reflects the behavior of real-world traffic. One of the most efficient ways to obtain this data is through the use of traffic cameras located at urban intersections. These cameras capture video footage of vehicles, cyclists, and pedestrians as they move through the intersection, providing detailed information on their trajectories, speeds, and behaviors.

Chapter 2

Related Work

2.1 Literature Survey

There are many works that try to address the problem of vehicle speed estimation using machine vision. For instance, Tang et al. introduced a methodology that consists in the calibration of a pinhole camera model based on vanishing points, followed by object detection using YOLO9000 as discussed in [11] .

In the field of intelligent transportation systems, a lot of research has been conducted on the tracking of vehicles for traffic analysis. To derive vehicle trajectories, many sensors and tools have been investigated such as LIDAR,GPS, Computer Vision.

2.1.1 LIDAR and GPS based methods

LIDAR sensors are very convenient as they provide depth information, allowing to easily obtain accurate ground position of the vehicles. The method presented in [7] relies on a network of single-row lidar to extract vehicle trajectories. However, LIDAR sensors are highly expensive and require an extensive calibration procedure, which limits their use for large-scale deployment.

GPS sensors have also been used to extract vehicle trajectories [17]. While GPS directly provides position and velocity information, its average accuracy is about

4 meters in position. Advanced GPS technologies improves accuracy of but not economically feasible for large-scale use.

2.1.2 Camera based methods

It appears that camera sensors offer a good balance between accuracy and cost. Although more work is needed to extract ground vehicle trajectories from traffic footages, cameras are very inexpensive and simple to calibrate. Additionally, traffic cameras are already a common feature of road infrastructure, and in some cases, online streams are accessible. As a result, we concentrate on methods based on fixed cameras in this work because they offer a practical, scalable, and lightweight method for gathering traffic data.

- **Classical computer vision**: Vehicle detection and tracking using traditional computer vision techniques has been extensively researched.[2] provides a thorough analysis of the methods currently used for vehicle tracking at road intersections. The *Urban Tracker*[6] uses background subtraction to detect moving objects, followed by feature points and spatial information for tracking and handling occlusions. However, this method only provides positions of the vehicles in pixels. In [8], the authors address the problem of detecting vehicles and estimating their ground location and shape by matching image intensity edges to a deformable generic 3D model of vehicle. These techniques yield intriguing results, but because they rely on background subtraction or edge detections to identify vehicles, they lack overall robustness and reliability.
- **Deep Learning**: In recent years, Deep Learning methods have shown impressive increase in performance for object detection [15]. These techniques typically employ tracking-by-detections, which involves identifying objects in each frame and connecting them to create tracks.

2.1.3 Sensor based detection

The paper [12] authored by Dominik Notz et al. presents a multi-stage approach for trajectory extraction and assessment using camera and radar sensors. The following is a mathematical analysis of the method.

The study used a naturalistic driving dataset collected from a highway infrastructure, which includes both camera and radar sensors. The data collected from the sensors were preprocessed to remove noise and calibration errors, which is an essential step for accurate trajectory extraction. The extracted trajectories were then compared with ground truth data obtained from GPS and IMU sensors installed in the test vehicle, which ensured the accuracy of the extracted trajectories.

The first stage of the method involves object detection using the YOLOv3 object detection model. Let I be the input image, and let O be the set of detected objects in I. The YOLOv3 model computes the probability of each object in O , denoted as P(O), and the bounding box coordinates of each object, denoted as $B(O) = \{x, y, w, h\}$, where x and y are the coordinates of the top-left corner of the bounding box, and w and h are the width and height of the bounding box.

The second stage of the method involves object tracking using the Kalman filter. Let $x(t)$ be the state vector of the object at time t, which includes the position, velocity, and acceleration of the object. The Kalman filter estimates the state vector of the object at time t based on the state vector at time $t-1$ and the observed position of the object at time t, denoted as $z(t)$. The state vector of the object at time t, denoted as $\hat{x}(t)$, is given by:

$$\hat{x}(t) = Ax(t-1) + Bu(t-1) + K(t)[z(t) - Hx(t-1) - Bu(t-1)]$$

where A is the state transition matrix, B is the control matrix, $u(t-1)$ is the control input at time t $1, K(t)$ is the Kalman gain, H is the observation matrix, and $z(t)$ is the observed position of the object at time t.

The third stage of the method involves trajectory extraction by matching the tracked objects with the ground truth data. Let T be the set of extracted trajectories, and let G be the set of ground truth trajectories obtained from GPS and IMU sensors installed in the test vehicle. The trajectories in T and G are represented as a sequence of positions, denoted as $P(T)$ and $P(G)$, respectively. The matching process involves finding the trajectory in G that best matches each trajectory in T based on a distance metric. In this study, the Euclidean distance was used as the distance metric.

The accuracy of the extracted trajectories was evaluated using various metrics, such as mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). Let N be the number of trajectories, and let $P(T, i, j)$ and $P(G, i, j)$ be the jth position of the ith trajectory in T and G, respectively. The MAE, MSE, and RMSE are defined as:

$$MAE = (1/N)\Sigma_{\mathrm{i}}\Sigma_{\mathrm{j}}|P(T, i, j) - P(G, i, j)|$$
$$MSE = (1/N)\Sigma_{\mathrm{i}}\Sigma_{\mathrm{j}}(P(T, i, j) - P(G, i, j))^2$$
$$RMSE = \mathrm{sqrt}(MSE)$$

where Σ_{i} and Σ_{j} denote the sum over all trajectories and positions, respectively.

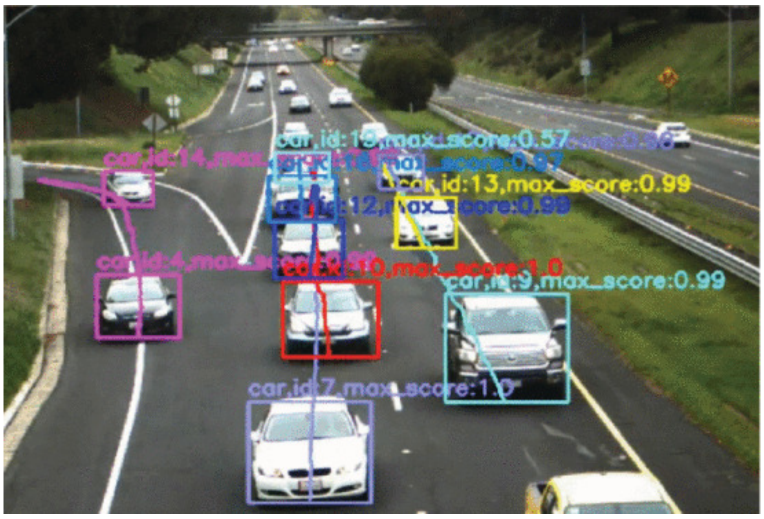

FIGURE 2.1: Example of a highway merge scene and Visualization of the detected vehicles and their past trajectories obtained by connecting the lower center points of all bounding boxes within on track. Source: Paper by Dominik Notz et al. [12]

Chapter 3

Objective and Problem Definition

3.1 Objective

As a part of this thesis, we try achieve the following set of the objective to do the analysis.

- Reviewing the past works done in the domain of Object detection, tracking and traffic analysis.
- Reviewing past work conducted on vehicle trajectory extraction.
- Reviewing past work conducted on vehicle velocity estimation.
- Devising methods to capture information of the world coordinate geometry from two-dimensional frames within videos.
- Assessing the model based on metrics that will help us in determining the accuracy and precision of the trained model.

3.2 Problem Definition

Vehicle Detection and Vehicle Trajectory analysis

- Past work has been done to develop models that carry out road traffic monitoring method based on image processing methods.
- Work has been done on analysing the traffic pattern by counting the number of vehicles and determining the category of the vehicles along with their direction of movement.
- Thus the task within the scope of Traffic analysis is to estimate the vehicle speed and tracking the vehicle trajectory i.e the path which the vehicle takes on the road.

Chapter 4

Background Material

4.1 R-CNN framework

R-CNN (Region-based Convolutional Neural Network) is a deep learning framework for object detection in images. It was first introduced by Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik in 2014 [4].

The RCNN algorithm is composed of three main components:

1. **Region Proposal**: A selective search algorithm that generates a set of candidate object bounding boxes in the input image.
2. **Convolutional Neural Network(CNN)**: A feature extraction network such as a VGG network that processes each candidate box and extracts features.
3. **SVM (Support Vector Machine)**: A classifier that is trained to classify each candidate box into one of the object classes or a background class.

4.1.1 Selective Search Algorithm

1. First step is to create as many regions as is needed to create the initial sub-segmentation, with each area belonging to no more than one item.

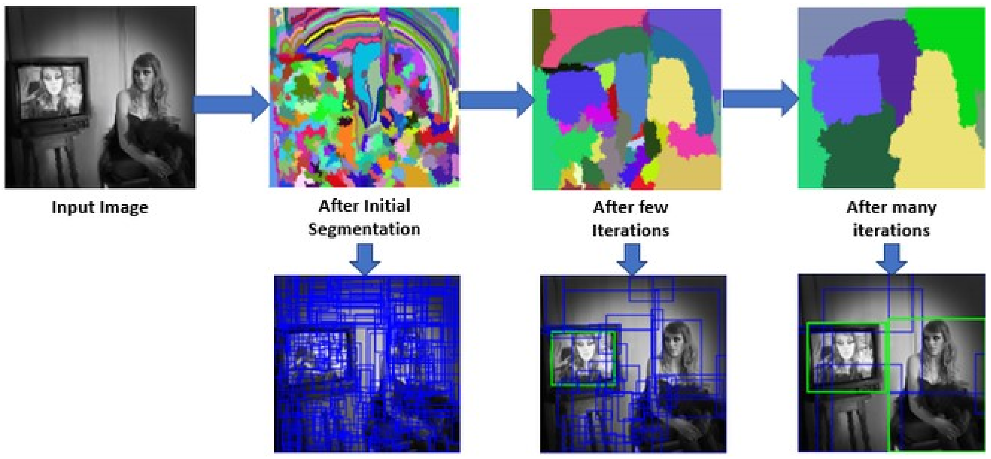

FIGURE 4.1: An example of Selective search Algorithm in work. Source: Stanford Course Teaching Material

2. Second step is to apply the Greedy algorithm to repeatedly combine related sections into larger ones.
   (a) Pick the two regions that are most similar among the group.
   (b) Consolidate them into a single, more expansive area.
   (c) Continue until just one region is left.
3. Third step is to create candidate object locations using the generated regions

4.1.1.1 Similarity in Segmentation

The selective search paper considers four types of similarity when combining the initial small segmentation into larger ones. These similarities are:

- **Color Similarity**: Specifically for each region we generate the histogram of each channels of colors present in image. In the paper 25 bins are taken in histogram of each color channel. Similarity is found using equation below: $S_{color}(r_i, r_j) = \sum_{k=1}^{n} min(c_i^k, c_j^k)$
  $C_i^k, c_j^k = k^{th}\, value\, of\, histogram\, bin\, of\, region\, r_i\, and\, r_j\, respectively$
- **Texture Similarity**: Texture similarity are calculated using generated 8 Gaussian derivatives of image and extracts histogram with 10 bins for each color channels.

$S_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k)$
$t_i^k, t_j^k = k^{th}\, value\, of\, texture\, histogram\, bin\, of\, region\, r_i\, and\, r_j\, respectively$

- **Size Similarity**: The basic idea of size similarity is to make smaller region merge easily.
  $S_{size}(r_i, r_j) = 1 - (size\,(r_i) + size\,(r_j)) \div size\,(img)$
  $where\, size\,(r_i)\,,\, size\,(r_j)\, and\, size\,(img)\, are\, the\, sizes\, of\, regions\, r_i\,,\, r_j\, and\, image\, respectively\, in\, pixels$

- **Fill Similarity**: Measures how well two regions fit with each other. If two region fit well into one another then they should be merged, if two region does not even touch each other then they should not be merged.
  $S_{fill}(r_i, r_j) = 1 - (size\,(BB_{ij}) - size\,(r_i) - size\,(r_j)) \div size\,(img)$
  $size\,(BB_{ij})\, is\, the\, size\, of\, bounding\, box\, around\, i\, and\, j$

Above four similarities combined to form a **resultant similarity**.
$S_{(r_i,r_j)} = a_1 * s_{color}(r_i, r_j) + a_2 * s_{texture}(r_i, r_j) + a_3 * s_{size}(r_i, r_j) + a_4 * s_{fill}(r_i, r_j)$
$where\, a_i\, is\, either\, 0\, or\, 1\, depending\, upon\, we\, consider\, this\, similarity\, or\, not$

4.1.2 Region Proposal

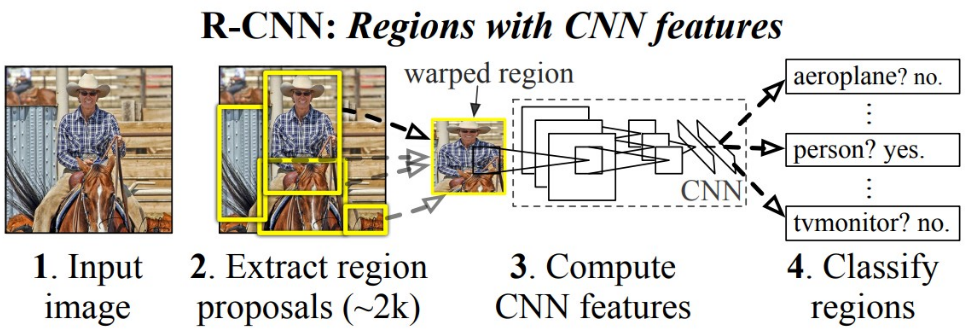

FIGURE 4.2: Major steps in R-CNN network. Source: R-CNN paper [4]

R-CNN combines CNN and region-based proposals. The detector has access to a set of candidate detections called region proposals. R-CNN only chooses a few of the sliding windows that CNN runs across the entire image. For an image, R-CNN uses 2000 regions. Segmentation algorithms, which employ selective search, are used in region proposals.

4.1.3 Object detection in R-CNN

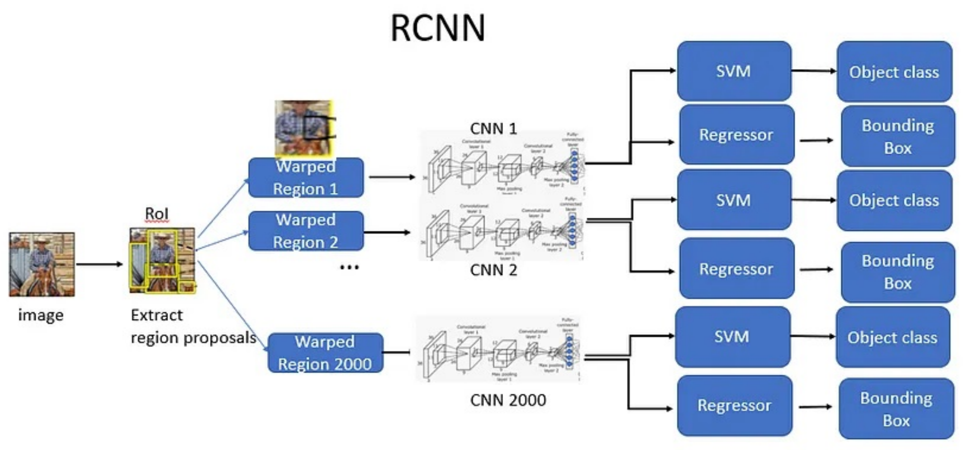

FIGURE 4.3: R-CNN network architecture. Source:Renu Khandelwal's blog

1. **Region Proposal**: Generate category-independent region proposals using selective search to extract around 2000 region proposals. Warp each proposal.
2. **CNN**: Warped region proposals are fed to a large CNN such as VGG network. It acts as a feature extractor that extracts a fixed-length feature vector from each region. After passing through the CNN, R-CNN extracts a 4096-dimensional feature vector for each region proposal
3. **SVM**: A SVM is applied to the extracted features from CNN. It helps to classify the presence of the object in the region. Regressor is used to predict the four values of the bounding box

4.1.3.1 Greedy Non-Max suppression

To all scored regions in an image, greedy non-maximum suppression is applied. Non-Max suppression rejects a region if it has an intersection-over union (IoU) overlap with a higher scoring selected region larger than a learned threshold. Finding an object with a single bounding box is the goal of object detection. However, with object detection, it is possible to find multiple detections of the same objects. Non-Max suppression guarantees that an object will only be detected once.

4.1.3.2 Scope of improvement

However R-CNN is slow and expensive. So the following improvements are made and implemented in Fast R-CNN:

1. Instead of deploying 2,000 ConvNets for each region of the image, use one ConvNet to process the entire image.
2. As opposed to R-CNN, which employs three different models, use a single model for feature extraction, classification, and generating bounding boxes.

4.2 Fast R-CNN

Fast R-CNN is a deep learning framework for object detection in images that improves upon the original R-CNN algorithm by being faster and more accurate. It was first introduced by Ross Girshick in 2015 [3].

The Fast R-CNN algorithm is composed of three main components:

1. **Convolutional Neural Network (CNN)**: A feature extraction network, such as a VGG or ResNet network, that processes the input image and extracts feature maps.
2. **Region of Interest (ROI) Pooling**: A layer that extracts fixed-size feature maps from the feature maps generated by the CNN for each candidate object proposal.
3. **Softmax Classifier and Regression Network**: A network that takes the fixed-size feature maps generated by the ROI Pooling layer and classifies and refines the object proposals

Fast R-CNN network takes image and a set of object proposals as an input. In contrast to R-CNN, Fast R-CNN extracts features for the entire image at once using a single deep ConvNet.

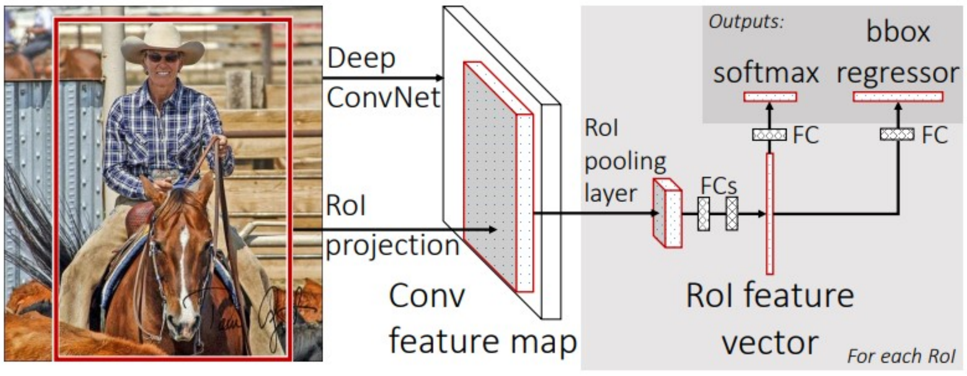

FIGURE 4.4: Fast R-CNN architecture. Source: Fast R-CNN paper [3]

Using selective search, a set of ROIs (Region of Interest) is also produced for the image. For the purpose of object detection, the region of interest (RoI) layer extracts a fixed-length feature vector from the feature map for each proposed object. With only one pyramid level, the RoI layer is a special case of the spatial pyramid pooling layer.Fully Connected layers(FC) needs fixed-size input. Hence ROI Pooling layer is used to warp the patches of the feature maps for object detection to a fixed size.

ROI pooling layer is then fed into the FC for classification as well as localization. Max pooling is used by RoI pooling layer. It creates a tiny feature map from the features contained in any legitimate region of interest.

A fully connected layer branches into two sibling output layers:

1. One with softmax probability estimates over k object classes and a general "background" class
2. And another with a regressor to produce four real-valued numbers for the refined bounding box position for each of the k object classes.

Comparison of R-CNN and Fast R-CNN

- Fast R-CNN only uses one Deep ConvNet to extract features. Compared to R-CNN, which uses 2000 ConvNets for each region of the image, using a single deep ConvNet significantly speeds up the image processing.
- Fast R-CNN substitutes softmax for SVM in the object classification process. SVM is slightly outperformed for objection classification by Softmax.

- Fast R-CNN employs multi task loss to complete Deep ConvNets training from beginning to end, improving detection accuracy.

However, Fast R-CNN uses a slow and time-consuming process called selective search as a proposal method to find the Regions of Interest. This is inappropriate for huge real-world data sets.

4.3 Faster R-CNN

Faster R-CNN is a deep learning framework for object detection in images, and an extension of the original RCNN and Fast R-CNN algorithms. It was first introduced by Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun in 2015 [14].

The key innovation of Faster R-CNN is the introduction of a Region Proposal Network (RPN), which replaces the selective search algorithm used in RCNN and Fast R-CNN for generating candidate object proposals. The RPN is an end-to-end deep neural network that shares the convolutional features with the object detection network, which makes it much faster than selective search.

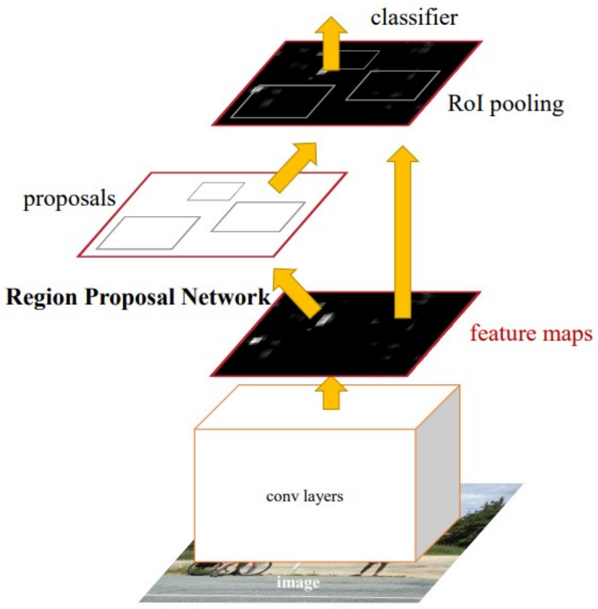

FIGURE 4.5: Faster R-CNN is a single unified network for object detection. Source: Faster R-CNN by Shaoqing Ren et al. [14]

Faster R-CNN consists of two stages:

1. First stage is the deep fully convolutional network that proposes regions called a Region Proposal Network(RPN)
2. Second stage is the Fast R-CNN detector. It uses RoIPool to extract features from each candidate box before performing classification and bounding-box regression.

4.3.1 Region Proposal Network(RPN)

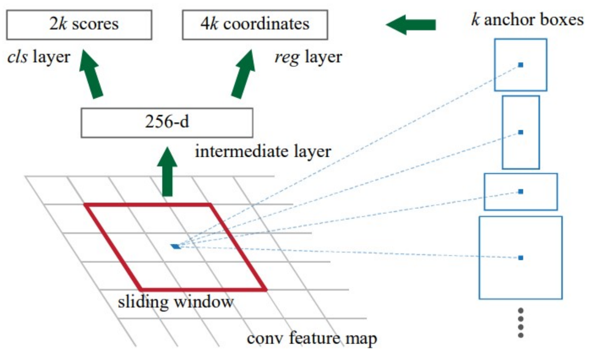

FIGURE 4.6: Region Proposal Network. Source: Faster R-CNN paper [14]

Region Proposal Network takes an image of any size as input and outputs a set of rectangular object proposals each with an objectness score. It does this by sliding a small network over the feature map generated by the convolutional layer. RPN shares computation with a Fast R-CNN object detection network.

Two sibling fully connected layers: a box-regression layer for the bounding box and a box-classification layer for object classification , are fed with the feature generated by RPN. It is efficient and processes 10 ms per image to generate the ROI's.

4.3.2 Anchors

An anchor coupled with a scale and aspect ratio is centred at the concerned sliding window. Faster R-CNN produces 9 anchors at each sliding window using 3 scales and 3 aspect ratios. Anchors help with translational invariance.

Multiple region proposals are predicted simultaneously at each sliding window location. The letter k stands for the maximum number of proposals that can be made for each location.

The *CLS layer* produces 2k scores that estimate the probability of object or not object for each proposal, while the *Reg layer* produces 4k outputs encoding the coordinates of k boxes.

4.3.3 Architecture and working of Faster R-CNN

Faster R-CNN is composed of 3 different neural networks

1. **Backbone network**: Feature network that uses a deep convolutional layer to produce feature maps from the input image
2. **Region Proposal Network (RPN)**: RPN uses 9 anchors for each sliding window and is used to identify various regions. This aids the translational invariance. Region of Interests (ROIs) are a collection of bounding boxes created by RPN that have a high probability of containing an object.
3. **Fast R-CNN** : Detection Network is the R-CNN which takes input as the feature maps from the convolutional layer and the RPN network. This produces the object's class and bounding boxes.

Thus, Faster R-CNN takes image as an input and is passed through the Feature network to generate the feature map. RPN uses the feature map from the Feature network as an input to generate the rectangular boxes of object proposals and the objectness score. The predicted region proposals from RPN are then reshaped using a RoI pooling layer. Warped into a fixed vector size.Warped fixed-size vector is then fed into two sibling fully connected layers, a regression layer to predict the offset values for the bounding box and a classification layer for object classification

4.4 Mask R-CNN

Mask R-CNN (Regional Convolutional Neural Network) is a popular deep learning framework used for object detection, instance segmentation, and image captioning. It was first introduced by Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick in 2017 [5].

Mask R-CNN is an extension of the Faster R-CNN object detection algorithm, which adds a parallel branch to the network that generates masks for each instance of an object detected in an image. The masks provide pixel-level segmentation of objects, which allows for more precise object detection and classification.

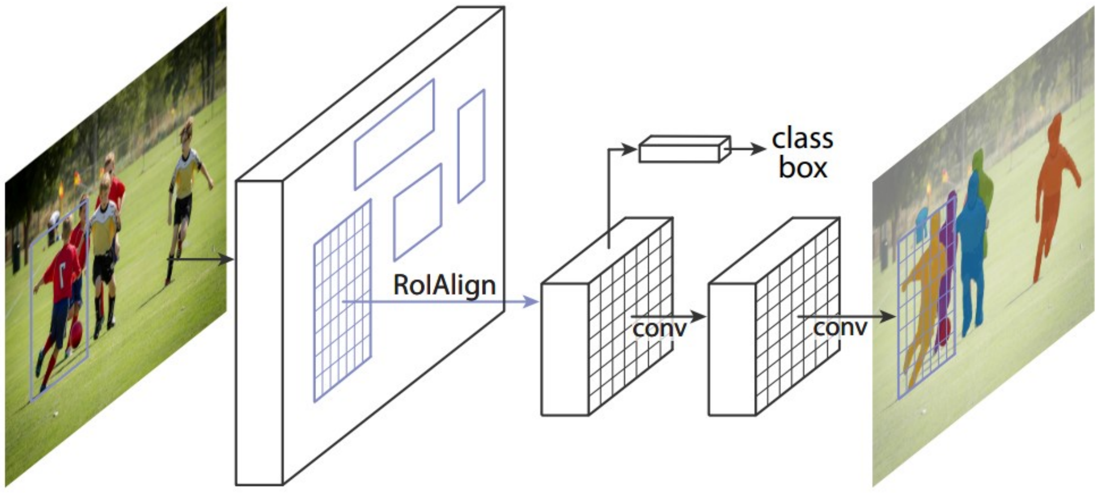

FIGURE 4.7: The Mask R-CNN framework for instance segmentation. Source: Mask R-CNN paper by Kaiming He et al. [5]

4.4.1 Framework of Mask R-CNN

The Mask RCNN has two stages. Based on the input image, it first generates proposals for regions where an object might be. On the basis of the first stage proposal, it then predicts the object's class, fine-tunes the bounding box, and generates a mask at the pixel level for the object. The backbone structure is connected to both stages. The Mask R-CNN algorithm is composed of five main components:

1. **Convolutional Neural Network (CNN)**: A feature extraction network, such as a ResNet network, that processes the input image and generates a set of feature maps.

2. **Region Proposal Network (RPN)**: It uses the feature map generated by CNN to generate multiple Region of Interest(RoI) using a lightweight binary classifier. It does this using 9 anchors boxes over the image. The classifier returns object/no-object scores. Non Max suppression is applied to Anchors with high objectness score.

3. **ROI (Region of Interest) Align**: The RoI Align network outputs multiple bounding boxes rather than a single definite one and warp them into a fixed dimension.

4. **FC Layer**: Warped features are then fed into fully connected(FC) layers to make classification using softmax and boundary box prediction is further refined using the regression model

5. **Mask Head**: Warped features are also fed into Mask classifier, which consists of two CNN's to output a binary mask for each RoI. Mask Classifier allows the network to generate masks for every class without competition among classes

4.4.2 Region of Interest (RoI) Align Layer

The ROI Align layer is a critical component of the Mask R-CNN algorithm that extracts fixed-size feature maps from the feature maps generated by the CNN for each object proposal. This layer is an improvement over the earlier ROI Pooling layer used in the Fast R-CNN algorithm, which suffered from misalignment issues that could lead to loss of spatial information and decreased segmentation accuracy.

The ROI Align layer addresses the misalignment issue by using bilinear interpolation to calculate the values of the output feature maps at non-integer spatial locations. Specifically, for each object proposal, the ROI Align layer divides the proposal into a fixed number of spatial bins and aligns these bins to the feature map grid using sub-pixel offsets. The values of the output feature maps at each bin location are

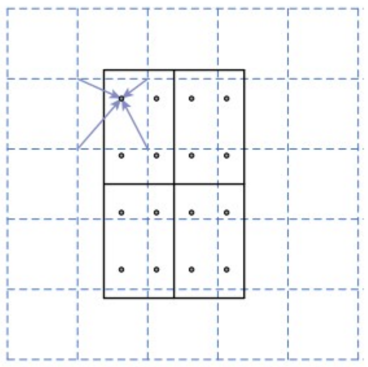

Figure 4.8: The dashed grid represents a feature map, the solid lines an RoI (2×2 bins here), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points. Source: Mask R-CNN paper [5]

then calculated by bilinearly interpolating the feature map values at the four nearest grid points.

Compared to the ROI Pooling layer, which simply rounds the bin locations to the nearest integer spatial locations and uses max pooling to calculate the output feature map values, the ROI Align layer preserves more spatial information and produces more accurate feature maps.

Multi-task loss is defined on each sampled RoI as $L = L_{cls} + L_{box} + L_{mask}$. The classification loss L_{cls} and bounding-box loss L_{box} are same as those defined in Fast R-CNN. The mask branch has a Km^2-dimensional output for each RoI, which encodes K binary masks of resolution mm, one for each of the K classes. To this we apply a per-pixel sigmoid, and define L_{mask} as the average binary cross-entropy loss. For an RoI associated with ground-truth class k, L_{mask} is only defined on the k-th mask (other mask outputs do not contribute to the loss).

4.4.3 Anchor Boxes

In Mask R-CNN, the anchor box is a predefined rectangular box used to generate object proposals in the Region Proposal Network (RPN). Anchor boxes are designed to cover a range of object sizes and aspect ratios, and are placed at evenly spaced intervals across the feature maps generated by the CNN.

The RPN uses a sliding window approach to generate object proposals based on the feature maps and anchor boxes. For each anchor box, the RPN predicts two scores: the probability that the anchor box contains an object, and the coordinates of a bounding box that tightly encloses the object if one is present. The RPN also predicts a set of binary masks for each object proposal.

During training, the RPN is optimized to predict accurate object proposals and binary masks for the ground truth objects in the training dataset. The predicted object proposals are then used as input to the ROI Align layer, which extracts fixed-size feature maps for each object proposal. The fixed-size feature maps are then processed by the Mask Head network to generate binary masks for each object proposal.

4.4.4 Network Architecture

The Mask R-CNN network architecture consists of two main components: the backbone architecture used for feature extraction over the entire image, and the network head for bounding-box recognition (classification and regression) and mask prediction applied separately to each RoI.

The backbone architecture is denoted using the nomenclature network-depth-features and is evaluated with *ResNet* and *ResNeXt* networks of depth 50 or 101 layers. A more effective backbone was recently proposed by *Lin et al.* called called a Feature Pyramid Network (FPN). It uses a top-down architecture with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R-CNN with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale.

For the network head, the Faster R-CNN box heads from the ResNet and FPN papers are extended. The head on the ResNet-C4 backbone includes the 5-th stage of ResNet, which is compute-intensive. For FPN, the backbone already includes res5 and thus allows for a more efficient head that uses fewer filters.

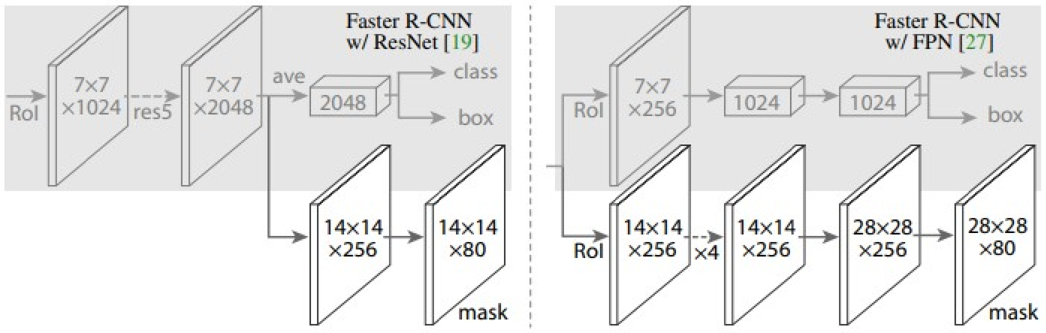

FIGURE 4.9: **Head Architecture**: Two existing Faster RCNN heads are extended. Left/Right panels show the heads for the ResNet C4 and FPN backbones, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context. All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1. *Right*: '×4' denotes a stack of four consecutive convs. Source: Mask R-CNN paper by Kaiming He et al. [5]

4.4.5 Instance Segmentation

The Semantic Segmentation detects all the objects present in an image at the pixel level. Outputs regions with different classes or objects. Instance Segmentation is identifying each object instance for every known object within an image.

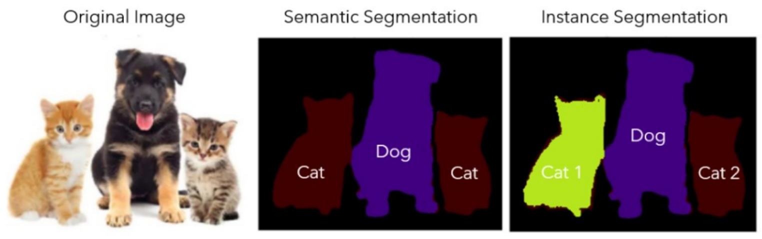

FIGURE 4.10: Difference between Semantic and Instance segmentation. (Source)

Chapter 5

Methodology

5.1 Training of Model

The segmentation model Mask R-CNN was trained based on the COCO dataset[9] with the relevant object categories. The Average Precision (AP) is calculated for each category of object, and for different object size, and over all the objects

	Average Precision (AP)							
Training Dataset	**overall**	**small**	**medium**	**large**	**car**	**person**	**bicyke**	**transporter**
COCO extended	55.95	49.72	56.42	62.23	94.07	36.15	60.71	32.88

5.2 Preprocessing of Raw Dataset

The input videos were pre-processed by first converting the videos from MTS format to MP4.This was done for two reasons.

- OpenCV does not support reading video files in the MTS format.
- The input video dataset is of 18.6 GB which poses some trouble while processing them by the kernel

The audio component in the videos is entirely unnecessary while detecting and tracking the vehicles. Thus, the audio was removed while converting the videos to MP4 format. Also, the bitrate of the videos was reduced to increase the compression ratio. This was done as a minor reduction in the high-frequency features of the frames in the video will not adversely affect the detection accuracy of the object detector.

5.3 Configurations for the input dataset

After the pre-processing of the input video, the configurations of the object detector model were tuned. Firstly, The Region of Interest was defined. For finding the coordinates of the boundary points on the image frame, OpenCV functions were used to map the locations of the mouse pointer click to the coordinates within the image. We find 4 points which define a polygon (in our case a rectangle) in anti clockwise direction. This defines the area in which the model will detect the vehicles. The world coordinate points corresponding to camera coordinate points need to be supplied for the camera calibration.

5.4 Data set Explaination

The data set consists of traffic videos captured using a Camcorder. The video consists of different vehicles moving in any particular direction. Our aim is to classify these vehicles and their direction.

The required class labels were defined. Four categories of vehicles used for classification: Car, Transporter(Truck and Bus), Person and Bicycle.

5.5 Trajectory Extraction

The method for extracting the trajectory of the detected objects consists of four major steps. Those are as follows:

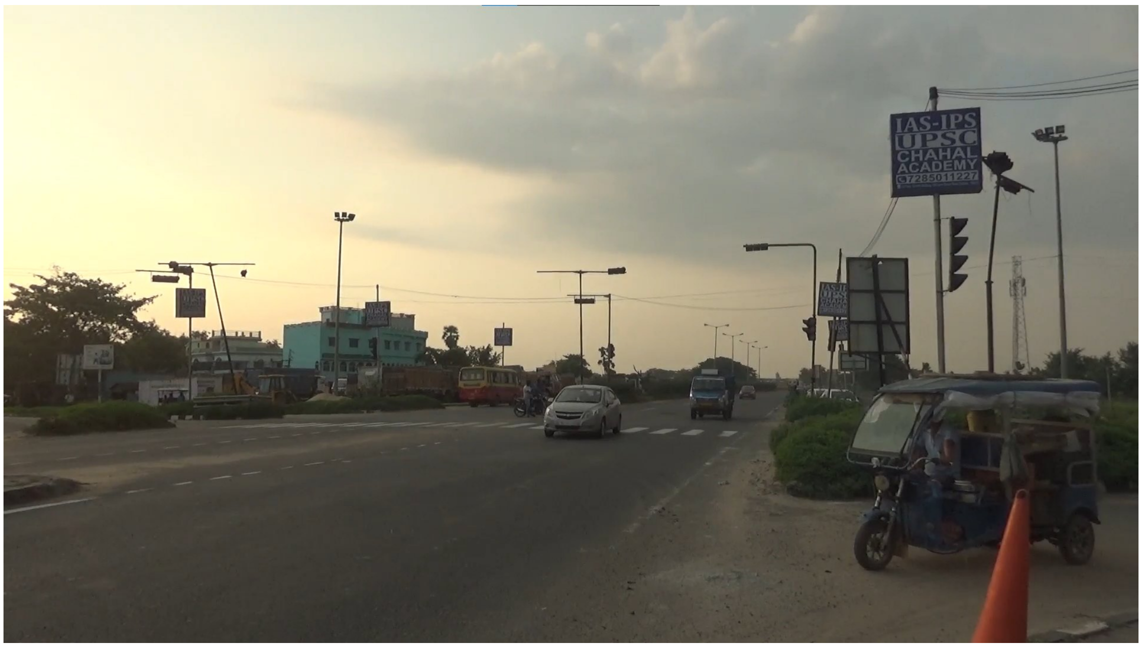

FIGURE 5.1: A sample frame from a video of the given dataset.

5.5.1 Segmentation

The first step is instance segmentation, in which objects are detected and segmented frame-by-frame using the Mask R-CNN, a deep neural network implemented in the Detectron 2 open-source platform. The state-of-the-art Mask R-CNN used for this purpose, which allows accurate segmentation of objects. The network weights are provided from training with the COCO-dataset, which is used as a starting point for transfer learning. The categories car, transporter, person, and bicycle are focused on in this work. The main reason for using the segmentation based approach is that it provides a better accuracy in estimating the vehicle reference point.

5.5.2 Tracking

The second step is tracking, where the objects are tracked using the tracking-by-detection approach, where two methods, SORT[1] and Deep SORT[16], are considered. A common tracker used in predictable situations with little likelihood of overlapping object trajectories is called SORT (Simple, Online, Realtime Tracker). To forecast tracks, SORT uses a tiny amount of CPU power. The Kalman filter, which multiplies matrices and does other linear algebra operations to forecast the location of the ensuing bounding box, is the most difficult component. Either the Mahalanobis (an alternate implementation) or the IoU (canonical version) metric is used by the SORT to estimate the items. For axis-aligned bounding boxes, the IoU metric may be calculated quickly; for rotated bounding boxes, it takes more time and resources. In terms of tracking accuracy and precision, SORT performs admirably. However, SORT fails in cases of occlusion and only returns tracks with a significant number of ID switches. The association matrix that was used is to blame for this.

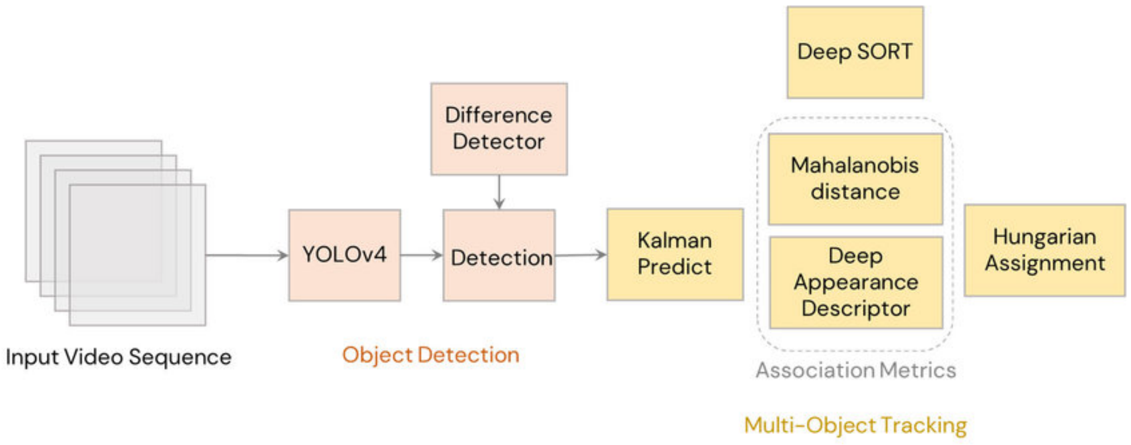

FIGURE 5.2: Overview of the object tracking Deep-SORT algorithm. (Source)

A more effective association metric is used by DeepSORT, which combines motion and appearance descriptors. DeepSORT is a tracking algorithm that tracks objects based on both their appearance, which is extracted as a feature vector using a CNN, as well as their velocity and motion. Both the methods discussed above are based on tracking bounding boxes. But the method employed to create the bounding box is not the conventional method used in YOLO networks. The method is discussed in the next step. Also both methods are validated to test the tracking capability only. To the test the tracking capability the following metrics were used: track fragmentations (Frgm.), MOTA, MOTP, mostly lost (ML) and mostly tracked (MT) are defined and explained in the paper "Simple online and realtime tracking with a deep association metric(SORT) by N. Wojke et al."[1].

5.5.3 Camera Calibration

The third step is camera calibration, which is necessary to convert pixel coordinates of the camera frame into geographic (world) coordinates. The process of estimating the parameters of a camera is called camera calibration. This means we have all the information (parameters or coefficients) about the camera required to determine an accurate relationship between a 3D point in the real world and its corresponding 2D projection (pixel) in the image captured by that calibrated camera.

$$\begin{bmatrix} u' \\ v' \\ z' \end{bmatrix} = P \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$
$$u = \frac{u'}{w'}$$
$$u = \frac{u'}{w'}$$

Where, P is a 3×4 Projection matrix consisting of two parts — the intrinsic matrix C that contains the intrinsic parameters which are determined by the internal structure of the camera and the lens, like the focal length fx, fy, and the center of the camera image px, py, and the extrinsic matrix $[R|t]$ that is combination of 3×3 rotation matrix R and a 3×1 translation t vector.

$$\begin{gathered} P = C * [R|t] \\ x_{pixel} = C * [R|t] x_{world} \\ C = \begin{bmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \\ x_{pixel} = \begin{bmatrix} u \\ v \end{bmatrix} X_{world} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \end{gathered} \tag{1}$$

The s is the Skew coefficient, which is non-zero if the image axes are not perpendicular.

For the calculation of the intrinsic and extrinsic parameters it is necessary to create two matrices X_{pixel} and X_{world} with related pairs of points, also known as Ground Control Points (GPC), in both coordinate systems. The library OpenCV has a method *calibrateCamera* which was used to calculate the Camera Matrix. The following re-projection error is optimised using this function: $\sum_{0\ldots n} \|x_{pixel,n} - x'_{pixel,n}\|$ using the ***Levenberg-Marquardt algorithm*** , where x'_{pixel} is calculated from applying the equation 1 with points from x_{pixel}. The total number of all points is referred to as n (GPC).

5.5.4 Object Reference point

An estimate of the object's precise location must be made and projected into street level, which serves as the trajectory's reference point. The 3D bounding box's ground plane closely resembles the vehicle's actual ground plane. The smallest rectangle containing the object's segmentation is the rotated bounding box. This approach is not used in the scenario where the edges of the rotated bounding box are parallel to the axes of the coordinate system. Instead, the ground plane is thought to represent the rotating Bounding Box.The bounding box's vertices are arranged in ascending order by vertical coordinate for the roof and ground edges. The highest v-coordinate point belongs to the ground edge.Then two potential points are identified that, given the rotation of the bounding box, could serve as the end point of the ground edge. The longer edge can be identified as the ground edge by comparing the separation between those two points. The roof edge is created by the other two points of the rectangle.The edges of a 3D bounding box can be created by employing the chosen edges and translating the related points into the world coordinate system. The front and back edges can be identified by rotating the ground edge by 90 degrees anticlockwise around height h using the rotation matrix R_h. Since a two-dimensional image cannot contain depth information. As a result, the category car's width w is set at a maximum of 2 metres. To determine the rear and front edge from the ground plane, cosine similarity between the motion vector and ground edge is used. The reference point is chosen as the centre of the rear axle, as is accepted by various traffic simulation software like *SUMO* [10].

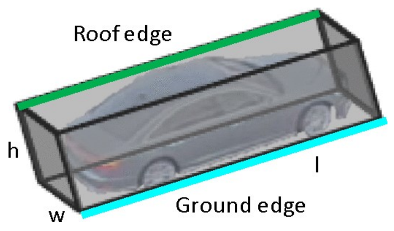

FIGURE 5.3: Bounding box in 3D constructed from ground and roof edge

5.6 Output format

A CSV file is also produced along with the output video which gives the trajectory information. It shows the vehicles detected in each frame with their corresponding locations and classes with the following headings.

- **track-id:** the ID of the object being tracked
- **frame-id:** the current frame
- **class-id:** the detected class of the object
- **velocity:** the instantaneous velocity of the object
- **coordinate-world-x-opt:** coordinate in x of the world coordinate of the trajectory with vehicle rear axle as the reference point
- **coordinate-world-y-opt:** coordinate in y of the world coordinate of the trajectory with vehicle rear axle as the reference point
- **coordinate-world-x:** coordinate in x of the world coordinate of the trajectory with centre of bounding box as the reference point
- **coordinate-world-y:** coordinate in y of the world coordinate of the trajectory with centre of bounding box as the reference point

Chapter 6

Experimental Results

6.1 Deliverables

6.1.1 Procedure

The following steps were followed :

- **Load the model** Activate Google Colab runtime environment with GPU. Upload the python code files into the drive.
- Check the state and presence of the GPU using the command **nvidia-smi**
- Install the required libraries by using the command **pip install -r requirements.txt**. Install *detectron2* by following the official documentation.
- Download the network weights of the Mask R-CNN model from the given link in the code file and store it in the weights folder.
- **Preparing the inputs:** Upload the Input Video and json file with specifications for the **Region of Interest** and the world to camera calibration points.
- Run the code using the command **python run_on_video.py** with the arguments: –input-path(give location of the input video file) , –valid-area (for the RoI) , –passpoints (file with camera calibration points)

6.1.2 Results

6.1.2.1 Output video with Detected Objects

Class 0: Bicycle. Class 1: Car. Class 2: person. Class 3: Transporter.
The heavy vehicles like truck and bus will be classified as 'transporter'. Each pixel of the object is coloured to show the specific object. The blue rectangle formed under a vehicle is the predicted base-plate of the vehicle. The speed in kilometres per hour (kmph) is displayed with each object. The confidence of of the predicted object class is also displayed alongside the detected object boundary.

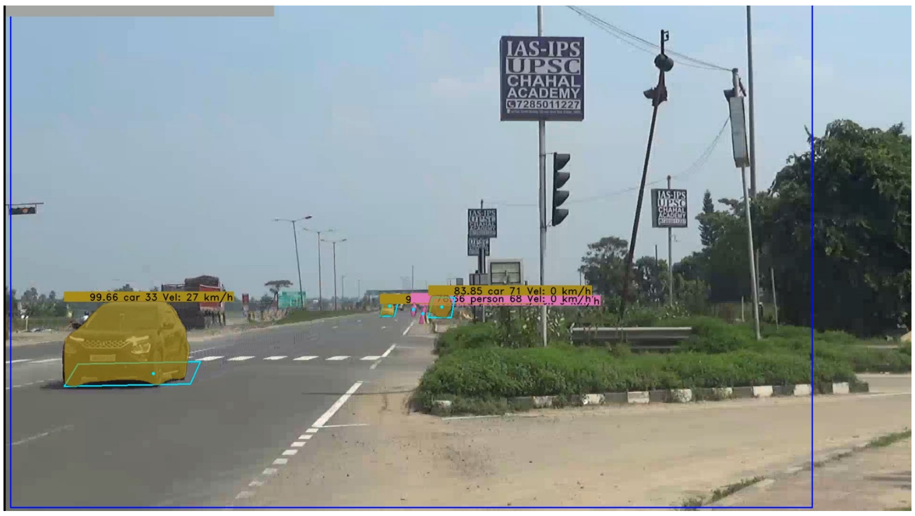

Figure 6.1: Detected vehicles in a frame of Video 00121

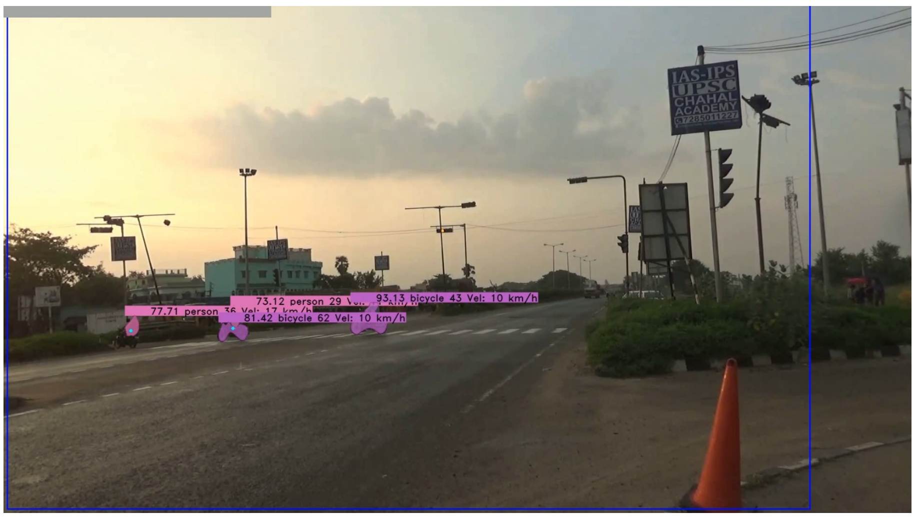
77.71 person 36 Vel: 17 km/h
73.12 person 29
81.42 bicycle 62 Vel: 10 km/h
93.13 bicycle 43 Vel: 10 km/h
IAS-IPS
UPSC
CHAHAL
ACADEMY

Figure 6.2: Detected vehicles in a frame of Video 00130

6.1.2.2 Output CSV File with Detections

The CSV file which is produced along with the output video gives the trajectory information. It has the world coordinates of the trajectory defined in the real world, the velocity of the object detected, the category of the object detected, and the ID assigned to the object by the SORT object tracker. This information is produced on a per frame basis. The exact format of the table has been described earlier.

frame_id	**class_id**	**track_id**	**velocity**	**coordinate_world_x_opt**	**coordinate_world_y_opt**	**coordinate_world_x**	**coordinate_world_y**
1258	person	83	3.059481167	678155.9363	5405296.452	678155.9363	5405296.452
1258	car	88	1.490473996	678145.7348	5405313.415	678145.0863	5405313.17
1258	person	92	0	678157.7732	5405294.546	678157.7732	5405294.546
1258	person	93	3.672338771	678156.2038	5405296.244	678156.2038	5405296.244
1258	car	97	0	678159.072	5405293.189	678159.0695	5405293.156
1258	transporter	99	0	678157.7316	5405296.472	678157.6693	5405296.402
1259	person	13	0	678157.782	5405292.576	678157.782	5405292.576
1259	car	57	18.05796681	678147.2738	5405302.519	678147.9452	5405305.332
1259	person	76	0.999079616	678152.4954	5405303.301	678152.4954	5405303.301
1259	person	83	4.19424912	678155.8756	5405296.472	678155.8756	5405296.472
1259	car	88	2.051727078	678145.7348	5405313.415	678145.0708	5405313.197
1259	person	92	0.610049858	678157.7591	5405294.571	678157.7591	5405294.571
1259	person	93	5.05812747	678156.2552	5405296.326	678156.2552	5405296.326
1259	person	95	1.180914574	678157.8836	5405293.495	678157.8836	5405293.495
1259	car	97	0	678159.072	5405293.189	678159.0695	5405293.156
1259	transporter	99	0	678157.7211	5405296.443	678157.6693	5405296.402
1260	person	13	0	678157.782	5405292.576	678157.782	5405292.576
1260	car	57	17.59733027	678147.2244	5405302.591	678147.899	5405305.492
1260	person	76	0.640660367	678152.4954	5405303.301	678152.4954	5405303.301
1260	person	83	4.19445799	678155.8615	5405296.496	678155.8615	5405296.496
1260	car	88	2.051727078	678145.7348	5405313.415	678145.0708	5405313.197
1260	person	92	0	678157.7732	5405294.546	678157.7732	5405294.546
1260	person	93	3.97012208	678156.2083	5405296.322	678156.2083	5405296.322
1260	person	95	1.180914574	678157.8836	5405293.495	678157.8836	5405293.495
1737	car	137	9.495916	678151	5405300	678151.4	5405302
1737	person	145	1.237704	678156.4	5405297	678156.4	5405297
1738	person	13	0	678157.8	5405293	678157.8	5405293
1738	person	92	0.928603	678157.9	5405294	678157.9	5405294
1738	person	106	0	678146.8	5405312	678146.8	5405312
1738	car	137	10.17137	678151	5405300	678151.3	5405302
1738	person	145	2.47727	678156.4	5405297	678156.4	5405297
1738	transporter	160	5.087342	678156.4	5405299	678156.1	5405300
1738	person	161	26.16971	678156.3	5405297	678156.3	5405297
1739	person	13	0	678157.8	5405293	678157.8	5405293
1739	person	92	0.928603	678157.9	5405294	678157.9	5405294
1739	person	106	1.498856	678146.8	5405312	678146.8	5405312
1739	car	137	10.69028	678150.9	5405300	678151.2	5405302
1739	car	140	1.855882	678157.7	5405296	678157.6	5405296

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Therefore, we reviewed some of the past work of notable scholars in the domain of traffic trajectory analysis using various methods which involve the use of additional hardware. Also a methodology for detecting the trajectory of a vehicle without the use of additional sensors was discussed in this report. While detecting the trajectory of vehicles at an intersection through a video camera the main issue was that the depth information of the world coordinates is lost while capturing a video. Thus a method was discussed in this report to derive a 3-D bounding box from a 2-D bounding box by using instance segmentation to create the bounding box instead of using a object detector such as YOLO[13]. Also it was found that the reference point chosen for detecting the trajectory of a vehicle plays a crucial role in the accuracy metrics. Thus an approach for the same has been discussed in the report which involves finding the base plate of a vehicle from its 3-D bounding box and then choosing the centre of the rear axle as the reference point.

7.2 Future Work

The B.Tech Project (BTP) work for this semester revolved around detection of different types of vehicles and determining the vehicular trajectory.

Future work includes more work in the following domains:

- To further improve the detection accuracy , the Mask R-CNN can be trained on a more varied dataset which has a data distribution closer to the Indian vehicles.
- Different approaches for estimating the depth in an image to world coordinate transformation can be utilized. This would result in better prediction of 3-D bounding boxes for the objects.
- Better algorithms can be applied to remove the distortion effect of the camera which would result in a more accurate camera calibration.
- Different methods can be employed to estimate the velocity of vehicle such as using Optical Flow of the consecutive frames and feeding it into a neural network can help improve the accuracy of the velocity estimation.
- The extracted trajectory can be analyzed to detect the violations of the traffic rules and also learn about human driving behaviour.

Bibliography

[1] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016.

[2] Sokèmi René Emmanuel Datondji, Yohan Dupuis, Peggy Subirats, and Pascal Vasseur. A survey of vision-based traffic monitoring of road intersections. *IEEE Transactions on Intelligent Transportation Systems*, 17(10):2681–2698, 2016.

[3] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[4] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[6] Jean-Philippe Jodoin, Guillaume-Alexandre Bilodeau, and Nicolas Saunier. Urban tracker: Multiple object tracking in urban mixed traffic. In *IEEE Winter Conference on Applications of Computer Vision*, pages 885–892, 2014.

[7] Annkathrin Krämmer, Christoph Schöller, Dhiraj Gulati, and Alois C. Knoll. Providentia - A large scale sensing system for the assistance of autonomous vehicles. *CoRR*, abs/1906.06789, 2019.

[8] Matthew J. Leotta and Joseph L. Mundy. Vehicle surveillance with a generic, adaptive, 3d vehicle model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1457–1469, 2011.

[9] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and

C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[10] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wiessner. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582, 2018.

[11] Hector Mejia, Esteban Palomo, Ezequiel López-Rubio, Israel Pineda, and Rigoberto Fonseca. Vehicle speed estimation using computer vision and evolutionary camera calibration. In *NeurIPS 2021 Workshop LatinX in AI*, 2021.

[12] Dominik Notz, Felix Becker, Thomas Kühbeck, and Daniel Watzenig. Extraction and assessment of naturalistic human driving trajectories from infrastructure camera and radar sensors. In *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pages 455–462, 2020.

[13] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[14] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[15] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[16] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017.

[17] Yu Zheng. T-drive trajectory data sample, August 2011. T-Drive sample dataset.