The Pennsylvania State University

The Graduate School

Department of Computer Science and Engineering

DETERMINATION OF REAL-TIME TRAFFIC FLOW PARAMETERS IN DIFFERENT DEVICES BASED ON QOI REQUIREMENTS

A Thesis in

Computer Science and Engineering

by

Manisha Mukherjee

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2014

The thesis of Manisha Mukherjee was reviewed and approved* by the following:

Thomas F. La Porta
William E. Leonhard Professor
Director, Institute for Networking and Security Research(INSR)

Guohong Cao
Professor of Computer Science and Engineering
Director, Mobile Computing and Networking (MCN) Lab

Lee Coraor
Associate Professor of Computer Science and Engineering
Director, Academic Affairs

*Signatures are on file in the Graduate School

ABSTRACT

In present times when cameras are inexpensive and ubiquitous, processing vehicular traffic videos obtained from cameras on the road to obtain information is useful. These videos may be processed to extract information such as the volume of traffic on the road, to track particular vehicles in surveillance videos, to measure their speed in order to get traffic flow details, etc. The videos may be obtained from webcams perched on traffic signals, at the side of the road, or even from commuters who can take these videos using the camera on their cell-phones. These videos may then be processed on the devices recording them, to extract the required information. However, real-time video is usually generated at 30 frames per second (fps). If the recording device cannot process frames at a speed of at least 30 frames per second, it will customize the video by reducing its frame rate and the quality of the extracted information may decrease (QoI). If the user has a QoI requirement that will not be met even by reducing the frame rate on the recording device, the video must be streamed to a more powerful server which will be able to deliver the desired QoI by processing the video at the required frame rate. However, streaming these video files take up a lot of bandwidth. At times, the desired QoI may be met by lowering the frame rate, which may allow the video to be processed locally, or streaming the video at the reduced frame rate to a powerful server if the recording device cannot process the video at the lowered frame rate. This saves bandwidth as the video was either processed locally, or streamed to a powerful server at a reduced frame rate.

The difference in the QoI arising due to the limitations in processing powers on various devices was analyzed. Videos obtained from various traffic cameras were processed to extract information about the number of vehicles passing by in a certain amount of time, the color of vehicles, the type of vehicle, and the speed of the vehicles passing by. The videos could be processed on a server (with higher processing power) or an android phone (with less processing power). If the QoI (accuracy, precision) requirement may be met on the phone, the video is

locally processed. However, when a QoI requirement may not be met on the phone, the formulated mathematical models show that less than the original frame rate may be sent to the server to meet the same QoI requirement. This work implements a system that helps decide when the video file can be locally processed to get a desired QoI compared to when the video (maybe with a lower frame rate) has to be streamed to an alternate powerful server, thereby saving many resources such as bandwidth and the battery charge on the phone.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor Dr. Tom La Porta for accepting me to the Institute for Networking and Security Research (INSR) and giving me the opportunity to join his research team at the Networking and Security Research Center (NSRC) and making me a part of the Networking Science Collaborative Technology Alliance (NS CTA). I would like to thank him for his patience, motivation, enthusiasm, and encouragement. His guidance and his faith in my abilities helped me always in making progress in the course-work, research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Masters study.

I would like to thank Dr. Guohong Cao for willing to serve on my thesis committee, and for reviewing and evaluating my research. Furthermore, I would like to thank Dr. Anand Sivasubramaniam for his encouragement and guidance regarding course work, academia and life in general.

I would like to thank INSR members: Bhaskar Prabhala, Simone Silvestri, Scott Rager, Matt Deering, James Edwards, Brett Holbert, Stefan Achleitner who advised me from time to time about the different topics, and Srikar Tati who helped me with the mathematical models. I would also like to thank Jeremy Song for helping me out whenever I was stuck with the Python implementation of the system.

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. I would also like to thank them for supporting my research.

I would also like to thank D.K. uncle, Sanju aunty, Shalini and Vedant Tiwari for becoming my surrogate family here, and for always making me feel at home despite being so far away from home. I thank Bhaskar uncle and Indrani kakima for their support. I specially thank

Sanju aunty, Indrani kakima and Chitra aunty for their delicious food at a time when I was too busy meeting deadlines to cook for myself!

I would like to thank Sanchit Khurana, for being a great support, my closest friend in State College, and for keeping me from going crazy, giving me advice, proof reading my thesis and what not! Whether academic, or personal, I could not have managed living so far away from home without his presence in my life. I would also like to thank the whole Bong group for their company and making me miss Kolkata a little less. I would like to thank Di Lu, Byron Zhaoyong Ba and Ahana Mukhopadhyay for being great roommates. I would also like to thank Claire Weaver for being such a great friend!

Finally, I would like to thank my loved ones, who have supported me throughout the entire process. I would like to thank my maternal uncle Dr. Debasish Banerjee (Mama) for dedicating his Diploma in Psychological Medicine (DPM) dissertation to me. I would also like to thank Sangita (Mamima), Drona and Oindrila (Kochi) Banerjee for their love and support. I would like to thank Dr. Jagat Jyoti (Kaka), Dr. Karabee (Kakima) for their prompt replies to my frantic Whatsapp messages and for their encouragement! I also thank Rudrani (Rudy) and Aniruddha (Rohin) for their kind support and well wishes. I thank my grandparents, Gopal Chandra Banerjee (Dadu), Maya Banerjee (Dimma), Sarbani Mukherjee (Thamma) and Shyama Pada Mukherjee (Barda) for their unending love, support and encouragement.

Most importantly, I would like to thank my sister, Annesha Mukherjee (Chhoti) for her patience with me and cheering me up whenever I felt slightly unhappy; Java (my hamster) for just being there always and being so lively; and my parents, Ajoyendra (Baba) and Mallika (Mummy) Mukherjee who always believed in me and had enough faith in me to send me so far away from home to study at Penn State and making all this possible in the first place. I will be forever grateful for your love.

Chapter 1

Introduction

Motivation and Background

In times where cameras are ubiquitous, processing the videos of traffic obtained from cameras to obtain information is useful. The videos can be processed and the following actions may be performed: detection of vehicles [1-13], tracking vehicles [3, 6, 10-16], classifying vehicles [3, 7, 9, 11, 14, 17, 18], determining the vehicle's color [19, 20], driver behavior [21] including fatigue and distraction [22], flow modelling [23], traffic density estimation [17] etc. The vehicle signature can also be found by monitoring the data for the same vehicle at different stations [24].

Before video analysis, various intrusive methods such as buried loop sensors, radars and infrared detectors [25, 26] were employed to obtain data about traffic. Single loop detectors were introduced in the 1960s [27] and are used for measuring data such as the volume [28] and lane occupancy [29]. This method however fails to measure speed (by assuming a constant vehicle length) or classify vehicles accurately. Dual loop detectors are two single loop detectors placed apart by a certain distance. As the distance is pre-determined, dual loop detectors can calculate the speed by dividing the distance between the two loops by the time taken by the vehicles to cross the two loops. Although dual loops can provide data such as vehicle speed and classification accurately, these are too few in number as they are quite expensive to set up. The cost for upgrading from a single-loop detector to a dual-loop detector ranges from $3,250 to $5,750 , which includes $750 direct cost for loop placement and $2,500–$5,000 indirect cost caused by

lane closure (Washington State Department of Transportation, WSDOT) [30]. Because of drawbacks like the high cost of lane closures ($2,500–$5,000), safety risks associated with lane closure, ability to extract limited parameters, and necessity to interpret the signals obtained from the intrusive devices, non-intrusive methods such as the analysis of video surveillance have been adopted.

Cameras are inexpensive and easy to install. They are also non-intrusive and traffic will not be disrupted during its installation. Once mounted on a traffic signal or a pole along the pavement, they can monitor multiple lanes with traffic moving in multiple directions. Real-time data may also be obtained from commuters who may be travelling in cars, or even walking along the roads by using the cameras present on cell phones.

The data extracted from the traffic videos can be used for a myriad of purposes. The count of the vehicles may be useful for traffic planning, designing, constructing, and maintaining roads. Determining the color of the vehicles from a traffic video is useful in cases where vehicles may have to be tracked. For e.g. in a bank robbery, if it is known from witnesses that the robbers have fled the scene in a red vehicle, the traffic surveillance videos from around the region of the robbery may be analyzed to track the red vehicles detected. This will narrow down the search space drastically. Extracting the speed of vehicles is important as traffic flow information may be obtained from it. Traffic congestion is a big problem globally. In the 13 largest cities, drivers now spend the equivalent of almost eight work days each year stuck in traffic, leading to a total of 3.7 billion hours of travel delay in the year 2003. This in turn led to the wastage of 2.3 billion gallons of fuel, leading to a total loss of 63 billion dollars [31]. Earlier, traffic congestion could be solved by expanding roads by building more lanes. However, with there being a limit to expansion, that is not a feasible option everywhere, hence the available roads must be used efficiently. This can be done by analyzing the different data obtained from observing the existing traffic. Real time

analysis of the data obtained can be used to provide commuters with information about less congested routes, and detect accidents on the observed roads.

Classification of vehicles, in particular trucks is an important data as [9]:

1. Trucks are heavier vehicles, have inferior brakes and have large turning radii, causing the pavement design and maintenance strategy to depend greatly on the truck volume in that area [32].
2. Truck's diesel engines emit Particulate Matters (PM), which accounts for 72% of the traffic emitted PM. PM causes myocardial infarction and respiratory symptoms [33]. Hence, for traffic safety and pollution control as well, classifying and obtaining data about trucks is important.

Quality of Information (QoI)

Previously, the traffic videos would be analyzed manually. However, this is time intensive and prone to human-induced errors, reducing the delivered QoI. Retrieving the information digitally will not be prone to those human-induced errors, thereby increasing the QoI. In addition to traffic regulation, in military applications, or in cases of toll collection, kidnapping, and robbery, extracting information from videos of traffic in real time without having to go through the videos of traffic manually is very useful. These videos may be processed locally on the devices recording them to extract the required information. Real-time video is usually recorded at 30 frames per second (fps). If the recording device cannot process frames at the speed of at least 30 frames per second, QoI [34] may decrease.

In [34], Bar-Noy *et al.,* define QoI as a multi-dimensional metric which comprises of many components such as correctness, accuracy, precision, freshness, timeliness, security, completeness and credibility. The values of all these components together constitute the quality of

the delivered information. For example, if the completeness of the data is reduced (if an image is cropped and sent), it may improve timeliness (as its file size would decrease and would be transferred faster over the network) and depending on the context, it may lead to more precision. In this dissertation, the traffic videos are compressed by reducing the frame rate. Intuitively, this should have a detrimental effect on precision, accuracy and completeness, thereby reducing QoI. However, if the size of the video decreases considerably, bandwidth is saved and timeliness of the data delivered increases, increasing freshness, thereby possibly causing an increase in the delivered QoI. This shows that QoI is a complex metric, and that there is a trade-off between the many components of QoI. Of these components, correctness, freshness, precision and security are intrinsic components as they are dependent on the information source and do not depend on the situation in which they are being used [35] (e.g. precision of the speed determined from the video of traffic is dependent of the frame-rate of the video. This is dependent on the device that is recording this video, and not in which situation this data is utilized). The other attributes are contextual as they are dependent on the situation in which they are being used [35]. If some information is available instantly, the timeliness is high. However, if it is not available for some amount of time, the timeliness of that information decreases. So timeliness is dependent on the situation in which it is used, irrespective of the source of the data. The system implemented in this dissertation deals with an intrinsic and a contextual metric of QoI, namely precision and accuracy, respectively.

To correctly understand the relation between accuracy and precision, consider a query for instance: “did at least one vehicle go by this road which was a truck in the last 30 seconds?” The system may have the adequate capacity to state that 5 vehicles have gone by the road in the last 30 seconds. This information may be accurate, as 5 vehicles may have passed by in the last 30 seconds; however, this is not precise (as it is not stated whether any one of these 5 vehicles is a truck or not). If another query is “how many cars went by this road in the last 20 seconds?” A

variety of outputs generated by systems with varying levels of accuracy and precision would be: “more than 8”, “at most 20 cars”, “8 within 5%” or, “exactly 9 cars”. The precision component may be increased so that the desired-QoI is achieved and the information is known to be correct.

Many a times, the QoI requirements may not be strict. Let the system be one that can calculate the QoI requirement from a given query. If there is a video of a highway, and the query is “is there a traffic jam on this road?” the precision required for the speed estimation is not high. Another instance of a query with a low QoI requirement may be when a user wants to know the effect of weather conditions on the traffic on a highway. In this case, the user wants to know whether the vehicles are moving slowly (say < 20 mph) or moving at the normal speed (say 60 mph). In these cases, because the precision requirement is not high, fewer resources may be taken to respond to the query. However, if the delivered QoI is higher than the requested QoI, more network resources may be occupied unnecessarily, thus increasing the delay to many other users on the network.

Depending on the desired-QoI and the approximated values of deliverable-QoI estimated by the system, the system decides whether it should process the video locally, or stream it to a server (at the same or lower frame rate depending on the deliverable-QoI estimate of the server). Streaming the video at a lower frame rate has several advantages such as saving power on the less powerful device (by off-loading the computation, as well as having to send fewer frames) and saving bandwidth while streaming the video.

To elucidate the usefulness of the system, let us consider a situation where a kidnapping has occurred. From some witnesses, it has been found that the kidnapper has sped away from the crime scene with the abducted child, in a blue car which was last seen on the highway. In order to find where the kidnapper is headed, the vehicle in which he/she escaped must be tracked. Let it be assumed that some of the roads the kidnapper took are either monitored by traffic surveillance cameras, or some commuters have taken the video of the traffic on their cell-phones, which have

the capability to capture the blue car in question. In order to find the path of the blue car, one would usually manually analyze all the surveillance videos of the surrounding areas to track the vehicle. However, not only is this time intensive, but this may also be prone to errors. The system developed in this dissertation, permits the application to ask questions such as "Is the road in the field of vision of the camera, a highway or a side road" (discussed in Chapter 5). By asking this question, and by getting the results, all the surveillance that covers highways will be in consideration from the video database. To answer this question, the device on which the video is present may or may not have the capacity to process the necessary frame rate to deliver the desired QoI. If it does have the required capacity, it will process the stored video being gathered in real time, and give as output "yes, as the average speed of the vehicles in this video is 20 mph with precision 15%, this is a side road", in which case the video would not be further analyzed (as 20 mph ± 15% will make the speed rage from 17 mph to 23 mph, which in both cases will be a side street). If the device does not possess the capacity to process the video at the frame rate required to meet the desired QoI, it will stream the video to a powerful server. Before doing so, it will calculate the frame rate required by the server to meet the application's QoI requirement. If the required frame rate is less than that of the original video (but more than the frame rate the device itself can process), it will stream the video at the reduced frame rate, thereby saving bandwidth and battery charge on the less powerful device.

After all the videos of the highways have been identified, the system can be asked a query such as: "has at least one vehicle passed by this highway that is blue?" This will not only narrow down the number of videos to be searched further, but will also efficiently track the vehicle in question.

Bar-Noy *et al.* [34] have defined QoI as QoI=*f(I,D,P,S)*, where *I* represents the information source, *D* represents the network delivery characteristics, *P* represents the transformations done to the data by the network (e.g. compression or fusion), and *S* the security

features implemented by the network. Applications that request information may have certain QoI requirements, which will be set by specifying different values for the different components of the QoI metric. This is called the desired-QoI whereas the actual QoI received by the application is called the delivered-QoI. A quantification of QoI is important, as:

- If the device calculates that it may meet the desired-QoI, it will process the video locally and no bandwidth is used up in streaming the video.
- If the device calculates that it may not meet the QoI requirements of the application, it may reject the query and offer the desired QoI by streaming the video to a powerful server. It may take significant capacity to transfer the video at the original frame rate. In the case of traffic videos, the original frame rate (of 30 fps in case of real-time video) may not be required. To deliver the desired QoI, the powerful server may require a lower frame rate. The video, if streamed at the lower frame rate (quality of video decreases), will save bandwidth and battery charge on the less powerful device.

In this dissertation, videos obtained from a stationary video camera perched at traffic signals have been analyzed. These videos monitor multiple lanes of bi-directional traffic. Without any prior knowledge of the videos (such as angle of the camera, length of the road monitored by the camera etc.), the videos have been analyzed in real-time on two platforms, to find out traffic flow parameters such as the total number of vehicles and the average speed of vehicles travelling on that stretch of the road. From these videos, not only have these parameters been extracted, but each identified vehicle has also been classified --as a car or a truck--and their color has also been detected. In case of stored videos, if there is no bound on the time taken by the videos to be processed, the accuracy and precision will almost be the same on all platforms, but timeliness will differ. However, in the case of processing real-time videos, where there is a time bound, the accuracy and precision of the information is dependent on the maximum number of frames the system can process per second. In order to reduce the size and quality of the video to match the

performance of the system on which it is to be analyzed, the frame rate of the video processed may be decreased. However, with a decrease in the frame rate, the QoI decreases. If the desired QoI is calculated to be achievable by the system, the video is processed locally. Otherwise, the video will be streamed it to a more powerful processor. To meet the QoI requirement of the user, the powerful processor may not need the original frame rate. Therefore, the device will calculate the lowest frame rate it must stream to the more powerful server to get the desired QoI before streaming the video.

This research not only focuses on extracting relevant information from traffic videos, but also provides a model by which bandwidth and battery charge on the current device may be saved when the QoI of the data requested by the user cannot be met on the current device, and must be streamed to a more powerful server. Bandwidth is saved by using the model to determine the lowest frame rate to be streamed by the device, and compressing and streaming the original video accordingly to get the desired QoI.

Chapter 2

Related Work

Monitoring traffic videos to extract information has been an active area of research in computer vision. The system developed in this dissertation first decides whether the QoI requirement can be met on the device. If yes, the video is locally processed. If no, the video is streamed to a more powerful device. It may happen that the video does not need to be sent at the full frame rate. If that is the case, the video is customized by selecting a lower frame rate at which if the video is streamed, the QoI requirement can be met. After the video is streamed, the vehicles are detected and information is obtained when the detected vehicles cross the virtual lines. Some of the existing techniques for analyzing QoI requirements, customizing the videos for saving bandwidth, vehicle detection, and vehicle tracking have been discussed below.

QoI

QoI has usually been used in data retrieval [36], sharing stored data [37], and characterization of data extracted using sensor networks [38]. In [39], the author performs a QoI aware analysis of wireless networks, where they define QoI to be composed of many contextual and intrinsic metrics. They also show that QoI-awareness alters the estimate of operational capacity of a network considerably. In this dissertation, the trade-off between a contextual metric: accuracy, an intrinsic metric: precision and frame rate has been studied.

Video customization techniques to save bandwidth

In [40], authors Ghinea *et al.* state that the users' understanding and perception of the information contained in multimedia clips does not undergo much loss even when the quality of the video clips are severely degraded. Degrading the videos will save bandwidth when they are being transmitted. Many techniques for "degrading" the quality of the video clips have been suggested. Some of them are discussed below.

In [41], *Goor et al.* suggest a scheme, where objects are prioritized by the content provider and just the objects of high priority are streamed, based on the object encoding properties of MPEG-4. In [42], based on the bandwidth constraints and priorities, the encoded objects are streamed. To differentiate between the systems in [41, 42], if there is a video of traffic with both vehicles and pedestrians in motion, and the system gives a higher priority to vehicles, the system in [41] will just detect the vehicles in motion in the foreground and stream them, whereas the system in [42] will detect both pedestrians and vehicles, but will decide to stream only the vehicles due to the bandwidth constraints. In [43], *Tseng et al.* build upon the existing MPEG-7 tagging system that also changes the video quality based on the content, to make it dependent on the device on which the video is viewed as well as the user preferences. In [44], Kishoni *et al.* analyze the bandwidth utilization constrains while using a network to transmit videos of a wireless surveillance system. In [44], the authors have created an information management policy that will state the ideal bandwidth allocated from the end user's perspective. The technique here uses concepts from [45, 46] about network flow for bit-rate services. To do this, Kishoni *et al.* have formulated a pricing based flow control method that allocates the ideal bandwidth to cameras that are simultaneously accessing the low-bit rate, wireless network to transmit videos. The pricing based algorithm calculates on every round the price per unit bandwidth. The cameras adjust their bandwidth such that it maximizes their net utility. This

adjustment in bandwidth usage is made by changing the frame rate and the detection threshold. Although these methods provide good customization techniques to save the bandwidth, prior information about the video, user, or device has to be known in order for the video customization to be successful. The system implemented in this dissertation needs no prior knowledge about the video to be processed. In this dissertation, depending on which camera wants to transmit information to the server, the ideal frame rate will be calculated for the video to achieve the desired frame rate and also save bandwidth. The detection threshold is not changed, as it may cause some overhead to the system.

Vehicle detection

After the frame rate at which the video is to be processed is decided, the video is either processed locally, or streamed to a more powerful server. After the video is on the device that can meet the desired QoI, it is processed to extract information such as the total count of vehicles, speed, color and type of vehicle. The first step to extracting any information out of surveillance videos is detecting vehicles in the video. This is done by background detection and object extraction. In [47], Setchell *et al.* use a model based method to detect the vehicles. Although this method is highly accurate, this requires the users to have a detailed geometric model of all the types of vehicles, which is unrealistic at times. In region based detection, background subtraction is first performed to distinguish the moving objects from the background. The foreground objects are detected by subtracting background image from the current image and if the difference is above an established threshold, the connected component, or the blob is determined. This method however, only works for free flowing traffic, and when there is congestion, two vehicles may collude, leading to the detection of one big blob, thus giving an error. In [48], Yu *et al.* detect the vehicles using a mixture of background subtraction and edge detection techniques. In the

implementation of the system in this dissertation, background subtraction is performed, after which the contours of the blobs are detected, which are then dynamically updated in every frame.

Vehicle Tracking

Following the detection of the vehicles, in order to extract traffic flow parameters from the traffic video, these detected vehicles must be tracked. There are three methods to do this. The first method is called the closed-loop tracking. In closed-loop tracking, after the vehicle is detected, it is tracked continuously throughout the length of the road (field of view, FOV) covered by the surveillance camera. This method is capable of tracking multiple targets [49]. The second method is called data association tracking. This method uniquely identifies and tracks a vehicle by locating a uniquely connected area of pixels. Cheng *et al.* [50] utilize this method and implement tracking using Kalman prediction. The third method, and the method used in this dissertation, is the tripline system [51]. In the FOV, there are certain regions, which can detect whether or not a vehicle has crossed it. It can do so by noting the change in pixels caused by the vehicles in the foreground mask. The only drawback of this method is that when a vehicle is crossing the region capable of detecting the change, the contour of the vehicle must already be detected.

In the system implemented in this dissertation, virtual lines are placed on the video, similar to the placement of buried wire sensors, and as vehicles cross these virtual lines, they are counted, color of the vehicle is determined and other parameters are calculated. The novelty of the system implemented in this dissertation lies in the fact that it combines all the above aspects of analyzing QoI requirements, customizing the frame rate of the video to save bandwidth, vehicle detection and vehicle tracking.

Chapter 3

Implementation

Detection of vehicles in video

The system used in this dissertation has been implemented using OpenCV [52] , which is an open-source, computer-vision library for extracting and processing meaningful data from images. In particular the OpenCV-Python library is used, which is a library of Python bindings designed to solve computer vision problems.

The algorithm used in this dissertation takes the video as input (stored file or video stream), and returns as output the count, color, type of vehicle and average speed of the vehicles in the FOV, following these steps [52]:

Step 1: Convert color space

Step 2: Background extraction

Step 3: Calculation of the foreground mask

Step 4: Filter the resultant image of Step 3

Step 5: Threshold the filtered image to detect contours on it.

Step 6: Filter contours and calculate circumcenter, *c*, of contour.

The steps are elaborated in the following section and are illustrated in Figure 3-1.

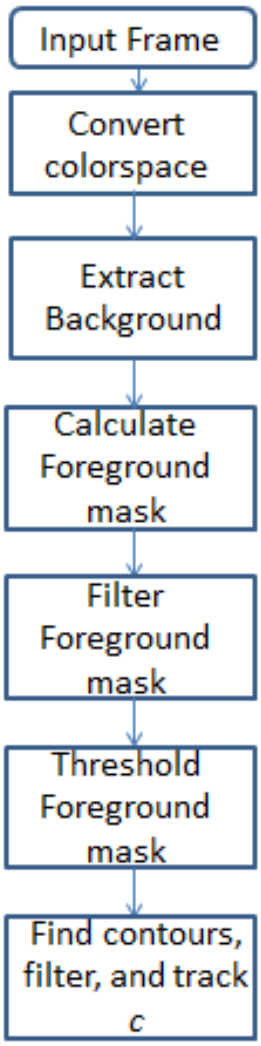

Figure 3-1: Flow diagram of the methodology followed in this dissertation

Step 1: Convert color space

The video stream (either stored or in real-time) is in the BGR color space. Every frame that is retrieved from this video is first converted to a gray color space (Figure 3-2). The method *cv2.cvtColor()* [52] is used for this purpose.

Figure 3-2: Illustration of Step 1. Original frame in BGR color space (left), Frame after processing in Step 1 in Gray color space (right)

Step 2: Background extraction

Background extraction is an important step, in order to calculate the correct foreground mask, which then allows the correct detection of the vehicles. If an image of the bare background is available, then this step may be skipped. However, in the system implemented in this dissertation, it is assumed that nothing except the video is available. Hence, to correctly create an image of just the background with no objects in it, the method of Running Average [52] is used. This method learns the average and standard deviation of each pixel from its model of the background.

The function *cv2.accumulatedWeighted()* [52] is used for this purpose. This function finds out the average of all the frames provided as input to it using the following relation:

$$dst(x,y) \leftarrow (1 - alpha).dst(x,y) + alpha.src(x,y)\ if\ mask\ (x,y) \neq 0 \quad \textbf{(1)}$$

In equation 1 [52], *src* is the source image provided, *dst* is the output image or the accumulator image, *alpha* is the weight of the accumulator image (i.e. how fast the accumulator updates or "forgets" the earlier image). The higher the value of *alpha*, the faster the update speed, so even small motion is detected. In the implementation of the system in this dissertation, the value of *alpha* was set to 0.01 as that value performed well experimentally. Figure 3-3 and Figure 3-4

show the result of the background extraction step after using the Running Average method for different values of *alpha*.

Figure 3-3: The last frame received as input from Step 1

Figure 3-4: Extracted background when *alpha* equals 0.01 (left) and when *alpha* equals 0.1(right)

Step 3: Calculation of the foreground mask

After getting the background from Step 2, the difference between the current frame and the background is calculated to get the foreground mask (Figure 3-5). Hence, the foreground mask will only contain objects that are not present in the background, which will be objects in motion. The method *cv2.absdiff()* [52] is used for this purpose.

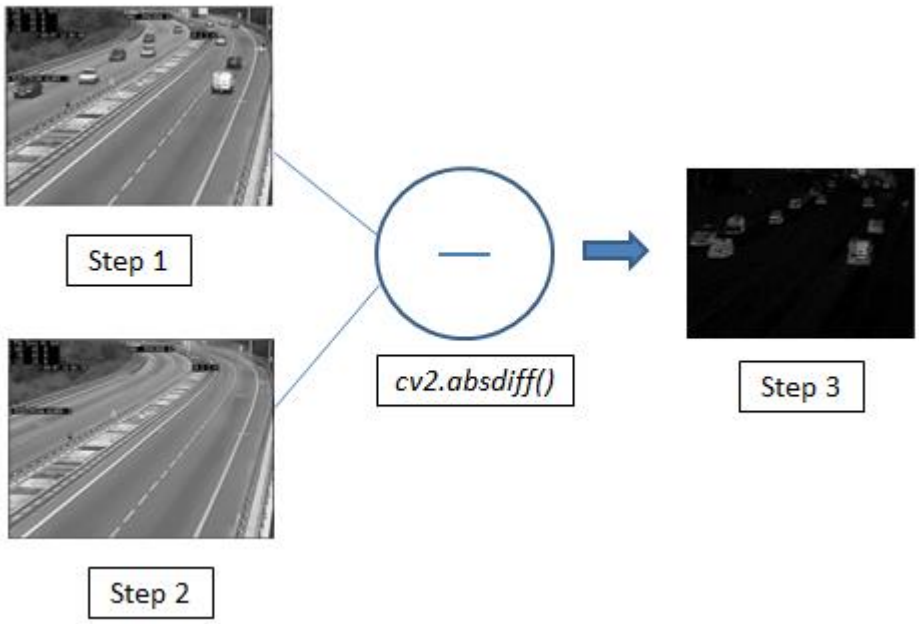

Figure 3-5: Generation of foreground mask by subtracting the input frame from the background

Step 4: Filter the resultant image of Step 3

For better and robust results, the small noise areas in the resultant image from Step 3 need to be eliminated. This is done in three steps:

a. Blur it using a Gaussian filter: This is done using the function *cv2.GaussianBlur()* [52]. The effect is illustrated in Figure 3-6.

Figure 3-6: Blurring image generated from Step 3 using a Gaussian filter

b. Erode it: This is done using the function *cv2.erode()* [52]. The effect is illustrated in Figure 3-7.

Figure 3-7: Effect of cv2.erode() on image

c. Dilate it: This is done using the function *cv2.dilate()* [52].The effect is illustrated in Figure 3-8.

Figure 3-8: Effect of dilate() on image

Step 5: Threshold the filtered image to detect contours on it.

If the value of the pixel in consideration in the image is greater than a threshold value, it is assigned the value white, else it is assigned the value black. The function used is *cv2.threshold()* [52].

Figure 3-9: Thresholding the filtered image

Step 6: Find contours and calculate circumcenter of contour.

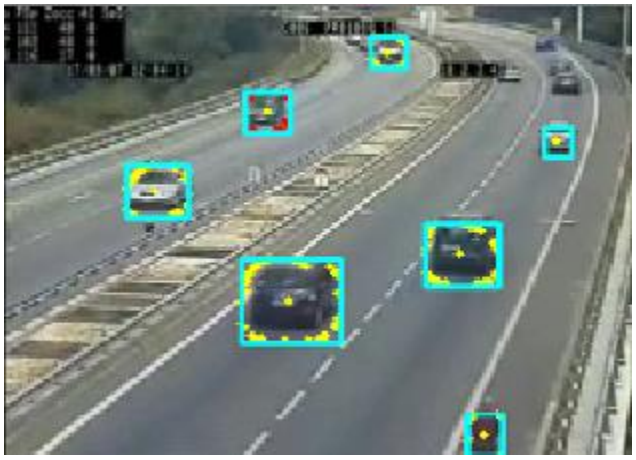

Figure 3-10: Illustration of Step 6

The method *cv2.findContours()* [52] is used to find contours (curve along the boundary joining all the continuous points having the same color, as illustrated by the yellow boundaries in Figure 3-10) in each frame in the binary image generated in Step 5. After finding the contours, the function *cv2.minEnclosingCircle()* [52] is used to find the circle with minimum area enclosing the contours big enough to be a vehicle in each frame. The center of these circles (illustrated by yellow dots in Figure 3-10), *c*, is then tracked to find out the parameters like count, color of vehicle, classification of vehicle and average speed of vehicles on that road. The techniques implemented to extract those parameters are further described in Chapter 4.

Chapter 4

Analysis

Technique for implementation of count

Experimental implementation of count

From Figure 4-1, it is observed that the distance between l_0 and l_1 is represented as *D*. If *c* crosses the line *l*, the count of the vehicles is increased by one. *c* is said to cross the line *l*, if the distance between *c* and *l* is less than *D*/2 (above or below *l*) .

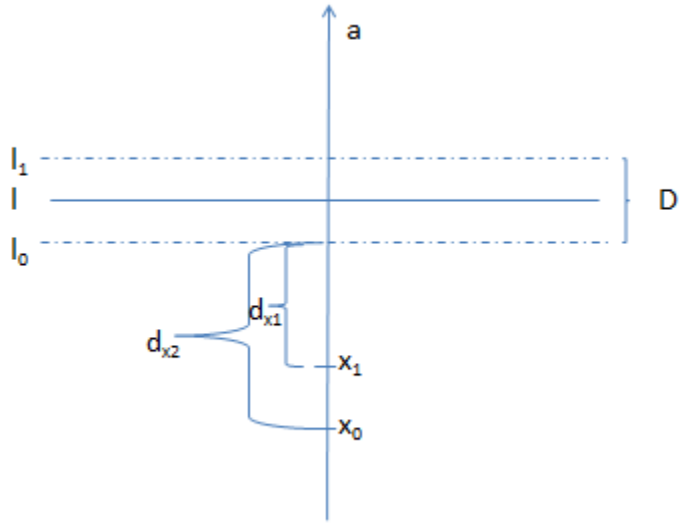

Figure 4-1: Schematic representation of a frame for counting the vehicles

Therefore, from Figure 4-1, it is observed that if *c* lies in between l_0 and l_1, it will be counted.

Mathematical model formulation for count

If the frame rate of a video is f fps, the time between each frame is $1/f$ second. d is the distance travelled by c in between two frames, and v is average velocity of the vehicles in the video. Then,

$$d = \frac{v}{f} \qquad \textbf{(2)}$$

If a car is moving and is detected in each frame, c will be detected along line a, as illustrated in Figure 4-2.

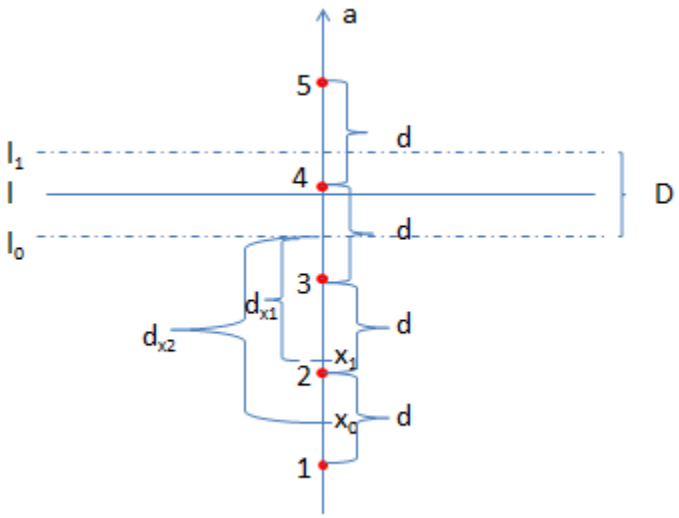

Figure 4-2: Distance travelled by c if every frame is sampled, and the vehicle is detected in each frame.

If every frame of the video is processed, c of the detected car will be on points 1, 2, 3, 4 and 5. However, if every second frame of the video is processed starting from 1, only points 1, 3 and 5 will be detected. In this case, the car will not be counted as it does

not lie between l_o and l_1 in any frame. On sampling frames that are farther apart, i.e. the frame rate is low, the distance, d, travelled by c in between each sampled frame will be large. When d is large and f is low, the chances of c lying between l_o and l_1 are higher if x_1 is farther from l_0 (but not farther than x_0). If c lies between l_0 and x_1 , in the next frame after having travelled d it will not be counted if it will not lie between l_0 and l_1. From these, statements, the following relations are obtained:

$$x_1 + D \leq d \quad \textbf{(3)}$$

$$x_0 \geq d \quad \textbf{(4)}$$

From Figure 4-2, it is observed that in the next frame, c will only be detected if it lies between x_1 and x_0 in the current frame. Taking the lower and upper bounds of x_1 and x_0, it is observed that:

$$x_1 = d - D \quad \textbf{(5)}$$

$$x_0 = d \quad \textbf{(6)}$$

And,

$$x_0 - x_1 = D \quad \textbf{(7)}$$

The accuracy, a_c, that the vehicle will be detected is defined as:

$$a_c = \frac{x_0 - x_1}{x_0} = \frac{D}{d} = \frac{D*f}{v} \quad \textbf{(8)}$$

Analysis of mathematical model

To find the value of D in equation 8, the range of velocities with which vehicles travel on a highway is considered to be between 40-70 mph due to the nature of the videos on which the model has been tested. D is set as the distance travelled by a car with the minimum speed, v_{min}, in between each frame in the unaltered video. By setting the value of D to this, irrespective of the speed of the vehicle in consideration, it is ensured that a vehicle, once detected between x_1 and x_0, and consequently detected in the region D, will not be recounted. However, if D is set to a value higher than this, there is a risk of a vehicle being detected multiple times. As the priority of the system implemented is to not recount the same cars, D, which is the distance between which the cars will be detected, is set as the distance travelled by the slowest car on the highway. Therefore, D is calculated as v_{min}/f_{max}.

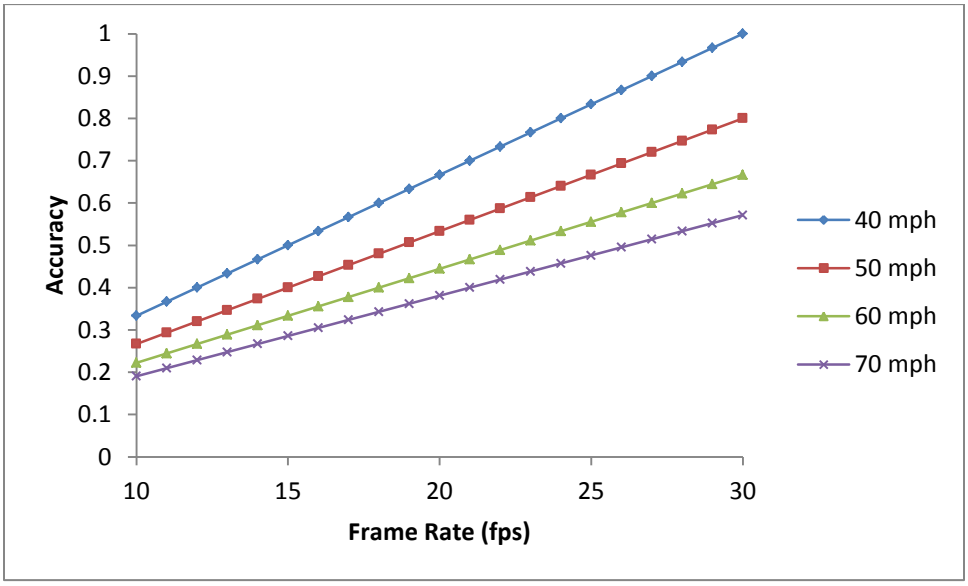

Figure 4-3: Frame rate *vs* accuracy for different speeds for base frame rate of 30 fps and speed as 40 mph.

Figure 4-3 shows the changes in accuracy by varying the frame sampling rate *f*, keeping 30 fps as the base frame rate. The vehicle speed was varied between 40-70 mph. The results are presented for different speeds. It is observed that when the average speed of the vehicles is 40 mph and the frame rate of the video is 30 fps; the accuracy of detecting the vehicle when every frame is processed is 1, or 100%. The accuracy decreases by half to 50% when every second frame of the video is processed, or the frame rate is reduced by half to 15 fps. Similarly, when the average speed of the vehicles is 70 mph, the best case accuracy achieved is 57.14%. This is because *D* is set according to the average speed of the vehicles being 40 mph as opposed to 70 mph in the video being analyzed. This leads to a decrease in the accuracy achieved as the faster vehicles may be missed, or not detected in *D*.

Technique for implementation of speed

Experimental implementation for speed

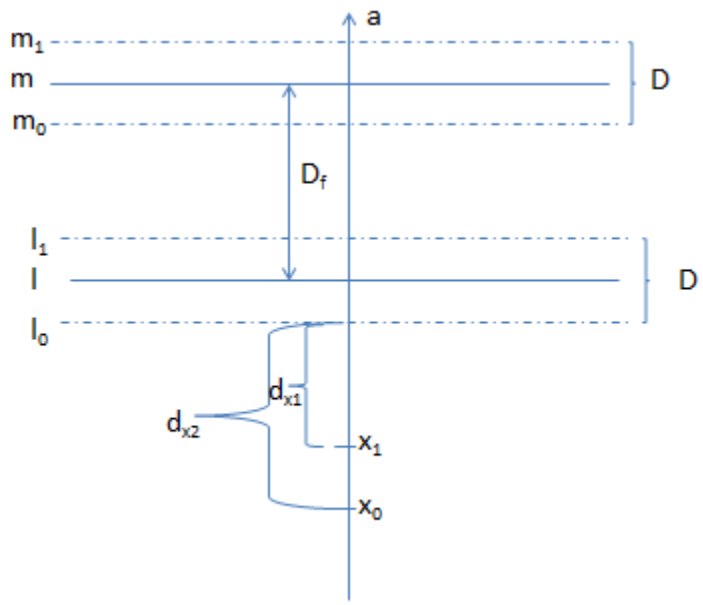

Figure 4-4: Schematic representation of a frame to detect speed

In Figure 4-4, *l* and *m* are two virtual lines, drawn on the frame in order to determine the speed. D_f is equal to *p*d*, where *p*>1. When *c* has already been detected within l_1 and l_2, the system time is recorded. After *p* frames, it will be detected within the lines m_0 and m_1 and the system time will be recorded again. The difference in these times will be calculated. Let that be *t*. The speed *s*, is then calculated as D_f/t.

Mathematical model for speed

In order to mathematically quantify the precision, the frame rate, *f*, of the video has to be taken into consideration. If *f* frames are processed per second, the precision for speed is limited by the measurement of time which is in increments of *1/f* seconds (implying that the precision for

speed measurement may be off by *1/f* seconds). During the time *1/f*, the system cannot measure speed, causing loss in precision. This loss in precision due to the error, *e*, is formulated as:

$$e = \frac{1/f}{T} = \frac{(1/f)*v}{D_f} \quad \textbf{(9)}$$

Where D_f is the distance between the two virtual lines, *v* is the velocity, f is the frame rate, and T is the time taken to travel the detection area (D_f).

Analysis of mathematical model for speed

The distance between the two virtual lines, D_f, is important. From equation 2, it is observed that shorter the distance, greater the error, and hence, lower the precision. However, if D_f is increased too much, the chances of error increases as multiple vehicles may simultaneously be present in the detection zone. For example, if one vehicle has just crossed the first virtual line *l*, but a different vehicle at the same instant crossed the second virtual line, *m*, this will lead the system to detect an extremely high speed of the first vehicle, which will be incorrect. This can be averted if each vehicle can be uniquely identified when it crosses each virtual line, i.e. when the vehicles are tracked. The analysis for speed will be done in two cases, one where the vehicles are tracked and the other where they are not tracked. In this scenario tracking means matching the objects in consecutive frames.

Case 1: No Tracking

In the case where the vehicles are not tracked, D_f is kept at the minimum distance between which the speed of the car may be accurately detected, i.e. the length of the car. The average length of a car was found out to be 4.12 meters or 0.003 miles.

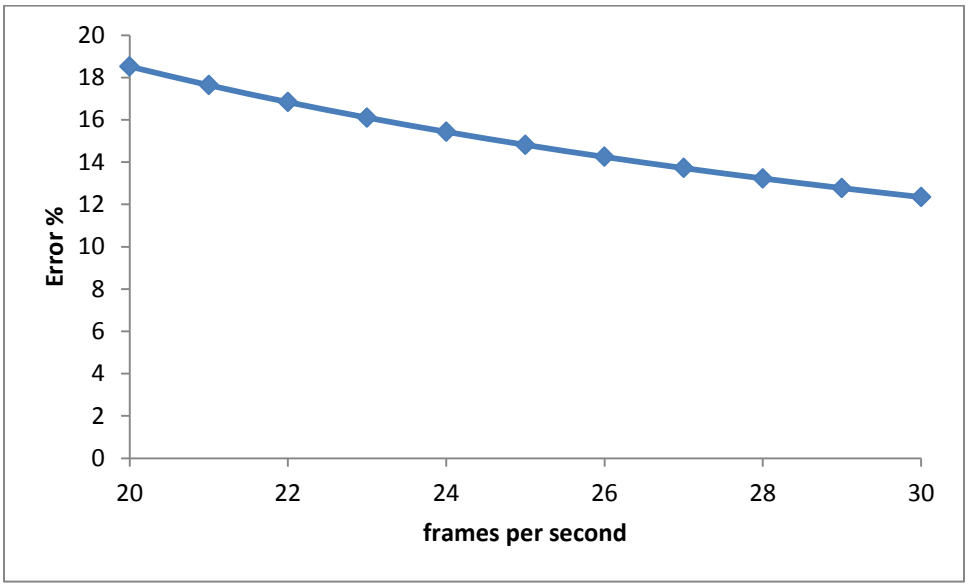

Figure 4-5: Error (%) *vs* frame rate in the case when tracking of *c* is not performed

Figure 4-5 quantifies the increase in the error percentage implying a loss in precision, with a decrease in the frame rate. It is observed that the error percentages range from 18.51 % to 12.34 % when the frame rates range from 20 fps to 30 fps respectively.

Case 2: Tracking

If the *c*'s path is tracked, the vehicles can be uniquely identified while crossing the virtual lines. In that case, D_f may be increased, to get lower *e*.

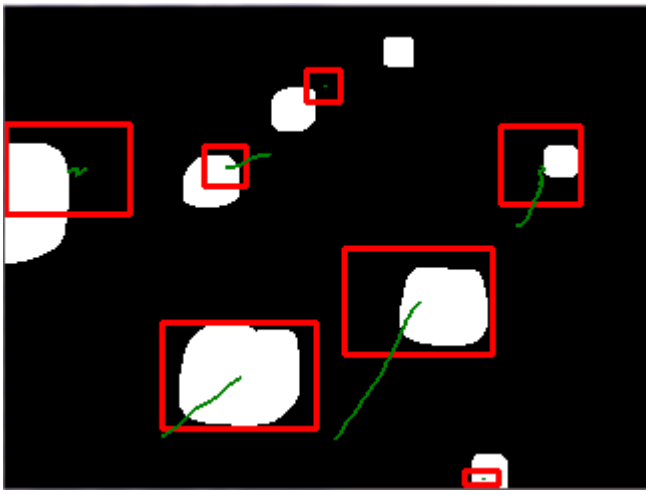

Figure 4-6: Experimentally tracking *c*

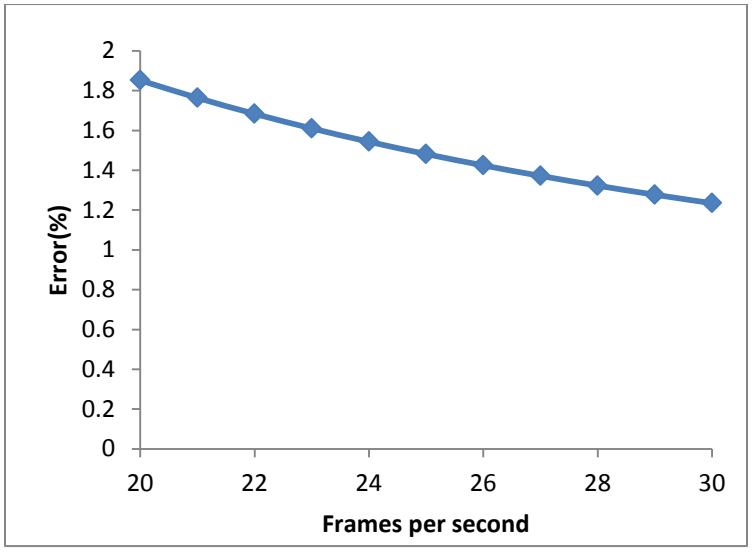

Figure 4-7: Error (%) vs frame rate in the case when tracking of c is performed

Figure 4-6 illustrates an experimental implementation of tracking of *c*. As an example, if the distance D_f is made to be equal to the length of 10 cars, i.e. D_f = 0.03 miles, the percentage error ranges from 1.85 % to 1.23% when the frame rates range from 20 fps to 30 fps, respectively, as shown in Figure 4-7. A possible disadvantage of tracking *c* is that information needs to be stored about subsequent positions of *c*, through each frame, and this may lead to storage overhead in case of real-time processing of large continuous video streams. The experimental technique employed in this dissertation does not employ the method of tracking *c*.

Experimental implementation for color and classification

From Figure 4-1, it is observed that a vehicle is counted when it is detected between l_1 and l_0. The instant where c is crossing l, and is detected within l_1 and l_0, the portion of the image containing the contour of the vehicle detected is cropped and the average color of that cropped image is displayed. Vehicles are classified as cars/trucks based on the size of their contour. Figure 4-8 illustrates the implementation of the system to calculate the different parameters: count, color, classification and the average speed of the vehicles in the FOV.

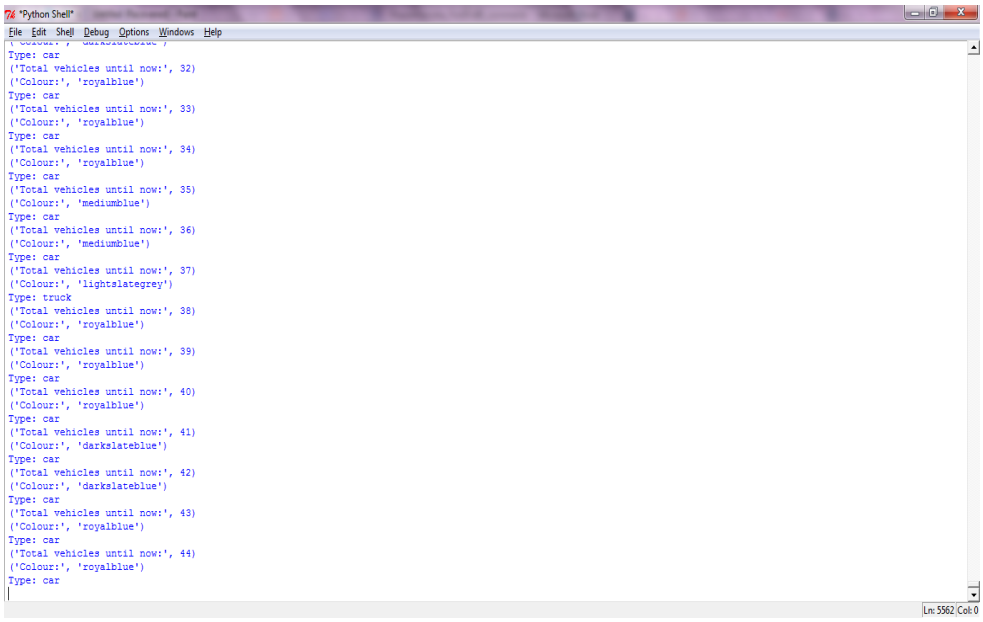

Figure 4-8 Implementation of the system to demonstrate the count, color and classification of vehicles.

In video 1 and video 2, the system can distinguish between the trucks and the cars with 100% accuracy. However, in video 3, the system had 7 false positives, and reported 10 vehicles as trucks, where as in reality, only 3 trucks were present in the video clip. These errors occur as the system classifies cars and trucks by comparing the sizes of their contours; this property is entirely intrinsic to the video, depending on the height from which the video was captured. In case of video 3, the height from which the video is recorded is much less than that of video 1 or

video 2, leading the contour perimeter of all the vehicles being bigger in general, than that as compared to the vehicles of video 2 or video 1, leading to a large number of false positives.

Chapter 5

Results and Discussion

Data Validation

The models developed have been validated using three stored videos of traffic. Their specifications and ground truths are presented in Table 5-1.

Table 5-1: Stored videos used of experimental validation [53, 54]

Video Name	Frame Rate (fps)	Total Number of vehicles	Average Speed of vehicles (mph)	Total length of video (seconds)	File Size (MB)
Video 1	14	44	60	33	2.41
Video 2	25	47	40	30	2.14
Video 3	25	29	35	68	10

Experimental results for count

In accordance with equation 8, for Video 1, $a_c = \frac{D*f}{v}$ = 0.31. From Table 5-1, it is observed that the ground truth is that there are a total of 44 cars in video 1. The number of cars detected in the video as predicted by the model should be 0.31*44 = 13.64 vehicles. Experimentally, the number of vehicles in video 1 was found to be 14, which is in accordance with the mathematical model (as a fractional number of cars may not be detected, so 13.67 ≈ 14

in this case). In video 2, a_c = 0.83. Hence according to the model, 39.01 vehicles will be found. The number of cars detected experimentally was 42. Similarly in video 3, a_c = 0.95. With a_c = 0.95, the model predicts that 27.55 cars will be counted. The number of vehicles counted experimentally was 29. A point to be noted here is that having an accuracy of greater than 1 means that the some vehicles may be detected more than once, leading to over-counting, and in turn, an inaccurate accuracy. It should be noted that an accuracy value greater than 1 is less accurate and not more accurate than 1.

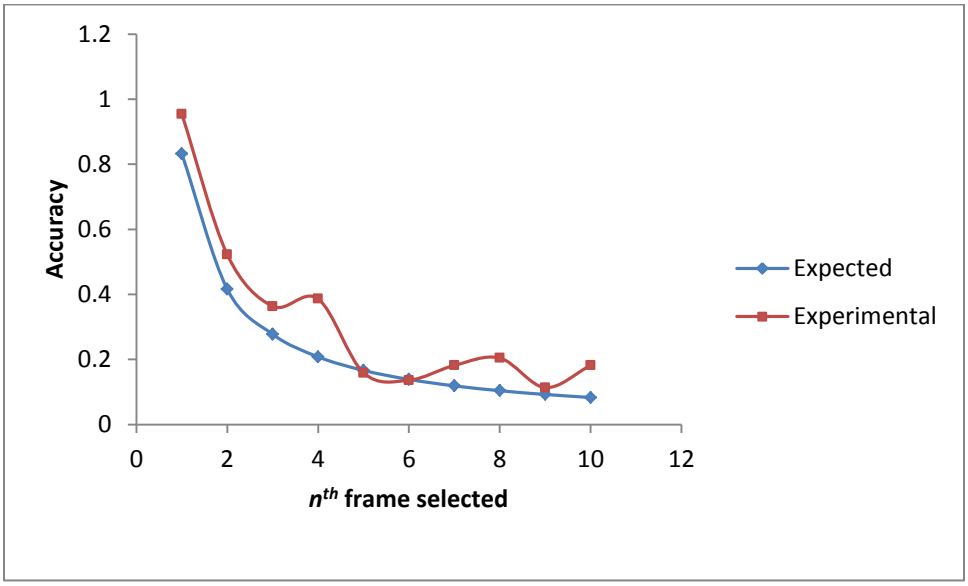

Figure 5-1: Experimental and Mathematical accuracy *vs* the frame rate selected for Video 2

Figure 5-1 illustrates the accuracy attained by selecting every n^{th} frame from the video 2 at the original frame rate (at 25 fps). The frame rate of the resulting video is 25/n fps. The maximum difference between the experimental and predicted model is 17.8%.

Experimental results for speed

For Video 1, error e is calculated as $e = \frac{v}{f * D_f} = 39.68\%$. Experimentally, the average speed of the vehicles was found out to be 65.43 mph, making the error percentage equal to 9.05%. For Video 2, error e, was found out to be 14.81%. Experimentally the average speed of the vehicles was found to be 32.07 mph. As the actual average speed of the vehicles is 40 mph, this data is not precise by 19.8%. In video 3, e = 12.96%. Experimentally the average speed of the vehicles was found to be 32.33 mph. This data is not precise by 7.63%. The fact that the experimental error percentages differ from those predicted by the model, can be attributed to the precision of the Python time module used in the implementation. This has been discussed in details in the Discussions section.

Effect of frame rate on bandwidth

Let there be three cases. In the first case, let the device recording the video transfer the video to a server for processing at the rate at which the video is gathered. In the second case, let the system calculate the minimum frame rate at which it must stream the video to meet the QoI requirement. In the third case, let the system first check if it can meet the QoI requirement locally. If yes, it will process the video locally. If not, it will calculate the minimum frame rate at which it must stream the video to meet the QoI requirement and stream the video according to the minimum frame rate. Let the system be able to meet the QoI requirement so long as it can process at least 15 fps.

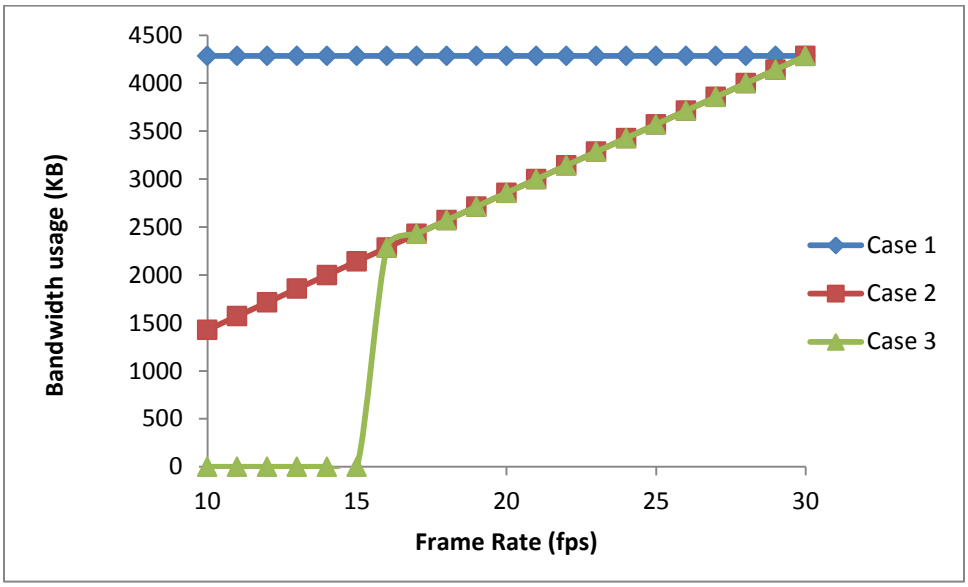

Figure 5-2: Bandwidth usage in the three cases

Figure 5-2 illustrates the bandwidth usage in the three cases. The frame rate of the video in question is assumed to be 30 fps, and the size of each frame is 4.76 KB (this is the average size of each frame used in the experimental analysis in this dissertation). In the first case, it is seen that no bandwidth is saved, as the full video is streamed to the powerful server. In the second case, a maximum of 2,856 KB is saved when the video is streamed at 16 fps instead of 30 fps, but some bandwidth is used in each case, as the video is always streamed. In the third case, 25,704 KB is saved when the maximum frame rate required is or below 15 fps. In this case, the video is processed locally, and nothing is streamed, thereby saving 25.10 MB. When the frame rate requirement is more than 15 fps, this system continues to stream the video at the lowest required frame rate, thereby saving more bandwidth than the first case. Hence, it is observed that the system developed in this dissertation is very effective.

Example of execution of system

Let a user be recording a real-time video (let that be video 2, for the experimental verification) on his/her android phone. Let this android phone be capable of processing video to extract information at a maximum of 15 fps. Let the user's query be: "What is the speed? Precision: 85% ". This means, that the percentage error permitted is15%. Let the experimental system not implement tracking. From Figure 4-5, it is seen that error % is 15.43% for 24 fps and 14.81% for 25 fps.

However, the android phone can only process 15 fps. Hence, the video must be streamed to the server. Here, instead of streaming 30 fps, the phone streams only 25 fps as obtained from the mathematical model, to get the desired QoI. This saves bandwidth of 5 frames per second ≈440KB (in case of video 2) and gives a result of 32.07 mph with19.82% error when executed on the system (reason for slight discrepancy in experimental results is discussed in the Discussions section).

In another example, let a user be recording a real-time video (video 3, for the experimental verification) on his/her android phone. Let this android phone process a maximum of 20 fps. Let the user's query be: "Is the road a highway or a side street? ". Let the minimum speed of vehicles on a highway be 50 mph and the maximum speed of a vehicle on a side street is 35mph. With a precision of 85% (error percentage = 15%), the minimum speed of a vehicle ranges from (42.5-57.5) mph and for side roads is (29.75-40.25) mph. From Figure 4-5, it is observed that the error % is 18.51% when the video is 20 fps. In this example, the android phone itself can process 20 fps. Hence, the video will be streamed locally. This saves bandwidth as the video did not have to be streamed and gives a result of 39.53 mph, which is below the upper limit of speed for side roads. Hence the video recorded by the user, captures a side road.

Hence, this system is useful for finding out the accuracy or the precision achievable by the platform it is on currently, and if not acceptable, determines the minimum frame rate it can be streamed to a server to meet the requested QoI.

Discussion

In both cases of speed and count, the experimental data was in accordance with the data values predicted from the mathematical model. The methodology used here can detect an accurate number of vehicles, the percentage error included in the average speed detection, the color and the type of vehicle in cases where there are multiple lanes as well as traffic moving in two directions. Another advantage of this method is that this can be run on real-time systems for long periods of time, as no intermittent tracking information is stored. .

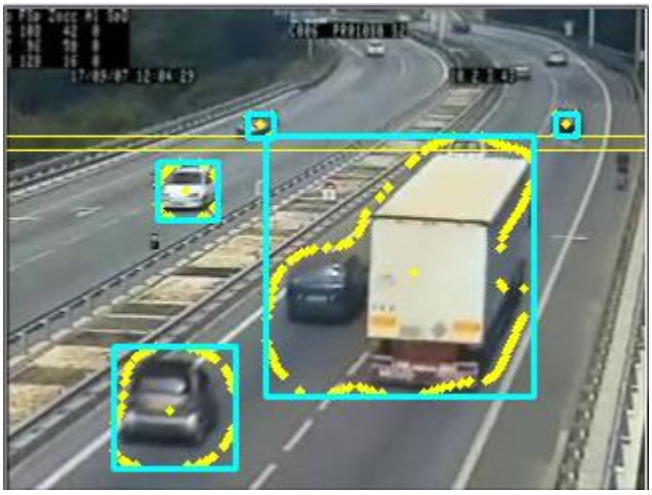

Figure 5-3: Examples of misdetections in implementation

A shortcoming for this method stems from the fact that the vehicles have to be detected correctly while crossing the virtual lines. This limits the height from which the video may be recorded, as, if the cameras are too high, the vehicles may be too small to be correctly detected.

Depending on the angle from which the video is obtained, the placement of the virtual lines will also affect the QoI. In order to correctly detect the vehicles, traffic always has to be in motion, otherwise the vehicle won't be detected. Sometimes if vehicles are too close together, or if occlusion occurs, they are detected as a single contour and counted only once. Occlusion is seen to occur in Figure 5-3. The truck and the two cars are detected as a single vehicle, and are counted as one vehicle instead of 3. From the experimental results, it is observed that neither precision, nor accuracy is 100% as all types of queries may not be addressed. Experimentally, some of discrepancies have been observed between the mathematically predicted and the experimentally obtained values of precision and accuracy. This has happened due to the shortcomings of the code discussed above. Another constraint is that a vehicle has to be correctly and completely detected in the region D between l_0 and l_1 (or m_0 and m_1) in order to be counted. It may happen that only a part of the vehicle is in region D, while c is not between l_0 and l_1 and c crosses l_1 in the next frame. In such cases, it is not counted and the vehicle is missed. For the measurement of speed, the Python time module is used. Although the measure of the time is returned as a floating point number, the system may not provide a precision better than 1 second. In video 1, the average speed of the vehicles is 50 mph. On an average, each will travel a distance of 0.0138 miles in one second. However the distance for which the time taken by the vehicle travel is measured is 0.003 miles. This has led to some discrepancies in the precision for the experimental and the mathematical values of speed.

In this implementation, the value of D has been considered as v_{min}/f_{max}. Another consideration for D may be v_{max}/f_{max}. This may result in over-counting cars which travel with velocity lower than v_{max}, but will not lead to the exclusion of any cars from being counted. This will satisfy a situation when the QoI query is "have *at most x* vehicles gone by this road?" If the QoI requirement is such, that there should be no over-counting, D should be set to v_{min}/f_{max}. This

may lead to the exclusion of vehicles with speed greater than $v_{\min}$ from being counted. This will satisfy QoI queries such as “have *at least x* vehicles gone by this road?”

Chapter 6

Conclusion

In this dissertation, information about traffic characteristics was extracted from stored videos for a wide range of QoI requirements. Mathematical relations were derived which quantified the relation between the frame-rate of the video and the quality of the information extracted. By using these relations, it was determined if the video could be processed locally. If not, it was calculated if the required QoI could be achieved by sending the video at a lower frame-rate. By processing the video locally, or streaming the video at a lower fame rate, bandwidth could be saved.

In the first chapter of this dissertation, the existing techniques for extracting information from traffic were discussed. Of these techniques, extracting information from traffic video was the most cost effective and popular method. In the second chapter of this dissertation, the existing techniques for analyzing QoI requirements, customizing the videos for saving bandwidth, vehicle detection, and vehicle tracking were discussed. Based on a combination of some of these existing techniques, a system was implemented which is discussed in the third chapter. Mathematical relations were developed between frame rate and accuracy in case of count; and frame rate and precision in case of speed in the fourth chapter of this dissertation. The fifth chapter discussed the results obtained from the mathematical models and the experimental implementations and saw that both models were in agreement of each other most of the times. The drawbacks of the existing experimental implementation were also discussed.

It was observed, that by dropping the accuracy requirement from 100% to 80%, and the percentage of error permitted from 12.34% to 15.43%, a bandwidth of 6 frames per second, or

28.56 KB per second of the video could be saved. For a 30 second video of traffic, this would result saving up to 856.8 KB of bandwidth.

Hence, by implementing the system and sending video at a lower frame rate, not only was battery saved on the less powerful device (by off-loading the computation, as well as having to send fewer frames), but bandwidth was saved as well.

References

1. Gupte, S., et al., *Detection and classification of vehicles.* Intelligent Transportation Systems, IEEE Transactions on, 2002. **3**(1): p. 37-47.
2. Tseng, B.L., C.-Y. Lin, and J.R. Smith. *Real-time video surveillance for traffic monitoring using virtual line analysis*. in *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*. 2002. IEEE.
3. Kilger, M. *A shadow handler in a video-based real-time traffic monitoring system*. in *Applications of Computer Vision, Proceedings, 1992., IEEE Workshop on*. 1992. IEEE.
4. Li, X., Z.-Q. Liu, and K.-M. Leung, *Detection of vehicles from traffic scenes using fuzzy integrals.* Pattern Recognition, 2002. **35**(4): p. 967-980.
5. Zhou, J., D. Gao, and D. Zhang, *Moving vehicle detection for automatic traffic monitoring.* Vehicular Technology, IEEE Transactions on, 2007. **56**(1): p. 51-59.
6. Robert, K. *Video-based traffic monitoring at day and night vehicle features detection tracking*. in *Intelligent Transportation Systems, 2009. ITSC'09. 12th International IEEE Conference on*. 2009. IEEE.
7. Buch, N., J. Orwell, and S.A. Velastin, *Detection and classification of vehicles for urban traffic scenes.* 2008.
8. Zhiwei, H., L. Jilin, and L. Peihong. *New method of background update for video-based vehicle detection*. in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*. 2004. IEEE.
9. Zhang, G., R.P. Avery, and Y. Wang, *Video-based vehicle detection and classification system for real-time traffic data collection using uncalibrated video cameras.* Transportation Research Record: Journal of the Transportation Research Board, 2007. **1993**(1): p. 138-147.

10. Bas, E., A.M. Tekalp, and F.S. Salman. *Automatic vehicle counting from video for traffic flow analysis*. in *Intelligent Vehicles Symposium, 2007 IEEE*. 2007. IEEE.

11. Morris, B. and M. Trivedi. *Robust classification and tracking of vehicles in traffic video streams*. in *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*. 2006. IEEE.

12. Wang, G., D. Xiao, and J. Gu. *Review on vehicle detection based on video for traffic surveillance*. in *Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on*. 2008. IEEE.

13. Chen, Y.-L., et al., *A real-time vision system for nighttime vehicle detection and traffic surveillance.* Industrial Electronics, IEEE Transactions on, 2011. **58**(5): p. 2030-2044.

14. Morris, B. and M. Trivedi. *Improved vehicle classification in long traffic video by cooperating tracker and classifier modules*. in *Video and Signal Based Surveillance, 2006. AVSS'06. IEEE International Conference on*. 2006. IEEE.

15. Beymer, D., et al. *A real-time computer vision system for measuring traffic parameters*. in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. 1997. IEEE.

16. Jung, Y.-K. and Y.-S. Ho. *Traffic parameter extraction using video-based vehicle tracking*. in *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*. 1999. IEEE.

17. Ozkurt, C. and F. Camci, *Automatic traffic density estimation and vehicle classification for traffic surveillance systems using neural networks.* Mathematical and Computational Applications, 2009. **14**(3): p. 187.

18. Rad, R. and M. Jamzad, *Real time classification and tracking of multiple vehicles in highways.* Pattern Recognition Letters, 2005. **26**(10): p. 1597-1607.

19. Chachich, A.C., et al. *Traffic sensor using a color vision method*. in *Photonics East'96*. 1997. International Society for Optics and Photonics.

20. Kogut, G.T. and M.M. Trivedi. *Maintaining the identity of multiple vehicles as they travel through a video network*. in *Multi-Object Tracking, 2001. Proceedings. 2001 IEEE Workshop on*. 2001. IEEE.

21. Saunier, N., T. Sayed, and C. Lim. *Probabilistic collision prediction for vision-based automated road safety analysis*. in *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*. 2007. IEEE.

22. Jiménez-Moreno, R., et al. *Video surveillance for monitoring driver's fatigue and distraction*. in *SPIE Photonics Europe*. 2012. International Society for Optics and Photonics.

23. Lei, Z., Z. Xue-fei, and L. Yin-ping. *Research of the real-time detection of traffic flow based on OpenCV*. in *Computer Science and Software Engineering, 2008 International Conference on*. 2008. IEEE.

24. Coifman, B., et al., *A real-time computer vision system for vehicle tracking and traffic surveillance*. Transportation Research Part C: Emerging Technologies, 1998. **6**(4): p. 271-288.

25. Harvey, B.A., G.H. Champion, and R. Deaver. *Accuracy of traffic monitoring equipment field tests*. in *Vehicle Navigation and Information Systems Conference, 1993., Proceedings of the IEEE-IEE*. 1993. IEEE.

26. MacCarley, A., *City of Anaheim/Caltrans/FHWA Advanced Traffic Control System Field Operational Test Evaluation: Task C Video Traffic Detection System.* California Partners for Advanced Transit and Highways (PATH), 1998.

27. (ITE), I.o.T.E., Traffic detector handbook , 2nd Ed., Washington, D.C., 1998.

28. Kwon, J., P. Varaiya, and A. Skabardonis, *Estimation of truck traffic volume from single loop detectors with lane-to-lane speed correlation.* Transportation Research Record: Journal of the Transportation Research Board, 2003. **1856**(1): p. 106-117.

29. Pushkar, A., F.L. Hall, and J.A. Acha-Daza, *Estimation of speeds from single-loop freeway flow and occupancy data using cusp catastrophe theory model.* Transportation Research Record, 1994(1457).

30. Wang, Y. and N.L. Nihan, *Can single-loop detectors do the work of dual-loop detectors?* Journal of Transportation Engineering, 2003. **129**(2): p. 169-176.

31. Schrank, D. and T. Lomax, *The 2005 Urban Mobility Report. Texas Transportation Institute, Texas A&M University*. 2005.

32. Officials, T., *AASHTO Guide for Design of Pavement Structures, 1993*. Vol. 1. 1993: AASHTO.

33. EPA, *(US Environmental Protection Agency) 2001.* National Air Quality and Emissions Trends Report, 1999: p. EPA 454/R-01-004. EPA. North Carolina

34. Bar-Noy, A., et al. *Quality-of-information aware networking for tactical military networks*. in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*. 2011. IEEE.

35. Alberts, D.S. and R.E. Hayes, *Understanding command and control*. 2006, DTIC Document.

36. Burgess, M., W.A. Gray, and N. Fiddian, *Establishing a taxonomy of quality for use in information filtering*, in *Advances in Databases*. 2002, Springer. p. 103-113.

37. Missier, P., et al. *Quality views: capturing and exploiting the user perspective on data quality*. in *Proceedings of the 32nd international conference on Very large data bases*. 2006. VLDB Endowment.

38. Bisdikian, C., et al. *Building principles for a quality of information specification for sensor information*. in *Information Fusion, 2009. FUSION'09. 12th International Conference on*. 2009. IEEE.

39. Balan, H.V., et al., *QoI-aware Analysis of Wireless Networks.*

40. Ghinea, G. and J.P. Thomas. *QoS impact on user perception and understanding of multimedia video clips*. in *Proceedings of the sixth ACM international conference on Multimedia*. 1998. ACM.

41. Goor, S. and L. Murphy. *An Adaptive MPEG-4 Streaming System based on Object Prioritisation*. in *Proceedings of Irish Signals and Systems Conference*. 2003.

42. Vetro, A., H. Sun, and Y. Wang, *MPEG-4 rate control for multiple video objects.* Circuits and Systems for Video Technology, IEEE Transactions on, 1999. **9**(1): p. 186-199.

43. Tseng, B.L., C.-Y. Lin, and J.R. Smith. *Video summarization and personalization for pervasive mobile devices*. in *Electronic Imaging 2002*. 2001. International Society for Optics and Photonics.

44. Kishoni, T., S. Callaway, and M. Byers. *Utility-based flow control for sequential imagery over wireless networks*. in *Systems and Information Engineering Design Symposium, 2007. SIEDS 2007. IEEE*. 2007. IEEE.

45. Kelly, F.P., A.K. Maulloo, and D.K. Tan, *Rate control for communication networks: shadow prices, proportional fairness and stability.* Journal of the Operational Research society, 1998: p. 237-252.

46. Kelly, F., *Charging and rate control for elastic traffic.* European transactions on Telecommunications, 1997. **8**(1): p. 33-37.

47. Setchell, C. and E. Dagless, *Vision-based road-traffic monitoring sensor.* IEE Proceedings-Vision, Image and Signal Processing, 2001. **148**(1): p. 78-84.

48. Yu, M., G. Jiang, and B. Yu. *An integrative method for video based traffic parameter extraction in ITS*. in *IEEE Asia-Pacific Conference on Circuits and Systems (2000: Tianjin China). IEEE APCCAS 2000... electronic communication systems*. 2000.

49. Sanders-Reed, J.N. *Multitarget multisensor closed-loop tracking*. in *Defense and Security*. 2004. International Society for Optics and Photonics.

50. Cheng, H.Y. and J.-N. Hwang. *Multiple-target tracking for crossroad traffic utilizing modified probabilistic data association*. in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. 2007. IEEE.

51. Kim, S.H., et al., *Real-Time Traffic Video Analysis Using Intel Viewmont Coprocessor*, in *Databases in Networked Information Systems*. 2013, Springer. p. 150-160.

52. Bradski, G., *The opencv library*. Doctor Dobbs Journal, 2000. **25**(11): p. 120-126.

53. Sobral, A., *Vehicle Detection with Haar Cascade*. https://www.behance.net/gallery/4057777/Vehicle-Detection-Tracking-and-Counting, 2012.

54. Goyette, N., et al. *Changedetection. net: A new change detection benchmark dataset*. in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. 2012. IEEE.