Extracting road traffic data through video analysis using automatic camera calibration and deep neural networks.

by

Vamsi Nadella

(Under the Direction of John A Miller)

Abstract

Due to a rapid increase in the amount of available traffic data, several companies and research groups are working on traffic forecasting. Vehicle traffic forecasting is predicting the amount of traffic and the speed of vehicles passing through a point. These data are provided by various sensors such as loop-inductance, microwave radars, laser sensors, and video cameras. Automated analysis on video live feeds in real time contributes to increased sources of data and helps as a redundant system for existing systems in case of unexpected device failures. We present a pipeline for extracting the traffic data from traffic videos which can be capable of filtering the incoming traffic from distractions such as non-vehicles, unwanted camera movements, etc. Calibrating traffic cameras automatically without explicit inputs is another important feature of this pipeline. This proposed pipeline will be capable of observing the count of vehicles passing through a point and their average speed with the help of pre-trained deep neural networks.

Index words: deep-neural-networks, image-processing, computer-vision, video analysis, traffic forecasting systems

Extracting road traffic data through video analysis using automatic camera calibration and deep neural networks.

by

Vamsi Nadella

B.Tech, V.R. Siddhartha Engineering College, 2016

A Thesis Submitted to the Graduate Faculty
of The University of Georgia in Partial Fulfillment
of the
Requirements for the Degree

Master of Science

Athens, Georgia

2019

Extracting road traffic data through video analysis using automatic camera calibration and deep neural networks.

by

Vamsi Nadella

Major Professor: John A Miller

Committee: Yi Hong
Shannon Quinn

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2019

Acknowledgments

I would like to thank my major professor and principal instructor, Dr. John A Miller, for his guidance, advice, and supervision. I would like to thank my committee members Dr. Yi Hong and Dr. Shannon Quinn for their support and guidance throughout this work.

I would like to thank my family members and friends for their continued support and inspiration. Without their love and sacrifices I would have never got to where I am today. My brother has been the pillar of my strength and would like to thank him for always being there for me and constantly guiding me.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Road traffic forecasting has been one of the major research fields for a long time and the forecasts play an important role in several places such as traffic flow handling, traffic navigation, emergency management, and city infrastructure planning. Federal government institutions such as the Department of Homeland Security were also highly interested in these technologies which aid for continuous surveillance and mass evacuations in times of emergency [4]. Basic statistics about road traffic data involve the number of vehicles passing through a point and their average speed. These two parameters are affected by several conditions such as accidents, traffic jams, bad weather or even slow moving vehicles. In order to facilitate better rider experience on roads, authorities need to study all the affecting parameters and the reasons causing them. Traffic monitoring consists of several systems for reporting each individual effect to authorities. Such as, for weather reporting, there were road weather sensing stations, microwave and infrared sensors for understanding the traffic flow, citizen reporting and emergency services data for information regarding accidents and collisions, etc. All these systems play an equally important role in everyday traffic handling tasks such as congestion handling. The knowledge that can be retrieved from a video about a particular incident is much more than that of the traditional sensors. The purpose of video cameras on roads have been limited to traffic surveillance for a long time but increasing research is taking place for the extraction of this knowledge from these cameras. Establishing a real-time video processing pipeline based on live traffic video feeds can support these systems. Existing systems in deployment involve calculating the speed from cameras attached with special laser apparatus but they are costly compared to the traditional cameras.

Information extraction from videos involves dealing with several challenges such as extreme weather conditions, a wide variety of imaging devices, unusual lighting conditions, unique geo-spatial scenes, etc. We came up with a series of tasks for solving the majority of these problems in order to develop a scalable, automatic and adaptive solution. Based on background subtraction and optical flow we initially create an image filter for each camera based on a two minute video to separate the incoming or outgoing vehicles from the rest of the objects in the video.

Creating this filter does not require any explicit inputs or manual efforts. As the filter consists of indices of the pixels that are coming towards the camera and going away from the camera, we can isolate the incoming traffic in the video frames assuming the direction of traffic flow does not change. We pass these frames through Faster R-CNN which is a deep neural network capable of object detection and pre-trained on eighty thousand images with eighty object classes[1] from Common Objects in Context (COCO) dataset [5]. The object classes consist of birds, persons, animals, cars, bikes, trucks, etc. The output of the neural network consist of object bounding boxes belonging to one of these classes. This helps us to recognize the vehicles on road. Tracking these vehicles individually over multiple frames provides us their pixel distances and counts.

In order to estimate the speed of the vehicle, we need to measure the distance traveled over time. Time taken by a vehicle to move between one point to another can be calculated by the number of frames it took divided by frames per second (fps). Finding the distance requires mapping the pixel coordinates to the real world coordinates. There were several mechanisms for establishing this relation, manually collecting distance between two points that are being projected in the image, estimating the scale ratio based on the various objects' real-world dimensions and pixel dimensions, by calculating the intrinsic and extrinsic parameters of the camera based on vanishing points[2], etc. Of all these methods, calibrating the camera based on vanishing points is shown to be effective and requires little or no manual intervention.

[1] http://cocodataset.org/explore

[2] A vanishing point is the intersection of several parallel lines in an image.

We calculate the vanishing points of the frames in the video feed based on Dubská method [6]. This method is shown to work in multiple geographic locations.

1.1 Purpose of this work

Road traffic forecasting is one of the major processes in everyday operations of a city or a state. The government agencies responsible for the smooth functioning of road networks require huge amounts of data for accessing the on-road situations and making necessary accommodations.

Our work provides an additional source of information regarding traffic flow without any manual intervention. This automatic process calculates the number of vehicles passing through a point along with their speed based on the live video feed from internet connected traffic cameras. Although there were existing methods which were able to collect this data for both the incoming and outgoing vehicles simultaneously, we are presently focused on collecting the data about vehicles moving towards the camera.

1.2 Structure of this work

Initially, we will go through a brief description of techniques, methods, and technologies that were used in this work in the next chapter. Similar methods which were implemented towards the goal of our project will be explained in the related work chapter. We later on, explain different techniques and algorithms used in this work and conclude with the results of our work.

Chapter 2

Background Work and Related Research

2.1 Background Work

In this section, we will go through a brief description of the procedures and technologies that are being used in this work.

2.1.1 Background Subtraction

Background subtraction or Foreground extraction is the process of differentiating the new objects from the existing background of a given image scene. In this process, each video frame is compared with an existing background model which is updated frequently based on the gradient of pixel values. In this problem domain, identifying an ideal background model which can be applied for every location is highly unusual. There were several problems such as sudden illumination changes, moving shadows, slow-moving vehicles, stopped vehicles and uncontrolled disturbances due to camera pole movements by air, etc.
Several techniques were proposed such as temporal mean filtering, min-max inter-frame differencing, methods based on Gaussian mixtures, etc [7]. Out of all the available techniques, the background subtraction based on Gaussian mixture models proposed by Z.Zivkovic in [8] have been shown to be effective and was highly accepted.

2.1.2 Deep Neural Networks (DNN)

Deep neural networks are an advanced implementation of artificial neural networks. DNNs have been widely used in image processing, video processing, natural language processing, speech recognition, and bioinformatics. In certain applications, some network architectures

have proven to surpass human accuracy standards. Our purpose of using DNN in this work is for recognizing vehicles on road. We use this knowledge to detect vehicles on the road. This process is easier and robust compared to traditional blob detection mechanisms. Some deep neural networks capable of object detection are explained below.

2.1.3 Faster R-CNN

Faster R-CNN network is an advanced regional proposal network. R-CNN is a Regional Proposal Network which depends on region proposal algorithms such as Selective Search[1] and Convolutional Neural Network[2] (CNN) showcasing great success in object detection but they are computationally expensive [3]. Faster R-CNN was able to cut down the heavy computational costs and achieve near real-time test predictions by using the shared convolutions from region proposals. This network is capable of taking an image of any input size and output the proposed object regions as a bounding box. Faster R-CNN has been trained and tested on multiple image datasets such as COCO and PASCAL VOC[3]. As a whole, this neural network attained remarkable accuracies and was trained end-end on several classes.

2.1.4 Mask R-CNN

Mask R-CNN is an object segmentation network developed by researchers at Facebook [9]. This network is an extension to Faster R-CNN. It makes use of the bounding box predictions from Faster R-CNN to provide a pixel level instance segmentation. Similar to Faster R-CNN, this is network is also trained on COCO dataset. The pre-trained network is made available by the author and we make use of this pre-trained network to detect segmentation masks of

[1]Selective Search algorithm is based on computing hierarchical grouping of similar regions either based on color,size or shape.

[2]CNN are a class of deep neural networks which are a regularized version of fully connected neural networks.

[3]PASCAL VOC is a dataset of images similar to COCO. Further details can be found at http://host.robots.ox.ac.uk/pascal/VOC/

vehicles on road. The purpose of these segmentation masks will further be explained in the later chapters.

2.1.5 Camera Calibration

For estimating the speed of vehicles on road, their movements are to be tracked over multiple frames and these movements are converted to real-world distances. Some methods involve manually collecting the actual distance between points in the image, some methods use special depth sensors to find the distance from the camera to the object and some methods are based on camera calibration [10]. The methods which require manual intervention or requires a special apparatus increases the cost and reduce scalability. With camera calibration, a relationship will be established between the image plane and world coordinate system, the intrinsic and extrinsic parameters of the camera need to be calculated based on the pin-hole camera geometry system. The intrinsic parameters such as focal length, principal point, and skew factor establish a relation between the image plane and image coordinate system whereas the extrinsic parameters such as rotation and translation map the image coordinate system to the world coordinate system. Researchers have been studying camera calibration for a long time, some techniques are based on known intrinsic parameters and some involve multiple images of the same scene from different views. These methods [11, 12, 13] achieve camera calibrations based on the known real-world distances such as the length of a road stripe.

2.1.6 Vanishing Point Estimation

A vanishing point(VP) is a point in the image plane where mutually parallel lines in three-dimensional space appear to intersect. Vanishing point estimation is performed using the method proposed in [6]. In this method, the orthogonal vanishing points are detected based on the trajectories of moving objects. The first vanishing point lies in the direction parallel to the vehicle trajectories. Feature points from each frame are initially detected based on the

minimum eigenvalue algorithm[4]. These feature points are tracked over subsequent frames and the points possessing a significant movement are considered to be the part of moving vehicles. These points are mapped to parallel coordinates system where each point is represented as a line connecting two parallel axes. These pixel points are extended as lines to infinity and are assumed to be intersecting at a vanishing point. A modification of Cascaded Hough transform which was proposed in [14] is used to estimate the vanishing point through a voting mechanism in a pairwise mapping space called the diamond space [5]. Estimating the first VP is considered to be the most stable compared to other VPs.

As the first vanishing point was calculated, the second vanishing point lies in the direction perpendicular to the object trajectories or otherwise parallel to the road plane. The diamond space voting process is again used in estimating the second VP but instead of random feature points, the edges of the moving vehicles are used. These pixel points which lay on the edges of the vehicles can be extracted through canny edge detection algorithm[6] which further is based on the background subtraction. Pixel points of the edges of a foreground blob are mapped to the parallel coordinate system and all the points which are closer to the first VP are eliminated from the voting process. The remaining edge points are used in the voting process to detect the second vanishing point.

The third vanishing point lies in the direction perpendicular to the first two VPs. It can be calculated based on the first two orthogonal VPs and assuming the principal point(optical center) lies in the center of the frame and calculated focal length.

[4]It calculates the gradient of pixel values in the image and ranks them using the eigenvalues of the image matrix.

[5]Diamond space is formed by combining all the transformations of the parallel coordinate system. A transformation on the parallel coordinate system results in the change of direction of axes.

[6]Canny edge detection algorithm finds the gradients of the image, then applies non-maximum suppression to remove spurious edges and choosing important edges based on threshold.

2.2 Related Research

In this section, we will discuss different approaches for traffic data extraction and camera calibration.

2.2.1 Video Analysis of Road Traffic Videos

In this section, we discuss some of the existing pipelines in this domain. We initially provide a brief description of the work in [15]. This work provides a two-stage process for extracting road traffic information. The initial stage involves camera calibration based on the vanishing points chosen manually from the image in order to calculate the intrinsic parameters such as focal length. Once the vanishing points are chosen the homography matrix[7] is calculated which maps the image plane to a rectified plane, this process is called affine rectification. The second part of the camera calibration process involves calculating the scale factor by finding the pixel distance for a road stripe at two locations in the rectified image. A similar process is followed for finding the pixel distance for the width of a lane. Mask R-CNN network is used for the purpose of object detection which recognizes vehicles at different scales. The bounding boxes are used in tracking and speed estimation. Tracking is performed by Simple Online Real-time Tracking algorithm (SORT) [16], it uses the Kalman filter to estimate the dynamics of target vehicles with a linear Gaussian state-space model. The bottom coordinates of the bounding box are used in the tracking process. This work also uses DeepSORT[8] tracker which is an another online tracker with competitive performance compared to SORT [17]. A comparison between SORT and DeepSORT shows that the SORT tracking algorithm has less error rate compared to the DeepSORT. This work is represented in computer vision and pattern vision workshop for NVIDIA AI city challenge, 2018. The performance of this pipeline stands in top 7th position in the AI city challenge. We compare our pipeline performance

[7]Homography matrix is used to map a point in an image plane to a rectified plane. This is used for image rectification with know orthogonal vanishing points.

[8]DeepSORT algorithm is an extension to the SORT algorithm with added deep association metric.

with this pipeline's and the comparisons are reported in the results chapter.

The second pipeline we are going to discuss is developed by authors of [6] as an extension to their camera calibration process. As part of this work the cameras are calibrated based on vanishing points and the vanishing points are used in constructing 3d bounding boxes around the vehicles, speed estimation, vehicle classification based on the 3d boxes, etc. Vehicles are detected from a frame by a continuously updated background subtraction model and edge detection model based on Hough transform [18]. Once the objects are detected they are tracked in the upcoming frames and the speed calculations are performed. The major problem of this process is the vehicle blob detection mechanism which is based on traditional object detection techniques consists of several problems such as shadows, occlusion, changes in the illumination conditions, etc. For these reasons the background subtraction model has to be manually tuned and this further causes problem when it comes to scalability.

2.2.2 Camera Calibration

In this work, the authors calculate the vanishing points based on the activity map. The activity map of road video consists of location and intensity of vehicle movements, thus eliminating inactive lines from the process [19]. The line structures are recognized from the activity map using Canny edge detector and these lines are used along with Hough transform[9] to recognize the vanishing point that exists in the direction parallel to the movement of vehicles. Road lines are drawn from the vanishing point to the end part of the frame, averaging the activity map values along these lines helps to create a one-dimensional vector which is mapped to a histogram. The peaks of histogram represent high movement activity which will be in the middle of the lane and valleys represent the boundaries of the lane. The road lanes thus recognized are used in calculating the second vanishing point which exists in the orthogonal direction of the road lanes. Perpendicular lines are drawn from the road lanes and the point of their intersection is calculated based on least squares method [20]. This

[9] Hough transform is a feature extraction technique which is used to find the instances of objects using voting procedure.

intersection point is considered as second vanishing point which lies on the Horizon line along with the first vanishing point. [21] The camera parameters such as focal length can be calculated from calculated vanishing points and with the known width and distance between the camera and road, the angle of the camera w.r.t to the road and angle of tilt of the camera can be calculated. With these known parameters, the scale factor will be calculated based on the known height of the camera and distance of the camera from the road boundary. This process calculates the calibration parameters and speed with 90 percent accuracy, but it requires scene specific information such as the distance of the camera from the road, etc. And one more major problem is that estimating the vanishing points based on the activity map is feasible only with roads that have a high number of lines and heavy traffic flow.

Chapter 3

Data Gathering

Research on traffic video analysis requires a variety of video datasets. Our aim is to build a robust pipeline which can be applied in a wide variety of locations with unique geospatial constraints, this requires a large variety of traffic videos. Collecting the data was made possible by Project Open Data[22]. The aim of this project is to implement principles of transparency, participation, and collaboration of several government agencies to increase the accessibility of information to the public in open machine-readable formats. Under this project, the federal government built an online platform(Data.gov) to publicize several federal datasets and to facilitate other government agencies in doing so. This helped us in gaining access to historical and live feed data of several cameras on interstates.

3.1 Iowa State Open Data Catalog

Under the open data project, Iowa State was one of the early few states to publicize their data in education, transportation, health, economy, etc. This catalog provides several kinds of road traffic data such as weather, traffic flow, construction sites, accidents, etc. Of all these, we are highly interested in traffic sensor data and video feeds. Historical video data will be provided by sending a request form, but the time limit for the videos is only for the last three days. In order to obtain a variety of videos, this time limit is not very helpful but they provide a live feed of both sensors and videos through an API endpoint which can be used to download the required data in required time frames.

The accuracy of the proposed pipeline can be calculated with reference to the ground truth from the microwave sensors. We group the cameras and sensors by calculating the distance

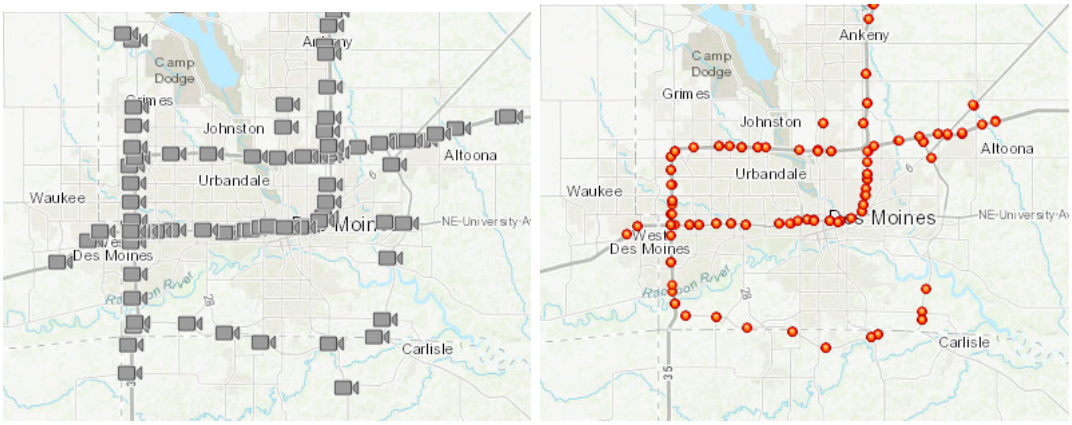

Figure 3.1: Screenshots from [1, 2] showing the location of cameras and sensors installed in Des Moines, IA.

between them based on their GPS coordinates. The metadata for the videos consists of an unique identifier, GPS coordinates, description of its location, the direction in which the camera is facing, etc. Sensor live feeds also consist of similar fields along with traffic counts, average speed, total occupancy, etc.

Chapter 4

Pipeline for Information Extraction

This chapter explains the entire process from pre-processing to the final data collection.

4.1 Image Noise Filter Creation

In this chapter, we will discuss the process of creating the mask for filtering the incoming vehicles from the rest of the video noise. It uses background subtraction and optical flow to determine the boundaries of incoming vehicle movements. This process is proposed in a way to remove any manual intervention and to achieve scalability.

4.1.1 Pixel selection through background subtraction

In this process, we train the background model on a video for a certain time(usually two minutes) and then start passing the new videos to the model to find the pixel indices of foreground objects. Once the pixel positions are found, their movements are tracked across several frames to determine the pixels that are moving towards the camera and away from the camera. The positions of these pixels are recorded throughout the video, on completion of the entire video we have the clusters of pixel positions.
The vehicle blobs extracted by the object detection process can be categorized to either incoming or outgoing based on their pixel positions and the masks.

4.2 Automatic Camera Calibration

Camera calibration, it is the process of mapping a relationship between the image plane and the world coordinates.

Figure 4.1: Original Frame

This image is frame snapshot of a video downloaded from IOWA DOT traffic cameras.

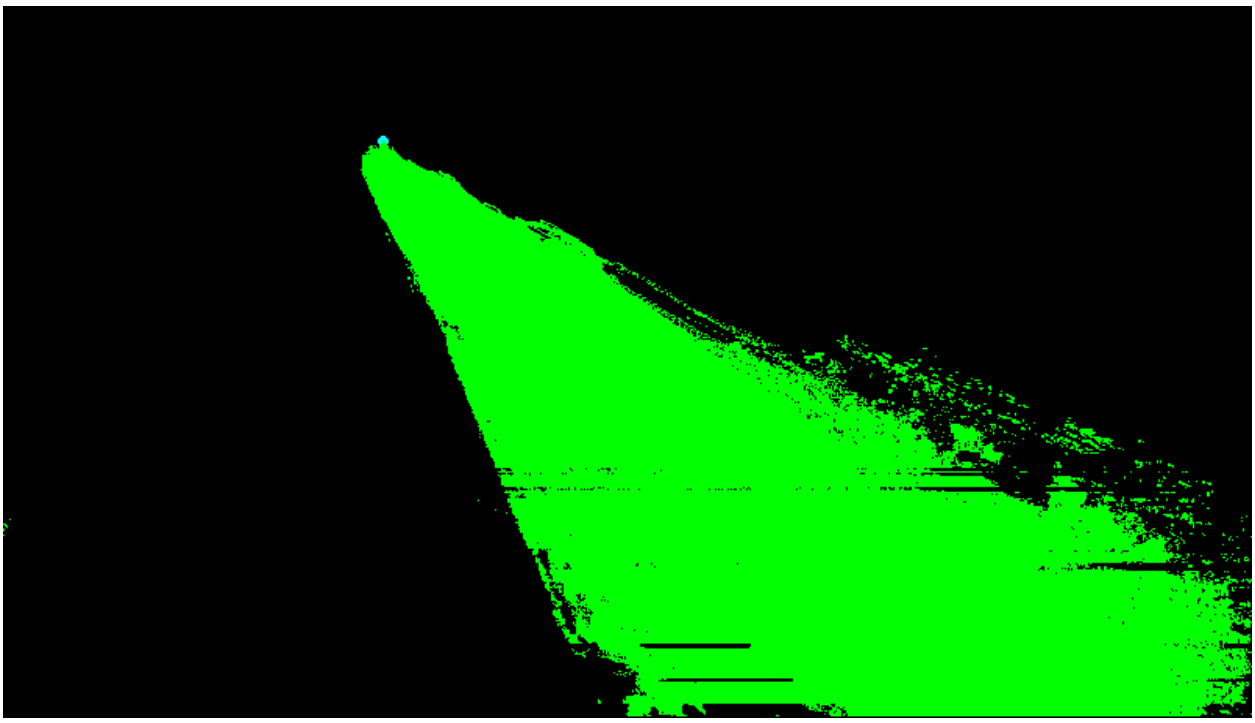

Figure 4.2: Image Filter created from the Video.

The figure shows the cluster of pixel points. These points are the positions of pixels consisting of incoming traffic.

4.2.1 Vanishing Point Estimation

Our process involves estimating the orthogonal vanishing points through the technique proposed by Dubska in [6]. The vanishing points tend to become stable with at least four minutes of video at 15 fps. The calculated vanishing points along with the principal point which is assumed at the center of the frame will be used to calculate the focal length and the third vanishing point as provided by [19]. For U,V are the vanishing points and P is the principal point in the image plane, focal length f can be calculated as

$$U = (u_x, u_y) \quad V = (v_x, v_y) \quad P = (p_x, p_y)$$

$$f = \sqrt{-(U - P) \cdot (V - P)} \tag{1}$$

The calculated focal length along with the known vanishing points can be used to calculate the world coordinates of the third vanishing point.

$$U' = (u_x, u_y, f) \quad V' = (u_x, u_y, f) \quad P' = (p_x, p_y, f)$$

$$W' = (U' - P') \times (V' - P') \tag{2}$$

where U', V', W' are the world coordinates of the vanishing points and P' is the world coordinate of the principal point. Sample screenshots from this process are shown in Figure 4.3, the lines represent the directions of orthogonal vanishing points.

4.2.2 Mapping Image Plane to World coordinates

This process along with the vanishing points require calculating the normal vector corresponding to the ground plane. [23] The normal vector can be calculated based on the third vanishing point and the focal length. From equation (2), the world coordinates of the third vanishing point represented as W' and that of principal point as P' can be used to calculate the normal vector of the ground plane as $W' - P'$. The distance between the ground plane and the camera is unknown, so an arbitrary value is chosen. The actual distance can be corrected using the scale factor but as we are interested only in the relative distances we

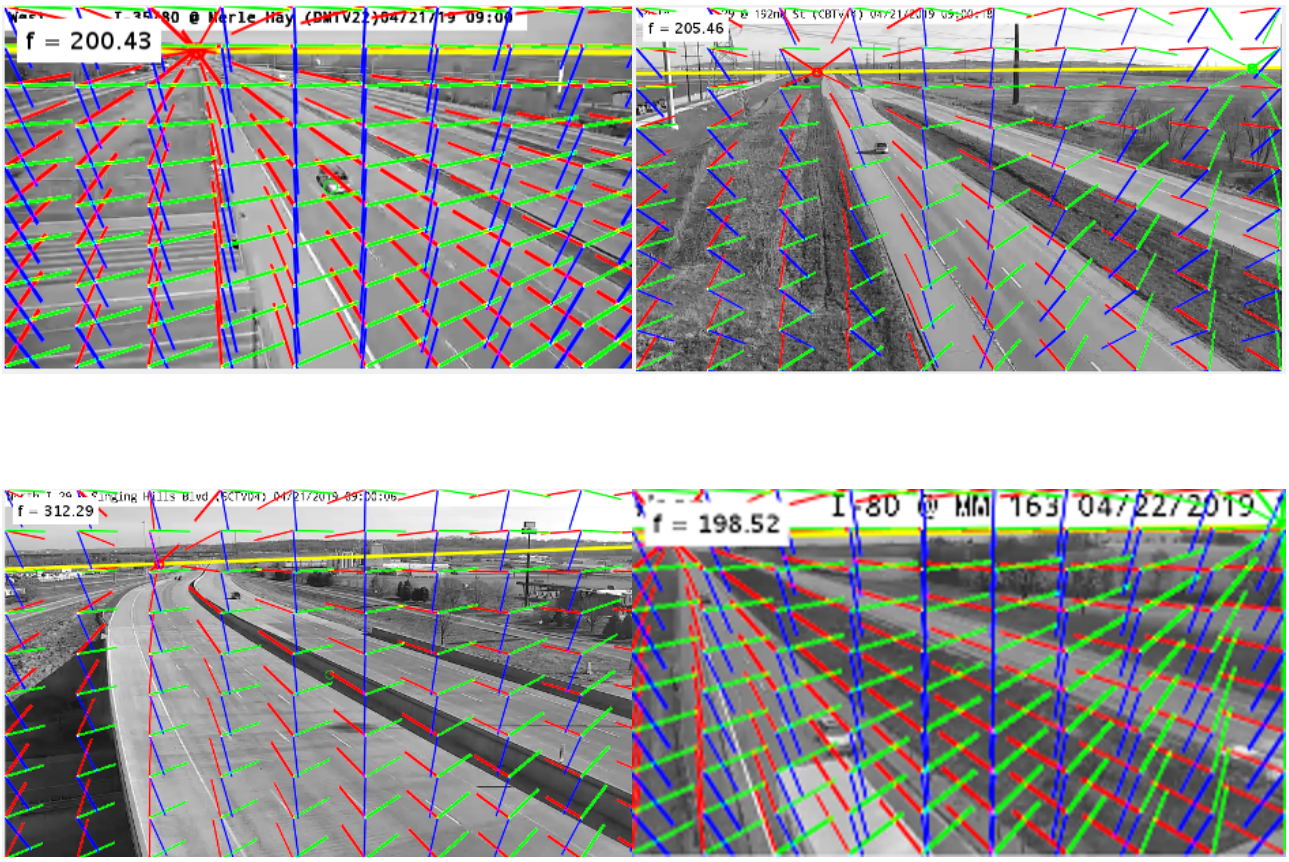

Figure 4.3: Screenshots from VP estimation at different locations.

Red color lines represent the VP in the direction of vehicle movement, green represents the VP orthogonal to the movement direction and parallel to the road plane. Blue lines represent the third VP perpendicular to the road plane.

will not be doing that. Based on the parameters of the ground plane and the distance d, the relative distances of the image points can be calculated as provided by [18]. For a point, (x, y) on image plane represented by X, its world coordinated can be calculated by a function w. For the road plane represented as $\wp$, principal point on image plane as O and $X' = (x, y, f)$,

$$w(X) = \wp \cap O\vec{X}' \tag{3}$$

4.2.3 Estimating the Scale Factor

The world coordinates of the points on the image plane can be calculated from the equation (3) and their distance can be calculated as the norm of their difference, but it does not have any measurement units. The relation between the distances and real-world measurements can be calculated as a ratio between the actual metric length and relative distance. For this process, we make use of the lengths of vehicles in metric units, as part of the calibration process the frames are provided as input to the Mask R-CNN which provides the pixel level segmentation of the vehicle blobs. We project a line *l1* from the first VP passing through the minimum column and maximum row value of the blob. Similarly, we project lines from the second vanishing point passing through the minimum column value corresponding to the minimum and maximum row values denoted by *w1,w2* [18]. We calculate the intersection points of *l1,w1* as P and *l1,w2* as Q. The distance r between P and Q represents the length of a car in relative distances. This can be represented based on equation (3) as below,

$$r = |w(P) - w(Q)| \tag{4}$$

We choose the average length of the car in metric units as 5 meters based on the vehicle registration records from IOWA DOT. The scale factor(λ) can be calculated as,

$$\lambda = 5/r \tag{5}$$

The average length of a car changes from one region to another based on the demographics, the above-stated number needs to be changed accordingly. Our initial idea was to use trucks

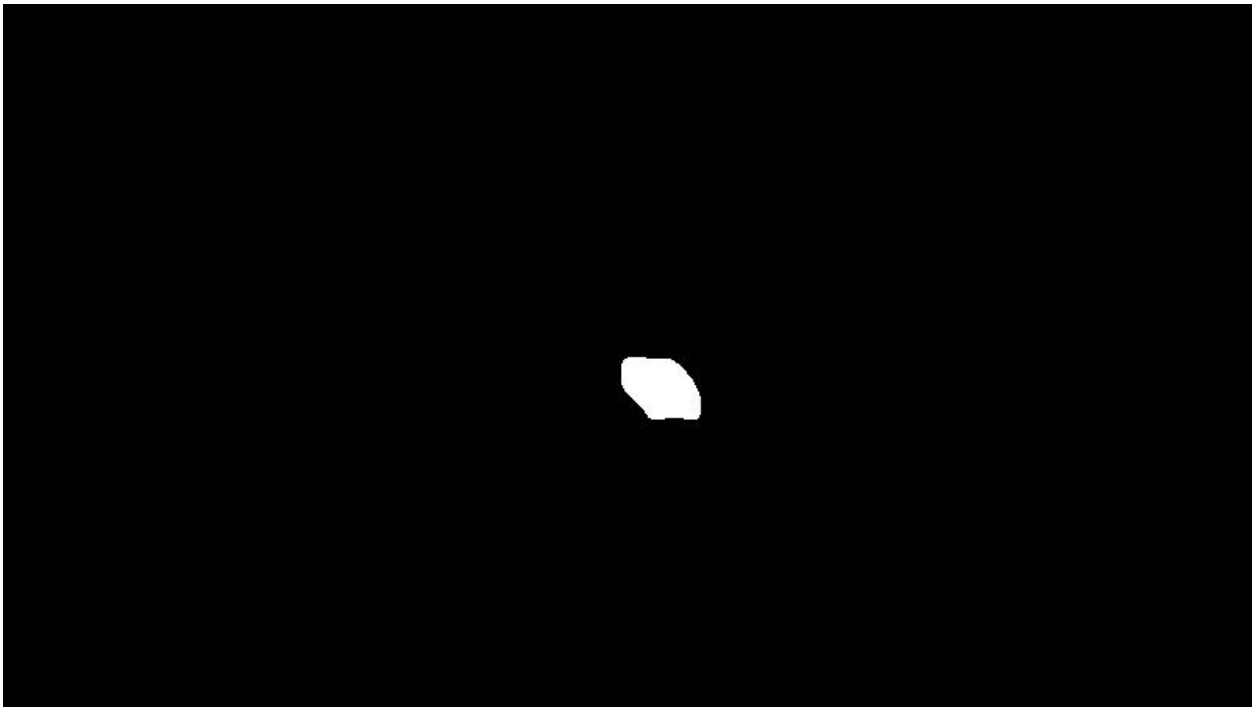

Figure 4.4: Pixel level segmentation mask of a car.

The image shows the segmentation mask of a car detected using Mask R-CNN network. As part of pre-processing, similar segmentation masks of all the cars in a video incoming towards the camera are used for scale factor calculation.

Figure 4.5: Images showing segmentation masks with lines intersecting from VPs.

These images shows the lines that are projected from the vanishing points passing tangential to the segmentation masks. The difference in lines from the two VPs is shown with the thickness of the lines.

instead of cars as their length are standardized by the federal regulations, but the segmentation masks of the trucks are affected by the position of the camera so a general ground couldn't be established. For example, the segmentation masks are better for cameras with higher relative height w.r.t to the road than the ones with less relative height.
Finally, we collect the world coordinate distances of all the cars passing through the lower third of the frame. We create a histogram of the lengths and choose the length with the highest counts. The scale factor thus calculated will be used further in calculating the vehicle traveled distance which will be provided from the tracking process.

4.3 Object Detection

Object detection is the process of recognizing real-world objects from an image or a video frame. A traditional process involves extracting low-level visual features through a sliding window process and then mapping these low-level features to form a semantic representation [24]. The semantic representations are further classified into real-world object classes such as faces, bikes, persons, etc. This process is prone to errors when dealing with unconstrained conditions in images such as varying illuminations, the scale of objects, different types of objects with similar features, etc. In order to overcome these problems, an application specific manual engineering of these pipelines needs to be performed. Convolutional Neural networks with region feature solved this problem by attaining stable accuracies for a variety of object types. Further developments of R-CNN such as Fast R-CNN and Faster R-CNN were efficient with similar stable accuracies and less computational power requirements.

4.3.1 Faster R-CNN

This network consists of two parts a Regional Proposal Network (RPN) and a Fast R-CNN detector. The regional proposal network takes an image as an input and provides object proposals with scores using feature detection algorithms and Convolutional Neural Networks.

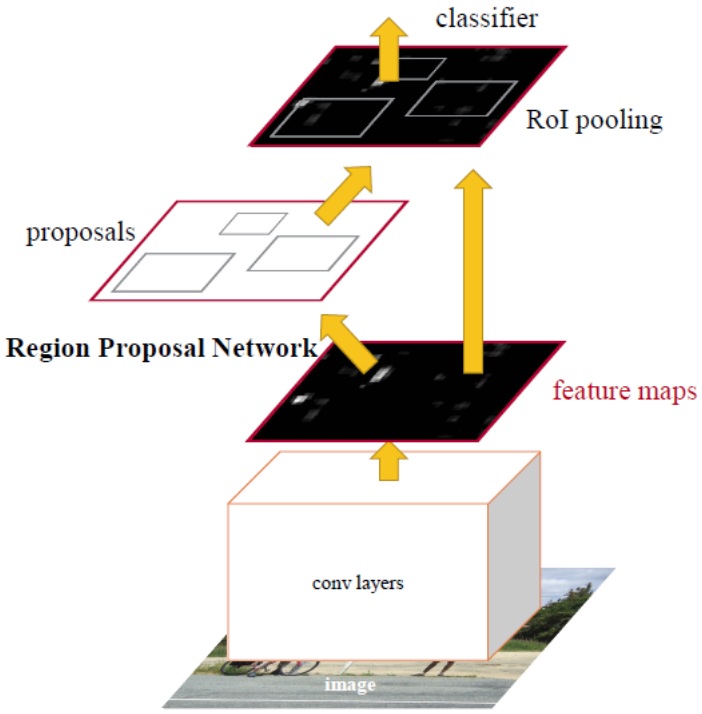

Figure 4.6: Image of Faster R-CNN applied to an image for inference. This image was provided from [3]

The image shows the object detection process of Faster R-CNN for a single image. In this process, the object proposals are created by the RPN based on feature maps and the proposals are used object detector for creating bounding boxes and classifying the objects.

The Fast R-CNN detector shares convolutions with region proposal network to output the bounding boxes of objects along with confidence scores and object classes.

4.3.2 Reason for choosing Faster R-CNN

The object detection module is an important part of this pipeline which can affect both the vehicle counts and speed calculations, so we tested our pipeline with three different deep neural networks namely Mask R-CNN, Faster R-CNN, and Single Shot Detectors(SSD). Our first trail was Single shot detectors, SSDs doesn't involve region proposals which makes them computationally less costly compared to Faster R-CNN and Mask R-CNN. But the object

detection fails in videos with a wide angle where the ROI of vehicles are small and are not recognized until they are very close to the camera. This makes it harder to calculate speed. Mask R-CNN which provides pixel level segmentation of vehicles along with the bounding boxes showcased marginally better detection rate than the Faster R-CNN but the computational cost of the pipeline increases drastically when compared to other networks, the program keeps running out of memory due to limitations in the availability of better computational power. With these reasons stated Faster R-CNN becomes a more suitable option for this application than the rest bearing in mind the detection rates and available hardware.

4.3.3 Implementation Details

We make use of the Faster R-CNN with Resnet-101[1] backbone which is pre-trained on COCO image dataset. This pre-trained network is made available though Tensorflow Object Detection API. The video frames are provided as input to this pre-trained network which provides bounding boxes of the objects detected on roads. We filter the bounding boxes based on its class and choose those the detections which have a confidence score of more than 60 percent and its class belonging to either cars, trucks or buses. We apply Non-Maximum suppression [2] to remove duplicate bounding boxes of the same object with more than 50 percent overlap.

4.4 Object Tracking and Speed Estimation

4.4.1 Object Tracking

Pyramidal implementation of the Kanade-Lucas-Thomashi(KLT) tracker is capable of tracking fast moving objects and has less computational cost compared to other trackers[25].

[1]Resnet-101 is a convolutional neural network which consists of 101 layers and is trained on more than a million images. This network is capable of classifying thousands of object classes.

[2]Non-Maximal suppression algorithm calculates the IoU of bounding boxes of same object class and filters the duplicate bounding boxes for which the overlap is more than a threshold.

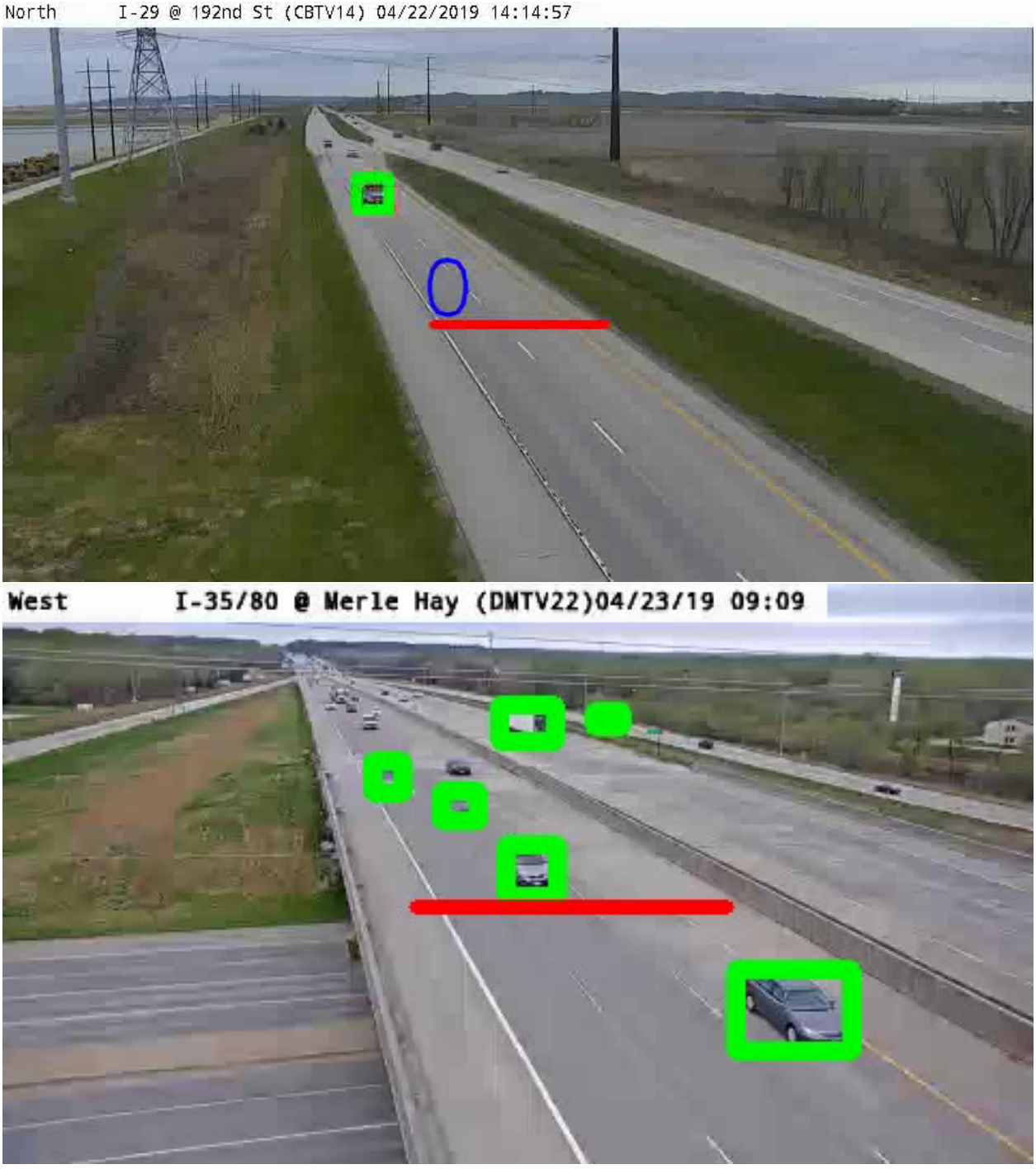

Figure 4.7: Sample images of detected vehicles by object detection process.

We find the centroids of the bounding boxes and pass them to the KLT tracker for tracking these objects over the subsequent frames. The object detector is not called for every frame, instead, it called in a certain frequency. We choose the frequency of object detector based on the fps rate which can be found from the metadata of the video. We track the objects in the frames for half a second and the tracker object is refreshed with new object detections from the object detector. This is to avoid the tracker losing track of target movements.
An imaginary line is drawn in the lower third portion of the frame for the road where the vehicles are either incoming or outgoing. The length of the line will be the width of the road which can be found from the mask created in the pre-processing step. The width of the imaginary line is chosen to be 5 pixels. The tracking process provides the history of object movements for the last half second. When the object passes through this imaginary line the vehicle count will be incremented.

4.4.2 Speed Estimation

The object detection along with the tracking process provides tracklets representing the movement of objects for every half second. The tracklets present in between the imaginary line and the bottom end of the frame are considered for speed measurement. We calculate the distance of world coordinates of the initial and final positions of each tracklet in the image plane. We then use the scale factor(λ) to measure the distance traveled in meters using equation (5). The speed can be calculated with distance divided by time(half a second in this case). We convert the speed in meters per second to miles per hour by multiplying it with 2.237 and the speeds thus found will be stored in a list. Finally the average speed will be calculated after the entire video. The sample snapshots of video frames after the object tracking and speed estimation process are shown in Figure 4.8.

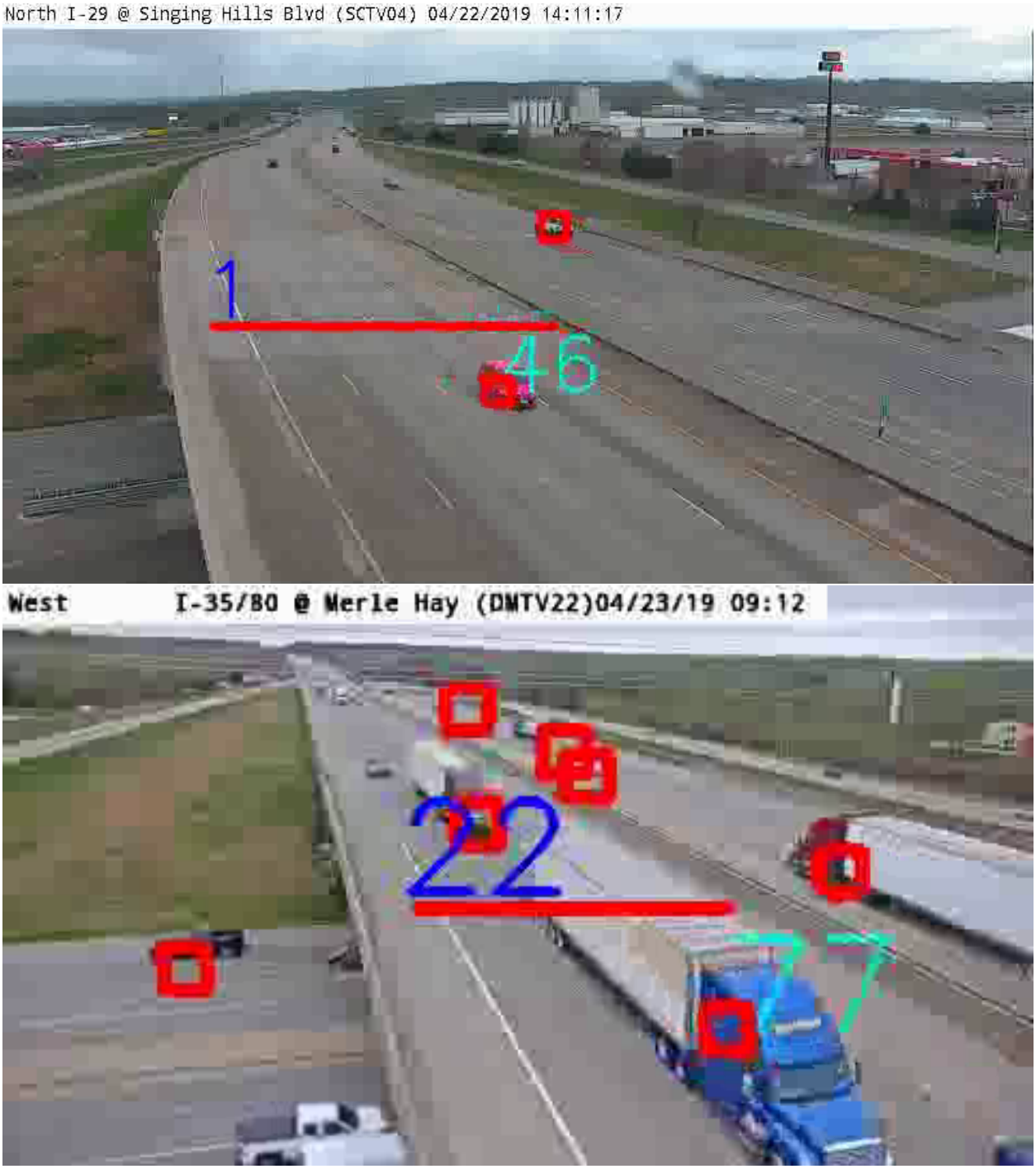

Figure 4.8: Vehicle passing the imaginary line.

The red color boxes represents the tracking window of the object tracker. The number of vehicles passing through the imaginary line is represented in blue digits. The speed of each vehicle is represented by a sky-blue color situated above its tracking window.

4.5 Pre-processing

The pre-processing step consists of mask creation, vanishing point estimation, and camera calibration. The pre-processing step has to be performed initially for every camera. The mask creation process usually takes a two-minute video, more vehicles passing through the camera location creates a better mask. Estimating the vanishing points requires at least four minutes of video for the VPs to be stable. Focal length and principal points are calculated from the vanishing points. These parameters are utilized in calculating the scale factor based on the segmentation masks from the videos.

4.6 Information Extraction

Once the initial mask creation and camera calibration process are completed, the pipeline will be ready for information extraction. New videos from the video feed are fed to the pipeline which outputs the traffic counts and average speed of vehicles.

Since surveillance is the major purpose of the cameras on road, their field of view can be changed by the camera operators with operations such as rotation, zooming in, zooming out, etc. The pre-processing needs to be re-initialized for creating masks and scale factors according to the new camera scene.

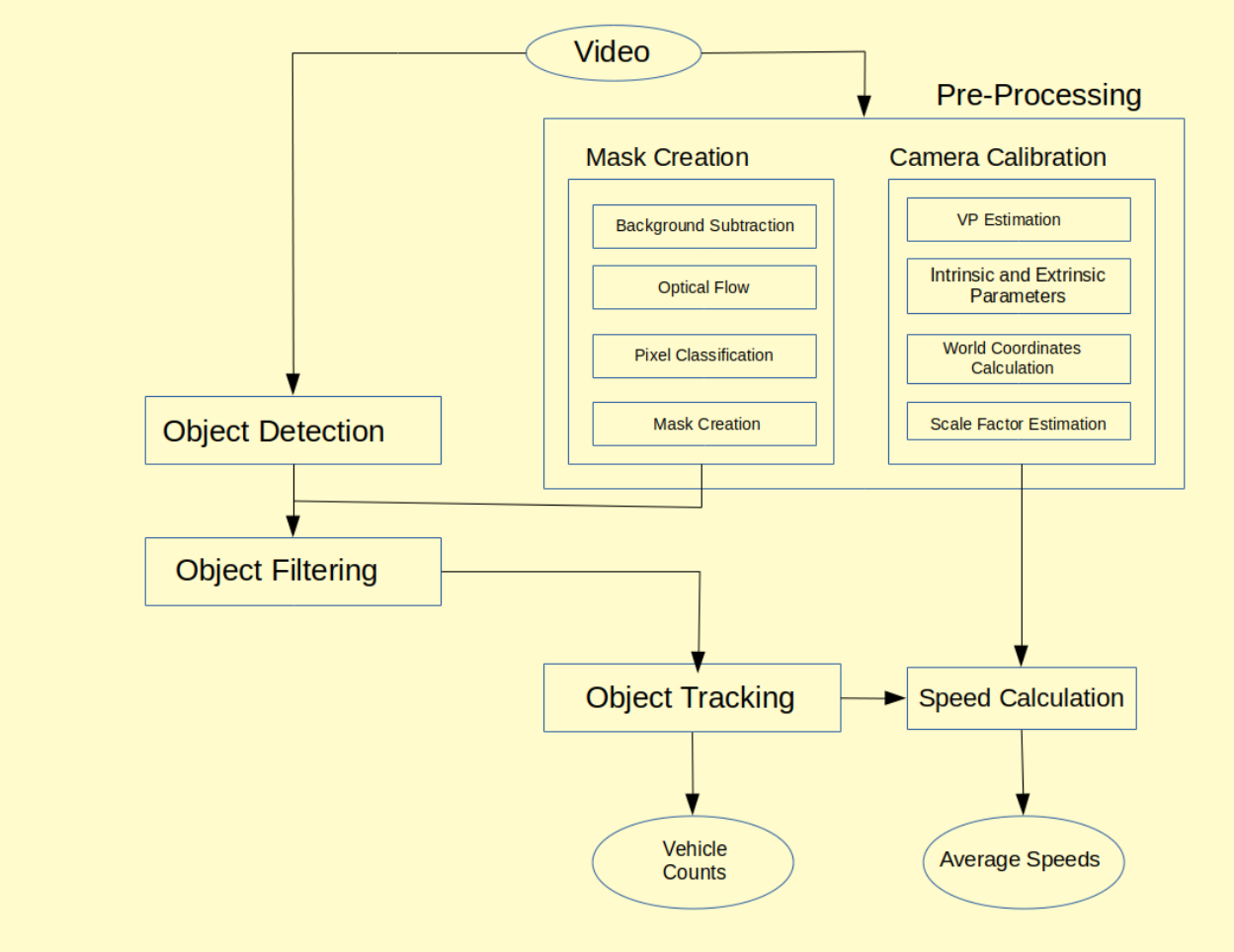

Figure 4.9: End-end pipeline image for information extraction.

The overall process of data extraction is shown in this image. The pre-processing step consists of image noise filter creation and camera calibration. New videos from the same camera are directly passed to the object detection module and further on for data extraction.

Chapter 5

Results

This chapter shows the performance of the proposed pipeline. Performance evaluation relating to multiple video types, comparisons with an existing deep learning based approach and the computational times will be provided.

5.1 Performance of the Proposed Pipeline

The predicted vehicle count and average speed from the pipeline are compared with data collected from the microwave sensors. Results from four groups of videos are shown. Each group consists of ten videos from the same camera, which share similar camera parameters. Group 1 and 2 consists of low-resolution and groups 3 and 4 consists of high-resolution videos. The best and worst performance of the pipeline are 3.6 miles per hour and 7.47 miles per hour. The overall RMSE of the pipeline is 5.23 miles per hour. Percentage of error for the predicted counts of vehicles for each group are provided in Table 5.1. A bar chart plotted for error rates is shown in Figure 5.1.

Group	RMSE (speed)	Counts error(%)
Group1	7.4706	11.3
Group2	6.0592	8.26
Group3	3.6216	5.4
Group4	4.2829	6.07

Table 5.1: Error rate of predictions compared with sensor data.

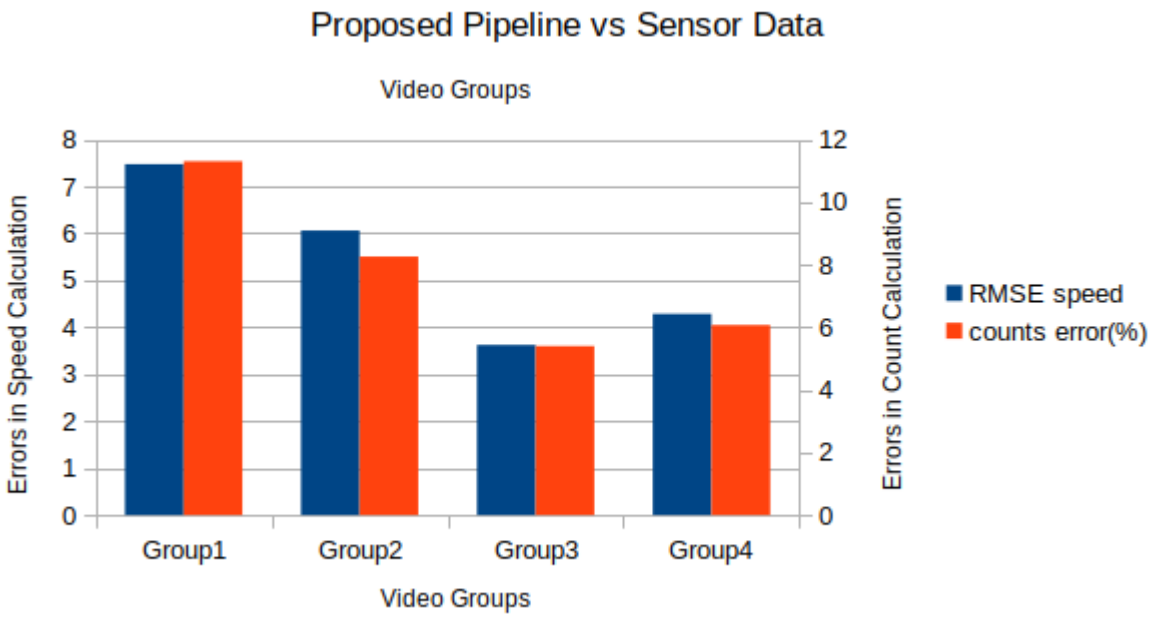

Figure 5.1: Error rate of multiple groups of videos compared with sensor data.

The blue color bars represented with respect to the primary y-axis shows the RMSE in speed estimation. Similarly, the percentage of count error is represented by the orange bar with respect to the secondary y-axis. The x-axis represents the groups of video, where all the videos in a group share same camera parameters.

5.2 Comparison of Performance with Videos of Different Resolution

The resolution of video effects the performance of the pipeline. The low-resolution videos in our dataset are of 480x250 resolution and that of high-resolution is 800x450. Average RMSE speed of low-resolution videos is 6.87 miles per hour and that of the high-resolution video is 4.03 miles per hour. In case of the percentage of errors in predicted counts is 10.6 percent for low-resolution videos and 5.6 percent for videos with high resolution. The results of this comparison are plotted in the bar graph and shown in Figure 5.2.

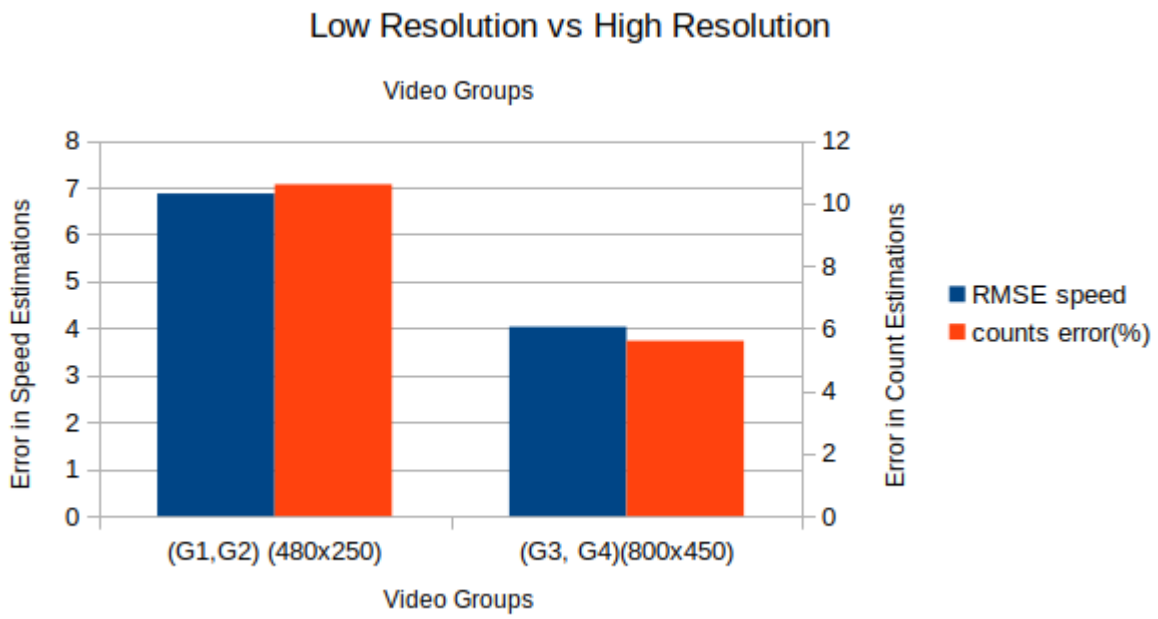

Figure 5.2: Error rate in Low-resolution videos vs High-resolution videos.

The comparison of the proposed pipeline's performance with videos of different resolutions. The primary y-axis represents the errors. The secondary y-axis represents the percentage of error in vehicle counts.

5.3 Performance Evaluation with Existing Pipeline

We compared our work to a similar deep learning pipeline which was based on Mask R-CNN and follows a semi-automatic procedure for calibrating the cameras [15]. This existing

pipeline was described in the related research chapter. We tested this pipeline on IOWA DOT dataset with distance points chosen manually. We report the comparison of performances of the two pipelines tested on five different videos. The RMSE of speed on these five videos for the proposed pipeline is 6.2 mph and that of the alternative pipeline is 4.51 mph. The predictions from the proposed pipeline and the existing pipeline are provided in Table. 5.2

Video	Proposed Pipeline speed	Existing Pipeline Speed	Ground truth
7806	69.0498	70	72
7120	69.8220	69.5	71
7707	61.0861	63	72
8168	64.8305	65	66.5
7613	60.1187	64.5	68

Table 5.2: Comparison of Proposed and Existing pipelines.

5.4 Computational Time

As part of this project, we have used two systems one with Nvidia K20 GPU which has an approximate memory of 5.6 Gb. The average speed of computation is 7.4 frames per second. Our second device is a virtual machine in Microsoft Azure with Nvidia K80 GPU. The Nvidia K80 has more cores and approximate memory of 12 Gb. We have used this machine for testing the proposed pipeline and existing pipeline. The computational time of our proposed pipeline is 15 frames per second whereas for the existing pipeline it is approximately 1 frame per second. For a video with the frame rate at 30 fps and video time of 2 minutes, the proposed pipeline takes up to four minutes for execution, whereas existing pipeline takes up to an hour for execution.

Chapter 6

Conclusion and Future Work

In this chapter, we will summarize our overall research and discuss our goals and results.

6.1 Conclusion

Our major goal of this project is to extract road traffic information from videos automatically. We have discussed the problems of location-specific camera calibrations along with the advantages of an automatic camera calibration process. We have discussed our approach from end-end explaining various techniques that were used in this process.

With the proposed pipeline, we have showcased a process which requires very little or no manual intervention in the data collection process. The overall error rate in speed estimation is 7 percent compared to the sensor data. Similarly, the overall error in vehicle counts is 8.9 percent.The overall error of the sensor systems for counts is 2.4 percent and for speed it is 1.20 percent as reported by [26], this data was the only available source closest to the ground truth. In comparison with the existing pipeline, we propose a trade-off as the difference in error is less than 2 mph and the computational time of the proposed pipeline is 4 minutes which is way faster than existing pipeline with computational time close to one hour.

Our existing dataset consists of videos from various cameras which are located at multiple locations. As these videos contribute very few data points about a particular location, we couldn't create a forecasting model based on the available data. Due to the existing time constraints, collecting sufficient data for use in forecasting has been pushed to future work. We thus conclude our work with the proposed pipeline for extracting traffic data automatically from road traffic videos which can be used for traffic forecasting.

6.2 Future Work

Collecting abundant data based on the proposed pipeline for creating a traffic forecasting model. Thus collected data can be used along with the sensor data.

Data collection based on the proposed work consists of significant time lag compared to real-time data feeds. This time lag effects performance of real-time forecasting models. A potential solution to this problem is fine-tuning the object detection network which involves using lightweight deep neural networks such as Single Shot Detectors which are specifically trained on traffic camera data.

Further developments of this work might involve increasing robustness in various weather conditions, extracting more information from traffic videos such as different types of vehicles, information on lane switching, identification and re-identification of the same vehicle from multiple cameras, etc.

Bibliography

[1] IOWA DOT. Arcgis portal camera view. http://www.arcgis.com/home/webmap/viewer.html?url=https://services.arcgis.com/8lRhdTsQyJpO52F1/ArcGIS/rest/services/Traffic_Cameras_View/FeatureServer&source=sd, 2019.

[2] IOWA DOT. Arcgis portal sensor view. http://www.arcgis.com/home/webmap/viewer.html?url=https://services.arcgis.com/8lRhdTsQyJpO52F1/ArcGIS/rest/services/Sensors_View/FeatureServer&source=sd, 2019.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[4] Torin Monahan. "war rooms" of the street: Surveillance practices in transportation control centers. *The Communication Review*, 10(4):367–389, 2007.

[5] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[6] M. Dubská, A. Herout, R. Juránek, and J. Sochor. Fully automatic roadside camera calibration for traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1162–1171, June 2015.

[7] Y. Benezeth, P. M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *2008 19th International Conference on Pattern Recognition*, pages 1–4, Dec 2008.

[8] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773 – 780, 2006.

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

[10] Wang Qi, Fu Li, and Liu Zhenzhong. Review on camera calibration. In *2010 Chinese Control and Decision Conference*, pages 3354–3358, May 2010.

[11] J. Bazin and M. Pollefeys. 3-line ransac for orthogonal vanishing point detection. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4282–4287, Oct 2012.

[12] H. Dinh and H. Tang. Camera calibration for roundabout traffic scenes. In *2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 674–677, Aug 2012.

[13] J. Douret and R. Benosman. A multi-cameras 3d volumetric method for outdoor scenes: a road traffic monitoring application. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 334–337 Vol.3, Aug 2004.

[14] Adam Herout (Brno University of Technology) Markéta Dubská (Brno University of Technology). Real projective plane mapping for detection of orthogonal vanishing points. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[15] Amit Kumar, Pirazh Khorramshahi, Wei-An Lin, Prithviraj Dhar, Jun-Cheng Chen, and Rama Chellappa. A semi-automatic 2d solution for vehicle speed estimation from

monocular videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

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

[17] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, Sep. 2017.

[18] Marketa Dubska, Adam Herout, and Jakub Sochor. Automatic camera calibration for traffic understanding. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[19] T. N. Schoepflin and D. J. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, June 2003.

[20] John W. Lorimer. The least-squares intersection of a family of lines and its application to phase equilibria. *Canadian Journal of Chemistry*, 60:1978–1981, 02 2011.

[21] D. J. Dailey and L. Li. An algorithm to estimate vehicle speed using uncalibrated cameras. In *Proceedings 199 IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (Cat. No.99TH8383)*, pages 441–446, Oct 1999.

[22] TODD PARK and STEVEN VANROEKEL. Project open data. `https://obamawhitehouse.archives.gov/blog/2013/05/16/introducing-project-open-data`, 2013.

[23] Z. Zhang, T. Tan, K. Huang, and Y. Wang. Practical camera calibration from moving objects for traffic scene surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(3):518–533, March 2013.

[24] Zhong-Qiu Zhao, Peng Zheng, Shou tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 2018.

[25] Ki-Sang Kim, Dae-Sik Jang, and Hyung-Il Choi. Real time face tracking with pyramidal lucas-kanade feature tracker. In Osvaldo Gervasi and Marina L. Gavrilova, editors, *Computational Science and Its Applications – ICCSA 2007*, pages 1074–1082, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[26] Hemin Jalal Mohammed Mohammed. Evaluating the accuracy of speed and volume data obtained via traffic detection and monitoring devices. `https://kuscholarworks.ku.edu/bitstream/handle/1808/19417/MOHAMMED_ku_0099M_14071_DATA_1.pdf?sequence=1&isAllowed=y`. Master's Thesis, University of Kansas, 2015.

Appendix A

Developer Guidelines

This section explains the implementation details of this work.

A.1 Environment Setup Details

This is pipeline is majorly developed in Python programming language. Tensorflow library is used for the deep learning networks that were tested in this project. Data collection was performed using Java classes developed with Java 1.8.111 version. The vanishing point estimation was performed using the Matlab source code provided by the authors of [6].

A.1.1 Tensorflow Object Detection API

Tensorflow Object Detection API is an open source deep learning project provided by Google under the Apache 2.0 license. This project is part of several deep learning models open sourced by Google. The Object detection project consists of several pre-trained deep neural networks which are trained on multiple image datasets such as COCO, PASCAL VOC, etc. Out of all the available pre-trained networks, we make use of Faster R-CNN based on Resnet101 backbone. This network is trained on COCO dataset which consists of several classes such as humans, birds, trains, cars, trucks, etc. This network is capable of extracting vehicle blobs on the road which are used in our process.

A.1.2 Software Packages and Versions

Some of the important packages and their versions are provided in the list below.

- Python => 3.5.5
- GCC library => 7.3.0
- Anaconda => 2018.12
- Matlab => 2015 b
- Cudnn => 7.3.1
- FFmpeg => 4.0
- Numpy => 1.16
- Protobuf => 3.6.0
- Open CV => 3.4.2
- Scipy => 1.1.0
- Sk-video => 1.1.10
- Tensorflow => 1.10

A.2 Parallel Implementations

Image Noise Filter creation process of this work was also implemented in Scala language as part Scalation library. This process is developed based Opencv java API. Script for building the OpenCV library on Ubuntu OS will also be open sourced along with rest of the work.