FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

TRAFFIC MONITORING SYSTEM USING COMPUTER VISION TECHNIQUES

Prepared by:

Aminah Ankoush	Khulood Sabri	Maha Hajja
1152161	1150697	1150144

Supervised by:

Dr. Mohammad Jubran

Dr. Aziz Qaroush

A graduation project submitted to the Department of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Systems Engineering.

BIRZEIT

January – 2020

شكر وتقدير

الحَمْدُ لله الذي بنعمتهِ تتمّ الصالحات، وبتوفيقهِ تتحقّقُ المقاصِدُ والغايات.

نتقدّم بجزيلِ الشكرِ والتقديرِ لكلّ من كانَ عَوْناً لَنا في مَسيرتِنا الجامعيّة التي نكلّل خِتامها بهذا العمل. أمّهاتُنا، آباؤنا، السّندُ المتين والحضنُ الدافئ الذي غمرنا بالحبّ والعطاءِ الجزيل. كادرُ الهيئةِ الأكاديميّة في دائرة هندسَةِ الحاسوبِ وهندسَةِ الكهرباء، ونخصّ بالذكرِ الدكتور الفاضل، محمّد جُبران، الذي لم يتوانَ في مساعدتِنا في إتمام هذا العمل.

المستخلص

في السنوات الأخيرة، بُذلت جهودٌ حثيثة لتطوير ما يعرف بالمدن الذكيّة، يتضمن هذا المفهوم آلية دمج التكنولوجيا الحديثة مع مرافق هذه المدن لتحسين مستوى ونوعية الحياة فيها. تعدّ أنظمة المواصلات الذكية من أهم أسس المدن الحديثة، وتسعى هذه الأنظمة لتنظيم السير وضمان سلامة الطرق. في هذا المشروع، تم استخدام مفهوم معالجة الصور (Image Processing) لتطبيق نظامٍ ذكيّ لرصد حركة المرور يعمل على معالجة الصور الملتقطة من كاميرات بثّ موزعة؛ لاستخراج معلومات مفيدة حول كثافة مرور المركبات وحركة السكان. يعد التقاط المركبات وتتبعها إلى جانب التقاط لوحات المركبات وتمييز أرقامها من العناصر الأساسية في مثل هذا النظام. وقد تم استخدام ما يعرف بالتعلم العميق (Deep Learning) في بناء النظام بهدف الوصول إلى أداءٍ أفضل. تم تقييم عدة أزواجٍ من الملتقطات (Detectors) والمتتبعات (Trackers) من حيث السرعة والأداء، وأظهرت النتائج أن استخدام ملتقط YOLO إما مع متتبع DLIB أو متتبع CSRT يعطي أفضل أداء. أما بالنسبة لالتقاط لوحات المركبات وتمييز أرقامها، فقد بينت النتائج أن استخدام Plate Recognizer، والذي يقوم بكلا المهمتين، يقدم أعلى أداء. نظرًا لكون بثّ الفيديوهات يستهلك سعةً (Bandwidth) كبيرة، فإنه لمن الضروريّ دراسة مدى إمكانية الموازنة بين السعة المستهلكة لبثّ الفيديو وكفاءة معالجة الصور الملتقطة. وقد أثبتت التجارب التي أجريت أنه من الممكن تقليل السعة المستخدمة إلى الخمس مع الحفاظ على مستوى الأداء.

Abstract

In recent years, there has been a lot of work done towards developing smart cities. The idea is to deploy technology within these cities to improve the quality of life of their residents and visitors. Smart transportation systems are considered as one of the important pillars of smart cities. These systems help in improving road safety and traffic management. This project implements a vision-based traffic monitoring system which processes images captured from distributed streaming cameras to extract useful information about traffic density and population movement. Vehicle detection and tracking besides license plate detection and recognition are the core elements in such a system. Seeking for higher performance, deep learning techniques and neural networks are employed. Combinations of different detectors and trackers were evaluated in terms of accuracy and speed. The results have shown that using YOLO detection model with DLIB correlation tracker or CSRT tracker gives the best performance. For plate detection and recognition, the evaluations demonstrate that Plate Recognizer software, which is capable of performing both detection and recognition tasks, provides the highest performance. The fact that the video streaming process consumes high network bandwidth, raises a big challenge to study the trade-off between the streaming bitrate and the efficiency of processing the captured frames. The experiments have proved that it is possible to reduce the consumed bandwidth to the fifth while preserving the same performance level.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

Abbreviation	Definition
ALPR	Automatic License Plate Recognition
AP	Average Precision
API	Application Program Interface
AVC	Advanced Video Coding
CCTV	Closed-Circuit Television Camera
CNN	convolutional neural network
CPU	Central Processing Unit
CSRT	Channel and Spatial Reliability Tracker
DNN	deep neural network
dt_Box	Detected Box
dt_id	Detector ID
ERD	Entity Relational Diagram
Faster-RCNN	Faster Region-based Convolutional Neural Networks
FFT	Fast Fourier Transform
FN	False Negative
FP	False Positive
FPS	Frames Per Second
GOP	Group of Pictures
GPS	Global Positioning System
GPU	Graphics Processing Unit
gt_Box	Ground Truth Box
gt_id	Ground Truth ID
HOA	Home Owners Associations
HOG	Histogram of Oriented Gradients
IOU	Intersection Over Union
KCF	Kernelized Correlation Filters
LSTM	Long Short-Term Memory
MB	Mega Byte
MIL	Multiple Instance Learning
MOSSE	Minimum Output Sum of Squared Error
MOT	Multiple Object Tracking
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
OCR	Optical Character Recognition
ORB	Oriented FAST and Rotated BRIEF
QP	Quality Parameter
RGB	Red Green Blue
RMS	Root Mean Square Error

RNN	Recurrent Neural Networks
ROI	Region of Interest
RPN	Region Proposal Network
RPN	Region Proposal Network
SDK	Software Development Kit
SIFT	Scale-Invariant Feature Transform
SOT	Single Object Tracking
SOT	Single object tracking
SQL	Standardized Query Language
SSD	Single Shot Detector
SURF	Speeded Up Robust Features
SVM	Support Vector Machine
TLD	Tracking, learning and detection
TN	True Negative
TP	True Positive
tr_box	Tracked Box
tr_id	Tracker ID
YOLO	You Only Look Once

Chapter 1 Introduction

1.1 Motivation

With the rapid expansion of urbanization, the number of vehicles on roads is increasing. This massive increase puts enormous pressure on cities to maintain better traffic management. Hence, precise and extensive data observation and analysis are needed to come up with strategic plans and achieve this traffic management. By adopting this intelligent traffic monitoring system, large sets of data will be aggregated and processed automatically to provide insights into traffic congestion reduction, road infrastructure planning, and traffic safety improvement. Implementing the system will be a win-win situation for both citizens and city administrators. It will not only help in making citizens' lives easier but also help administrators to keep their eyes on the roads and detect any incidents.

In specific, the importance of traffic management increases in developing countries like Palestine. That is because developing countries have limited funds and instrumentation, besides the fact that their cities are often old and do not have the required infrastructure to handle the rapidly increasing number of vehicles. In Palestine, recent statistics have shown that the number of vehicles has reached 202,270 [1]. Moreover, this number is expected to increase by more than 100,000 in the next 5 years [2]. In addition to being a developing country, Palestine also suffers from the Israeli occupation which limits its ability to build bridges and tunnels or even construct new streets in many regions.

1.2 Problem Statement

This project proposes and implements a vision-based traffic monitoring system that mainly focuses on extracting useful information about traffic. System implementation encounters many critical challenges to overcome. Starting with investigating vision-based approaches that would achieve the purpose of vehicle identification. Then, comparing the performance of different algorithms to implement these approaches. Another challenge arises from the fact that the system would consume a significant bandwidth during streaming video content over the network which in turn results in high cost and network congestion. Thus, a comprehensive study is required to find a good tradeoff between the streaming bitrate and the quality of vehicle identification. In brief, the problem investigated in this work is how to

design vision-based traffic monitoring system with high accuracy in a bandwidth-limited environment.

1.3 Contributions

The main contributions of the project are:

- Comparing different algorithms for detecting and tracking vehicles in terms of accuracy, required bitrate, and delay.
- Comparing different models for detecting and recognizing license plates in terms of plate and character accuracy.
- Collecting and annotating a dataset of more than 2000 images for Palestinian license plates.
- Investigating the trade-off between the transmission bitrate and the accuracy of different object detection and tracking as well as character recognition algorithms.
- Designing and implementing a surveillance web application that generates various statistics and reports about roads traffic.

1.4 Report Outline

The report consists of ten chapters. **Chapter 2** gives a general overview of the system design. **Chapter 3** gives a brief background about object detection, object tracking, character recognition and video streaming. **Chapter 4** summarizes some related works to object detection and tracking as well as plate recognition. **Chapter 5** describes the implementation methodology of the system. **Chapter 6** talks about the datasets used in the training processes, the experiments and the evaluations. **Chapter 7** illustrates the evaluation process of different detectors, trackers and recognizers, and discusses the evaluation results. **Chapter 8** studies the relation between the streaming bitrate and the performance of evaluated models. **Chapter 9** presents the features of the developed application and the frameworks used. **Chapter 10** discusses the limitations encountered, the conclusion and the future work.

Chapter 2 System Overview

2.1 System Description

The system proposed and implemented here is based on the client-server network architecture. The client-side of the system consists of the cameras placed at the common crossroads and highways, which in turn stream frames captured continuously through a limited bandwidth network to the server-side. The cost restrictions on the network bandwidth require a good selection of streaming parameters, including frame rate, resolution and quality. At the server node, the frames are processed using computer vision techniques and neural networks to detect vehicles, track their trajectories and identify their plate numbers. Figure 2-1 shows an overview of the system design.

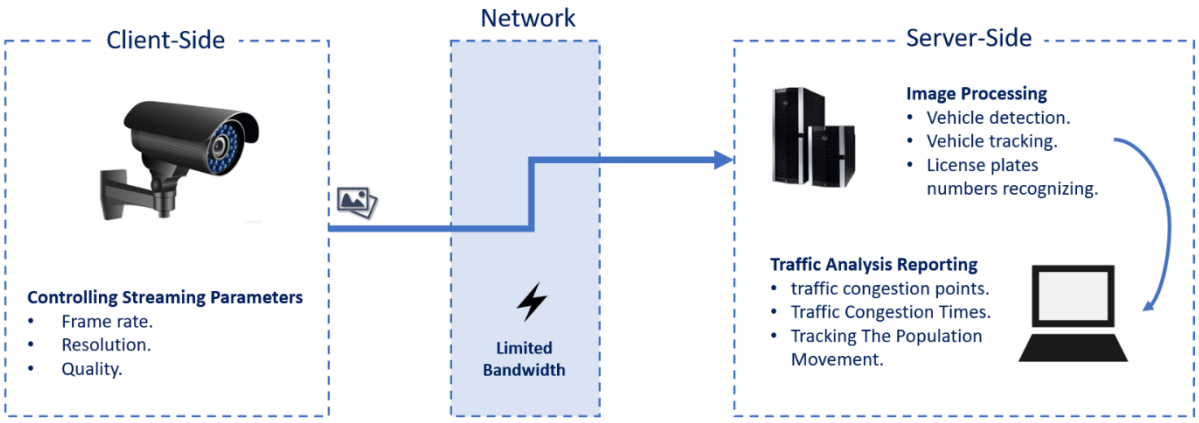

Figure 2-1: Overview of the System Design

Later on, the system could implement the results of the processing step in diverse traffic applications. These applications could be in analytic fields that help in urban road planning and traffic infrastructure management, such as analyzing traffic congestion points and times, identifying dangerous roads where a lot of accidents happen, and monitoring the demographics and the population movement by recognizing vehicles plate numbers. On the other hand, the results could be used in traffic-controlling applications. For example, controlling traffic lights based on traffic density, or detecting traffic violations and punishing disobeying drivers. Examples of these violations are running red lights, exceeding the speed limit, and parking at prohibited places. In the implemented application, the main features provided currently are in analytical and statistical fields based on the requirements suggested by Ramallah's

Municipality in Palestine, which adopted the idea of running and validating the proposed traffic monitoring system including the web application using their CCTV camera.

2.2 Ethical View

In the proposed system, streets will be monitored using CCTV cameras. To keep it legal and ethical, visible warning signs indicating that the street is monitored should be placed by the responsible authorities. Moreover, it should be guaranteed that the data captured by the system is accessible only by the authorized people and for legal purposes. As Palestinians living under Israeli occupation, these videos could be used against us in some situations. Therefore, the videos should be only live streamed using a secure channel and should not be saved.

From the point of view of some ethical theories, the system seems to be ethically acceptable. According to the 'Act Utilitarianism' moral theory, which believes that the performed action should produce the greatest good for the greatest number of people [3], using the system might be harmful if the data were used for privacy infringement of citizens. However, it will bring important advantages to the community; The data extracted by the system could be used for better management of traffic and streets design. Therefore, the benefits of using this system actually exceeds its harms.

From 'Social Contract' theory perspective, people live in a society according to a contract or an agreement between them for their mutual benefit. Thus, their moral obligation is dependent on that contract [4]. In the Palestinian society, the placement of surveillance cameras is not considered a crime by law as long as they target public places. However, placing a surveillance camera that targets a private place such as somebody's house needs proper permissions, otherwise it is considered a crime. In this system, the cameras will be positioned in public streets which is considered legitimate in Palestine.

Chapter 3 Background

This chapters gives an overview about techniques used for object detection and tracking. Moreover, it presents some methods used for vehicle license plate detection and recognition.

3.1 Object Detection and Tracking

Computer vision is a field of computer science that refers to how a computer can process, analyze and understand digital images to retrieve meaningful information and automate important tasks that take humans big effort and time to do [5]. Object detection and object tracking are considered among the most challenging tasks in the field of computer vision. They are being used in many practical applications across different domains, including surveillance systems, medical imaging, retailing industry and many other applications [6]. In this chapter, objection detection and object tracking are investigated in details.

3.1.1 Object Detection

Object detection is the process of localizing objects within a scene and recognizing their categories. Localization involves distinguishing the object from the background and other objects, while category recognition deals with classifying the object into one of the predefined classes or categories [6]. The techniques used for object detection fall into either classical approaches or deep learning-based approaches.

1) Classical Approach

This approach is based on extracting hand-crafted features of the image, such as colors, edges and corners in order to build a feature vector that represents the image. Many algorithms were implemented to extract this feature vector, including scale-invariant feature transform (SIFT), histogram of oriented gradients (HOG) and Speeded Up Robust Features (SURF). After constructing the feature vector, it is passed to a machine learning model such as Support Vector Machine (SVM) to train it on the classification process [7].

However, this approach for object detection faces many challenges and problems. First, the variable number of objects in the image, while machine learning models usually expects the data to be represented by fixed-sized vectors. And since the number of objects

in one image is not fixed, the correct number of outputs is not known beforehand. Second, vulnerability to the variations in object's size. Additionally, the assumption of classical classification models that the target object covers most of the image although in some situations it covers only a small percentage of it [8].

The solution for these two problems is usually using the sliding windows technique. Windows of different sizes are passed through the image to localize objects. However, this procedure is inefficient especially in real time applications. That is because it takes multiples of the original time needed for only one image. This raises the need for more efficient and robust techniques for object detection which are provided by the deep learning approach [8].

2) Deep Learning Approach

Deep learning has been a huge game changer in machine learning. It overwhelmed classical approaches and became state-of-the-art in computer vision in general and object detection in specific [9]. Deep learning or neural networks mimic the human brain and its neurons in processing data and recognizing patterns. Deep learning algorithms learn progressively about the image as it goes through the network layers. Early stage layers learn how to detect low-level features, and subsequent layers combine these features together into a more complete representation [10].

Convolutional Neural Networks (CNN) are the state-of-the-art models for object detection with deep learning. A CNN is composed of multiple layers. Each layer has number of nodes (neurons) with several inputs. Neurons take a weighted sum of its inputs and then pass them to an activation function to get the output. CNN in general consists of three types of layers, *convolutional layer* followed by a *pooling layer*, and finally a *fully connected layer* as shown in Figure 3-1 [11].

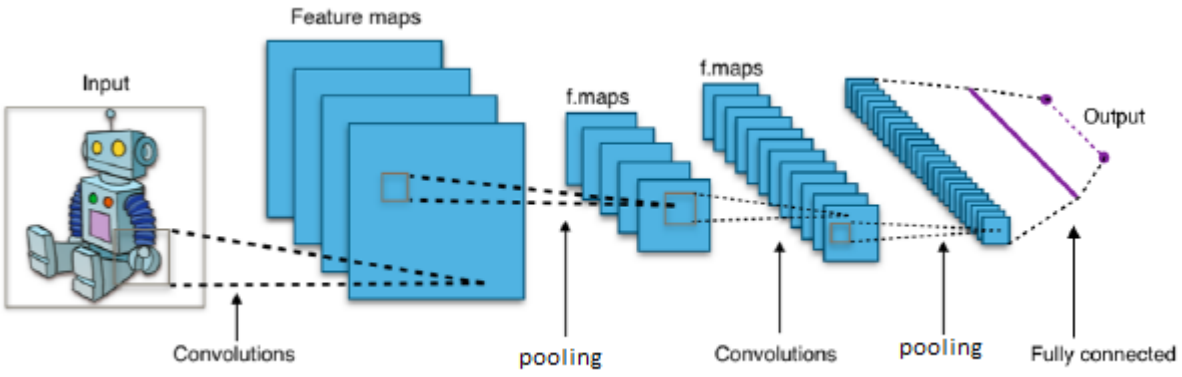

Figure 3-1:Layers of Convolutional Neural Network Layers [11]

Convolutional layers are the main building blocks of the CNN. They have a set of filters represented as two or three dimensional arrays of weighted matrices. Each filter is traversed across the input and an inner dot product between the filter and the input is done at each location. The output of the convolution with a filter is called a feature map. The convolutional layer would make the filter learn to capture a certain structure or pattern, so that the filter will find this pattern anywhere else. The **Pooling layer** is used to reduce the dimensionality of the feature maps. The output of pooling is a generalized version of the features that is more robust to local transformations. The common methods used for pooling are Maximum Pooling and Average Pooling. For the **Fully connected layer**, its main objective is to classify the output of the previous layers into a label. The feature maps are flattened and converted into a single vector that represents the probabilities of expected labels or classes [12].

Many architectures of deep learning models are built and available for use. To train a model, a large dataset of labelled images is required. The performance of the resulted model improves as more data is fed to it. The rest of this section discusses in details three popular deep learning models used for object detection, Faster R-CNN, SSD and YOLO.

a) Faster Region-based Convolutional Neural Networks (Faster-RCNN)

It is one of the most well-known object detection neural networks. it's also the basis for many networks like 3D object detection and segmentation. Faster-RCNN is a combination of three sequential neural networks - **Feature Network**, **Region Proposal Network**, and **Detection Network** [13].

The **Feature Network** is used to extract the good features of the image to generate a feature map. The output of this network does not change the shape or the structure of the original image. Then, this feature map is fed to the **Region Proposal Network** *(RPN)*, which generates number of Region of Interests (ROIs) that has high probability of containing objects. The output of this network is a set of bounding boxes and a value for each one to indicate whether it contains an object or not. The outputs of the two previous networks are fed to the **Detection Network** to generate the final output of classes and bounding boxes. Figure 3.2 shows the network pipeline of aster RCNN [13].

CNN-based object detection techniques handle detection as a pipelined classification problem. In other words, they first generate region proposals and then send them to classifiers to adjust the output. This leads to longer time in training and testing and so making them inappropriate for real-time object detection. To overcome this bottleneck of CNNs, new efficient architectures were built including SSD and YOLO models [14].

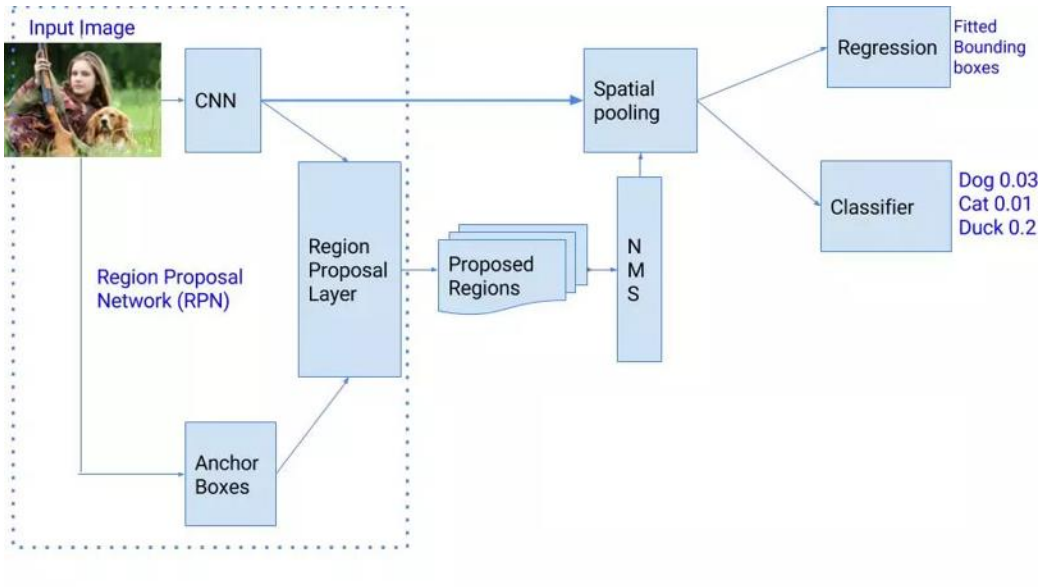

Figure 3-2: The Architecture of Faster-RCNN Neural Network [13]

b) Single Shot Detector (SSD)

As opposed to RCNNs, SSD has no region proposal network and designed for real-time detection. It is applied on the image for single time. The idea behind it involves two improvements; First, generating multi-scale feature maps independently and then applying small convolutional filters to predict classes and offsets of boundary boxes. The multi-scale feature maps are used to achieve high detection accuracy even on low resolution images [15]. Figure 3-3 illustrates the architecture of the SSD neural network. It is clear that as getting deeper in the network, the size of the generated feature maps gets smaller. Finally, a non-maximum suppression layer is used to remove duplicate predictions pointing to the same object to reduce the number of output predictions [16].

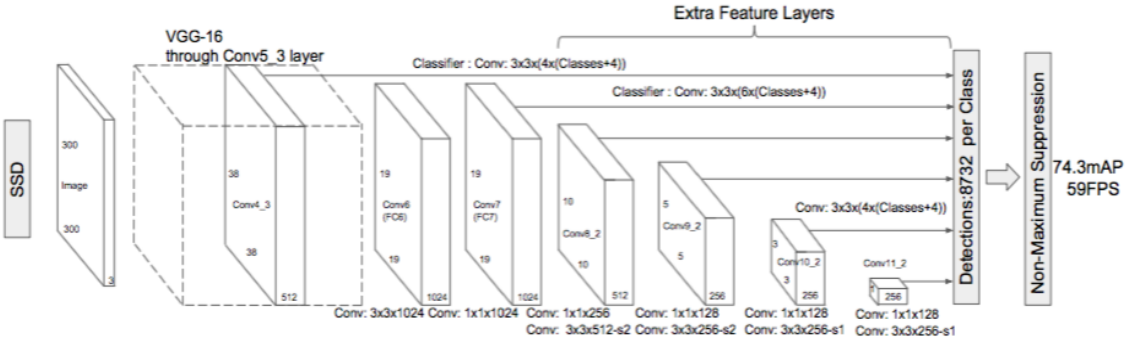

Figure 3-3: The Architecture of SSD Neural Network [16]

c) You Only Look Once (YOLO)

YOLO is the state-of-art in object detection with deep learning. As the name says, You Only Look at the image Once (YOLO) but in a clever way. The basic idea of YOLO is presented in Figure 3-4. First, the image is divided into a grid of S by S cells; Each cell is responsible for predicting a number of bounding boxes around it, and their corresponding confidence scores. The confidence score measures the probability that the bounding box contains an object. Moreover, the cells predict a class for each bounding box by generating a class probability distribution map over the image. The confidence score is then combined with the predicted class into the final score. Taking only scores over a predefined threshold, it ends up with the objects detected [17].

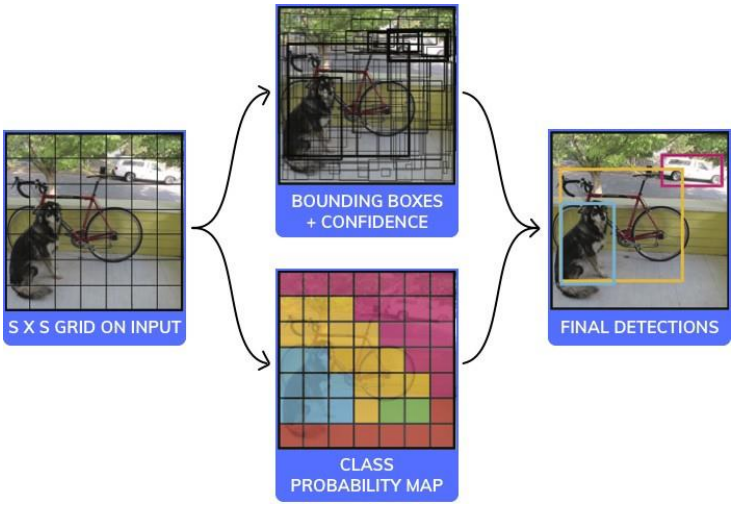

Figure 3-4: Steps of YOLO Detection Process [17]

Recently, a third version of YOLO (YOLOv3) came into light to solve the problems of older versions, especially the difficulty in dealing with small objects. YOLOv2 used the architecture of darknet-19, with a total of 30 layers. Whereas, YOLOv3 uses Darknet-53, a 53-layers network with 53 more layers for detection task, giving a total of 106 layers. The complexity of the new architecture led to slower performance in terms of speed, but more accurate predictions in terms of accuracy compared to YOLO v2 [18]. Figure 3-5 shows the architecture of the network of YOLOv3.

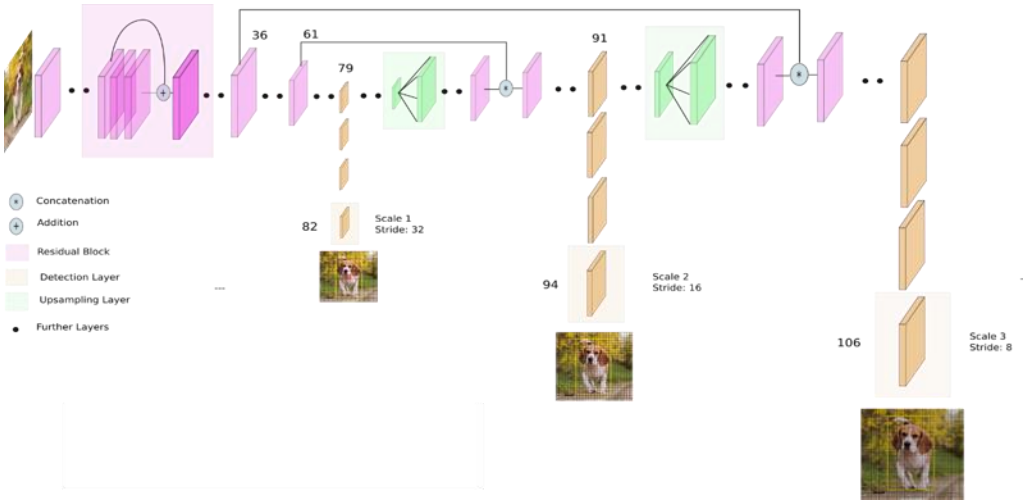

Figure 3-5: The Architecture of YOLOv3 Neural Network [16]

3.1.2 Object Tracking

Object tracking is an important computer vision term which is used frequently in the company of object detection. It involves the process of locating an object in successive frames of a video based on its initial state, and being able to determine if it is the same object appeared in the previous frames [19].

In Single object tracking (SOT), the main challenge is designing a sophisticated model that deals with scale changes, illumination variations and out-of-plane rotations. While in Multiple object tracking (MOT) additional issues are considered. First, locating the objects in the frame and assigning unique identity for each one. Then, maintaining their identities and trajectories in a video sequence [20]. The applications on multiple object tracking vary widely from traffic surveillance and human-computer interactions to virtual reality and many other applications [21].

The first step in MOT is initializing the locations of objects. Based on the initialization method, trackers can be grouped into two sets, Detection-Based Trackers and Detection-Free Trackers. In Detection-Based Tracking, as the name suggests, a trained object detector is applied to obtain the initial locations of the objects. And so, the performance of the tracker depends heavily on the accuracy of the detector. Hence, using a CNN-Based detector could greatly enhance the performance of the tracker. Detection-Free Tracking on the contrary uses manual initialization of a fixed number of objects in the first frame. Detection-Free trackers cannot deal with the case when new objects appear [22].

In the processing step, tracking algorithms can be categorized into online models and offline models. The difference is whether or not observations from future frames are considered when handling the current frame. In online or causal methods, only information from past frames are used. Offline or batch models, on the other hand, have access to the entire video, which means that information from both past and future frames can be used. However, in real-time applications, online models are used since future frames are obviously unavailable [22].

For the purpose of predicting the new locations of tracked objects, many approaches have been suggested. These approaches represent the states of the objects as a probabilistic

distribution based on existing observations from past frames. Kalman filter, Extended-Kalman filter and Particle filter are some of these distribution models. Kalman filter is used in the case of linear system and Gaussian-distributed object states. While Extended-Kalman filter can be used in nonlinear systems. In Particle filter, the distribution is represented as a set of weighted particles [22].

The last step in a tracking algorithm is to determine the correspondence between objects and detections based on the previous predictions, known as association process. Similarity measures considered in this step can include appearance measures, shape and distance measures. Appearance features are calculated from local and regional features of pixels in the bounding boxes e.g. color histogram, HOG and optical flow features. Whereas, distance and shape measures look at the geometry and the position of the bounding boxes. These measures can be weighted differently and then summed or multiplied to give a value that indicates the overall similarity between detected boxes [22].

In real-time applications, there are significant reasons to combine trackers with detectors rather than using only detectors. First, tracking is faster than detection because it preserves information about the appearance, location and direction of the object while detection always starts from scratch. Second, tracking algorithms attach an identity to each object and keep track of them, whereas the output of a detector is an array of rectangles that includes the objects. Third, tracking is more resistive to deformations and appearance variations compared to detection [23].

OpenCV (Open Source Computer Vision Library) is an open source computer vision library. The library has more than 2500 optimized algorithms, including both classic and state-of-the-art algorithms. OpenCV provides a variety of algorithms for object tracking. Currently in version 3.4.1, it contains 8 different implementations for single object trackers. These are BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE and CSRT. DLIB library, which is a toolkit that contains machine learning algorithms [24], also provides a common tracker, known as correlating tracker. Each algorithm of them has its own advantages and disadvantages. Table IV.1 shows a brief comparison between them as reported by OpenCV [23].

In this system, the used tracker should have reasonable speed to run in a real-time environment. It also should give good accuracy. Based on the comparison shown in Table 3-1, four trackers were chosen to be evaluated. These trackers are DLIB, CSRT, MEADIANFLOW and KCF.

Table 3-1: Comparison Between Different Types of Trackers [23]

Tracker	Pros	Cons
BOOSTING	Opened new areas for boosting in computer vision e.g. face detection and pedestrians' detection.	Very old (over a decade), slow, and unreliable when tracking fails.
MIL	Built upon BOOSTING algorithm with more enhancements, gives better performance than BOOSTING and does a reasonable job under partial occlusion.	does a poor job of reporting failure and does not recover from full occlusion.
KCF	Gives better accuracy and speed than MIL and reports tracking failure better than BOOSTING and MIL.	Does not recover from full occlusion.
TLD	Works the best under occlusion over multiple frames, deals best with scale changes.	Causes lots of false positives making it almost unusable.
MEDIANFLOW	Excellent tracking failure reporting and works very well when the motion is predictable and there is no occlusion.	Fails under large motion e.g. fast moving or changeable objects.
GOTURN	Based on CNN, robust to viewpoint changes, lighting changes, and deformations	Very slow and does not handle occlusion very well.

MOSSE	robust to variations in lighting, scale, and non-rigid deformations, very fast and operates at a higher fps (450 fps and even more	lags behind the deep learning-based trackers and not as accurate as CSRT or KCF.
CSRT	Gives higher accuracy than others.	A bit slower compared to others.
(DLIB) Correlation	robust and capable of running in real-time applications	does not handle occlusion and viewpoint changes very well.

3.1 License Plate Detection and Recognition

Identifying the license plate of a vehicle requires two essential steps; First, Detecting the position of the plate relatively to the vehicle, and then recognizing the characters of its text. For plate detection, the same classical and deep-learning detection techniques discussed earlier are applicable. In the following sections, plate recognition is illustrated and different pre-built systems for detecting and recognizing plates are introduced.

3.2.1 Plate Recognition

Optical Character Recognition (OCR) is the process of recognizing the patterns of characters in digital images or scanned documents. The text is presented to the computer as an image and the machine should turn that image into text, then produce something similar to a DOC or TXT file. For the OCR system to recognize and understand text using OCR, the text has to be segmented first, and the boundaries for each character should be found. Then the OCR would take each letter independently and assign an output to it [25].

One of the popular deep-learning based text recognition tools is “Tesseract”. Tesseract was first developed by Hewlett Packard Labs. Then in 2005, it was published as an open-source software by HP. That was until 2006 when google took over and actively developed it. Tesseract started supporting many image formats starting from version 3 and above, that version was based on the normal traditional computer vision techniques. However, when deep learning started overcoming traditional machine learning ways in its accuracy and

speed, it was just a matter of time until tesseract employed deep learning too. In tesseract version 4, Long Short-Term Memory (LSTM) neural networks are used [26].

LSTM network is a modified version of Recurrent Neural Networks (RNN). RNN consists of networks that perform the same function on all the inputs, but each output depends on both the input and the outputs calculated before. However, a disadvantage here is the gradient vanishing, meaning the previous outputs will start to vanish with time and will not affect the new outputs anymore. On the other hand, for LSTM, the act of remembering information for long periods is actually what they are built to do, not a thing that the network should be struggling with. So, the vanishing gradient is not a problem for LSTM [27].

3.2.2 Automatic License Plate Recognition (ALPR) Systems

1) Sighthound

Sighthound delivers ALPR cloud services to detect and recognize license plates from over 100 countries, states, and territories. Sighthound is built over deep learning technologies; it uses a sequence of three deep convolutional neural networks. In the first CNN, the plate is detected, it is localized and classified within an image. Then, the plate characters are segmented using multi-step segmentation, where the complexity of the segmented characters increases every step. After that, the segmented characters are passed to a binary CNN classifier to confirm the existence of characters in these segments. And finally, each character segment is classified into one of 35 classes. It should be mentioned that all CNNs were fine tuned to be robust to different variations such as lighting, blurriness, size and angle [28].

Two datasets of USA and European license plates of 956 images were used by sighthound team to test the end-to-end ALPR system. The detection part had an accuracy of 99.37% and the recognition part had an accuracy of 94.0% [28].

2) Plate Recognizer

Plate Recognizer is a company that provides ALPR services via API Cloud and SDK. Plate Recognizer system is trained on datasets from 100 countries, and is optimized

to work under different conditions such as low resolution, blurry and angled vehicles. Plate Recognizer suggests that its services could be the best choice when plate recognition is integrated in broader systems such as a parking lot or Homeowners Associations (HOA) gate access. Unfortunately, Plate Recognizer did not disclose its system implementation [29].

3.3 Video Streaming

Media streaming and video streaming in specific has gained a great popularity in the last decade. Actually, some studies have proved that video streaming is responsible for 25-40% of all internet traffic [30]. There are two types of streaming, live streaming and on-demand streaming. In live streaming the user watches a stream that is already in progress, such as TV programs or camera streaming. On the other hand, user can start, stop, pause, resume and seek an on-demand streaming [31].

In this system, videos are live streamed from surveillance cameras positioned at different streets. Thus, it is important to understand and control the critical limitations on this process, especially the high bandwidth required for streaming. Many strategies were proposed to overcome this problem. This chapter will illustrate some of these solutions that basically focus on advanced video compression techniques or quality adjusting.

3.3.1 Video Compression

Video compression or video encoding is the process of compressing a video and being able to re-approximate it [32]. There are two types of data compression, lossless and lossy. In lossless compression the original data can be reconstructed. On the other hand, in lossy compression the original data can only be approximated. However, if a good lossy compression algorithm is used, the decompressed data will not be distinguished from its origin. Many software and hardware video compression standards were developed. These standards are called codecs, and they are lossy compression standards. MJPEG and H.264 are two examples of these popular codecs [33].

1) MJPEG

MJPEG or Motion JPEG is a frame-by-frame compression technique, which means that each frame is compressed individually as a static image. This makes the

decompression process much faster and simpler, but it results in higher bandwidth usage compared with other compression algorithms. A Motion JPEG video may be 10 times larger than H.264 video [33].

2) H.264

H.264 or Advanced Video Coding (AVC) is the most common compression standard used in surveillance videos and many commercial media applications [33]. H.264 uses spatial and temporal compressions. First, it partitions the frame into macroblocks (n x n pixels). For spatial compression, a macroblock is predicted from surrounding previously-coded macroblocks. This is called intra prediction. However, for temporal compression, macro-blocks in similar regions of previously-coded frames are used for predicting the block, which is called inter prediction. After that, H.264 subtracts the macroblock from its prediction to form residual [34]. Figure 3.6 illustrates the difference between intra-prediction and inter-prediction.

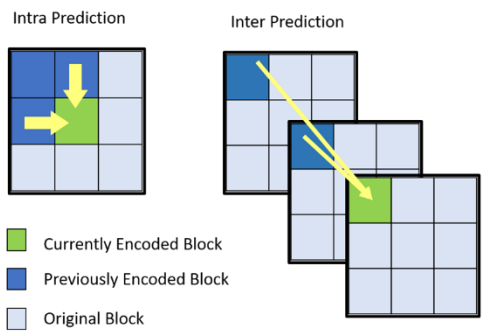

Figure 3-6: Difference between Intra-prediction and Inter-prediction Used in H.264

H.264 uses the Group of Pictures (GOP) structure, which means that the frames of a video sequence are either I-frames, P-frames or B-frames as shown in Figure 3.7 [33].

- I-frames: Inter frames are only spatially compressed.
- P-frames: Predicted frames are spatially and temporally compressed. The temporal compression depends only on previous frames in the video sequence.
- B-frames: Bidirectional frames are spatially and temporally compressed. The temporal compression depends on previous and subsequent frames in the video sequence.

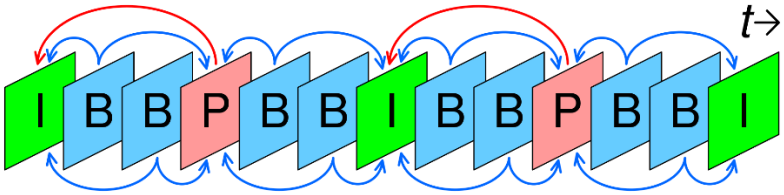

Figure 3-7: Group of Pictures structure Illustration [35]

After constructing the residual blocks, Integer Transform is applied. The transform results in a set of weights. When these weights are combined with a standard basis pattern the residual block can be recreated. After that, the transformed block is quantized; i.e. each weight coefficient is divided by an integer value that depends on the quality parameter (QP). The QP parameter ranges from 0 to 51, and the larger the QP the less the bandwidth used, and less video quality is produced [34].

Finally, the quantized weights are encoded; they are converted into binary codes using variable length coding or arithmetic coding [34].

3.3.2 Adjusting Video Quality

Adjusting video quality means reducing the video size by sacrificing some of its quality as long as it remains acceptable for the target use. There are three quality parameters that can be adjusted:

1) Spatial Resolution

Spatial Resolution or resolution refers to the number of pixels in an image. Resolution is identified by the width and height of the image (width x height) or by the total number of pixels in the image [36]. Resolution-based bitrate adaptation decreases the number of pixels in the horizontal and/or vertical dimension of each video frame [37].

The effect of resolution adaptation on the image quality highly depends on the scene content; Complex scenes with small details are usually more affected by resolution reduction. However, the need of these details is dependent on the application. Actually, for noisy images the loss of details increases the image quality. Moreover, the effect of resolution is also affected by the focal length (optical zoom) and the distance to the objects when the shot was taken. The effect decreases when the shot is taken from smaller distances [33].

2) Frame Rate

Frame rate or temporal resolution indicates the number of frames streamed per second (fps) in a video. The higher the frame rate of a video, the smoother the objects move [33]. Frame-rate-based bandwidth adaptation relies on encoding a lower number of frames per second. There are two methodologies of doing that: periodic frame dropping or consecutive frames dropping. Usually, the first method is preferable since the reduced data is less noticeable [37].

Decreasing the frame rate of a video does not necessarily decrease the consumed bandwidth. That's because in some codecs that use the difference between frames such as H.264, reducing the frame rate has an effect on the compression process [33].

Moreover, reducing the frame rate of different videos affects them differently depending on their content. Videos in which the movement is fast or continuously changes its direction are highly affected [37].

3) Video Quantization

Quantization is the lossy part of lossy video compression standards. It can be done using different algorithms and on different domains of the video such as color and frequency. Adjusting the quantization parameter (QP) of a video changes its quality and its bandwidth usage. This change varies significantly between different video compression standards. According to many researches, H.264 has produced a very satisfying video quality for different QP values [37].

Overall, many strategies were proposed over the years to reduce the bandwidth consumed by video streaming. There are also many researches that study the trade-off between the quality of the user experience when watching a video and the bandwidth required to transmit that video. However, studies on the trade-off between video streaming bandwidth and the quality of applying image processing techniques on it are still fresh.

Chapter 4 Related Work

This chapter introduces some of the previous work in the field of using image processing techniques in object detection and tracking as well as license plate detection and recognition.

4.1 Deep Neural Networks for Object Detection

The authors in [38] used Deep Neural Networks (DNNs) to detect a potentially large number of object instances with varying sizes in the same image using a limited amount of computing resources.

The work in the paper demonstrates that DNN-based regression is capable of learning features which are not only good for classification, but also capture strong geometric information. The system proposed in the paper consists of a total of seven layers as the general architecture for DNN. The first five are convolutional layers and the last two are fully connected layers, with replacing the soft max at the output with the regression layer to get a binary mask for the objects. However, this didn't work well for objects that are close to each other, so several masks were generated to deal with this.

A huge dataset is needed to train the proposed DNN for localization. Thousands of images were taken for each class with different scales and perspectives. Besides, the model needs to be trained for classification first, and then trained further for localization. The proposed DNN model performs well for detecting a small number of classes, e.g. 20 classes for the VOC dataset. The performance degrades as the number of classes increase.

4.2 Vehicle Detection Based on Convolutional Neural Networks

The authors in [39] propose a convolutional neural network (CNN) architecture to identify and localize vehicles in an image. The proposed CNN architecture is developed using Keras deep learning framework with Python and TenserFlow. It consists of six convolutional layers and five max pooling layers. The input layer takes batches of size 128x128x3. ReLU activation function is used in the hidden layers, while the sigmoid activation function is used in the output layers. The same network is used for classification and detection. However, the output layer produces one unit in classification and four units for the bounding box in detection.

Both detection and classification networks were trained on 100 vehicle images and 1866 background images after applying several transformations on them. They were trained for 100 epochs using Adam optimizer and a learning rate of 0.001. Training was done with and without using Fast Fourier Transform (FFT). During classification without FFT, precision and recall values were 100% and 97.6% respectively. Using FFT increases recall to 99.6% and keeps precision at 100%. For detection, the overlap between the ground truth box and the detected box (IOU) was used to evaluate results. If the IOU in an image is greater than 0.5, then it is true positive, otherwise it is false positive. It should be mentioned that in a background image the ground truth box is the whole image. Without using FFT, all detections were true positive. Using FFT resulted in 287 true positives and 246 false positives.

4.3 Detection and Tracking of Objects for Autonomous Vehicles

In [40], the authors present an algorithm for object detection, classification, and tracking. The proposed approach uses deep-learning network YOLO combined with data from a laser scanner to detect and classify objects. Once the objects are detected, the Oriented FAST and Rotated BRIEF (ORB) feature descriptor is used for matching objects between frames to find association or adding them as new objects. The state of the objects is updated using Extended Kalman Filter allowing for both localization and tracking of objects in the environment.

The algorithm developed in this paper is tested on the datasets provided by Oxford University Robotcar [41]. The Robotcar is equipped with cameras, LiDAR and GPS to collect data while traversing a crowded route in the center of Oxford. The results of the proposed algorithm as mentioned in the paper are good with some shortages. First, the images captured by the camera at night are not good and so the performance is low. Moreover, the tracking method, which depends mainly on matching key points, fails at night and tracks each object as a new object in every frame.

4.4 Thai License Plate Recognition Based on Deep Learning

In [39], license plate detection and recognition using deep learning techniques were presented. The focus was on Thai motorcycle license plates, which has a three-line license plate type.

The dataset for training and testing the proposed model was collected using mobile phone cameras. The dataset consists of 710 real images of license plates with high resolution and various orientations. For segmentation, a two-step method was used. Firstly, the segmentation of the three lines was done using SSD model that was trained using MobileNets and Inception-v3 with 590 images. After that, the character segmentation was trained and tested using the same approach. The accuracy for testing the rest of the images (120 images) was calculated to be 96.94% for line recognition and 91.76% for character recognition.

The challenge for this study was that many license plates were covered by frames for decoration, which made the characters not recognizable even by human eyes. However, the detection and segmentation for these conditions still worked and the characters were recognized if visible.

Chapter 5 System Implementation

Tis chapter explains the implementation of the proposed system. In the system, images are captured using live streaming cameras. Then, to localize and identify vehicles in these images, detection and tracking algorithms are combined together to work alternatively on them. In the first frame, the vehicles are detected by a detection algorithm, an identifier is assigned to each vehicle, and a tracker is initialized for it. After that, the trackers work for ***n*** frames, where ***n*** is relative to the frame rate of the video and need to be tuned carefully. In the n^{th} frame the detector works again to detect new vehicles, in this frame detector boxes and trackers boxes are matched to determine the correspondence between tracked vehicles from previous frames and detected vehicles. For matching, the intersection over union (IOU) measure is used where the IOU between two boxes indicates the overlap between them. If a detector's box and a tracker's box have an intersection area that is large relative to their union area, then they can be matched. After that, trackers are updated according to the matching results. A tracker that is not matched to any detector is deleted which corresponds to a vehicle left the monitored area. Whereas a detector box that is not assigned to any tracker is considered as a new object and so given a new identifier. Otherwise, detected boxes are given the identifiers of their matched tracked boxes. The same process is repeated continuously every ***n*** frames.

For license plates of vehicles, they are detected and recognized every *m* frames, where ***m*** is also relative to the frame rate. However, when there are no vehicles in the frame, or there is a small difference between this frame and the last frame in which the plates were detected, plate detection and recognition is not applied for the purpose of saving time in a real time application. Figure 5-1 illustrates how the components of the system work together on a sequence of frames.

The next step is assigning detected plates to the IDs of the tracked vehicles. This is done using the IOU and the Euclidean distance. If the IOU between a plate and a vehicle is larger than some threshold, then the plate is assigned to this vehicle. If a plate could not be assigned to any vehicle, the system searches for a vehicle that is close enough to it, i.e. the distance between them is below some threshold.

It should be mentioned that the system exploits the plate detection to improve the performance of the tracker and vice versa. If the same plate number is assigned to two different vehicle IDs,

then the system will know that the tracker has made a mistake and that these two tracked vehicles are actually one vehicle. Reversely, plate recognition errors are decreased when multiple plate numbers are assigned to the same vehicle ID. In this case, these numbers are reduced to the valid numbers that match the Palestinian plate patterns. After that, the valid number that has the largest frequency is chosen. In other words, the recognition that is assigned to this vehicle in most frames is chosen.

Eventually, some information and statistics discussed later on detail are extracted from the recognized number of a vehicle.

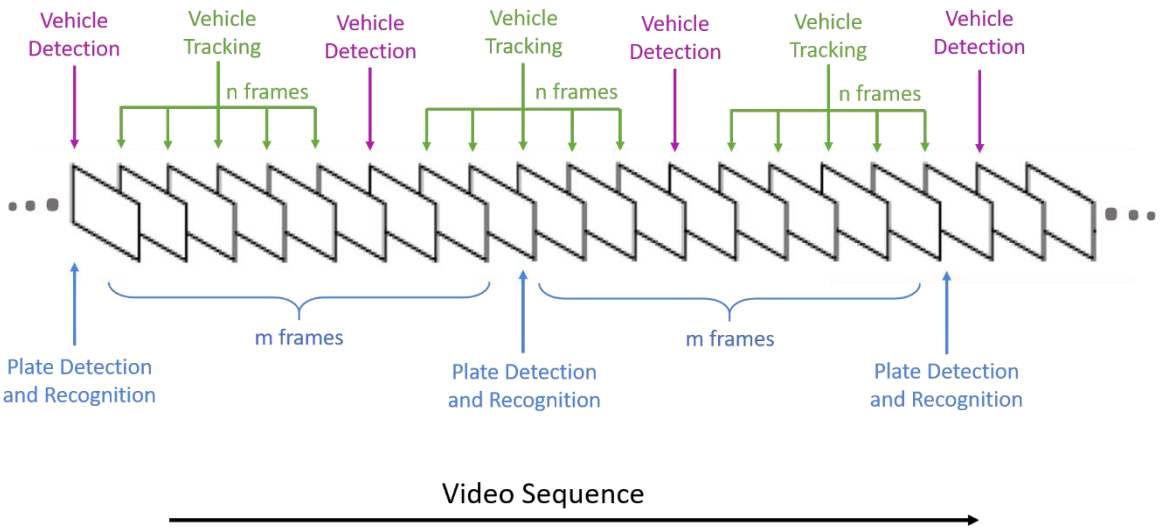

Figure 5-1: Detection and Tracking of Vehicles and Detection and Recognition of Plates as Applied to a Video Sequence

Chapter 6 Datasets

This chapter discusses the process of obtaining and preparing the datasets used to train and test the algorithms and models used in the system.

6.1 Vehicle Detection and Tracking Dataset

UA-DETRAC dataset was used in the evaluation experiments of the detectors and trackers. This dataset is a benchmark for object detection and multiple objects tracking. It consists of 100 annotated video sequences taken at 24 different locations including highways, traffic crossings, and T-junctions. The videos contain variations in illumination, scale and occlusion conditions. They have a frame rate of 25 fps and a resolution of 961x540 pixels [42].

The annotations of UA-DETRAC determine the bounding boxes, the ids and the types of the vehicles. They also describe some conditions such as weather and occlusion. Additionally, regions with low resolution are identified in the annotations as ignored regions [42].

UA-DETRAC is divided into 60 training and 40 testing sequences. The division was done randomly. However, it was ensured that the training and testing data share the same conditions. The training data consists of 82085 frames and contains 598281 vehicles. Only the training data of the UA-DETRAC dataset was used for the evaluation of this system. It is worthy to mention that the models used in this system are not trained on this training data [42].

6.2 Plate Detection and Recognition Dataset

There are many public annotated datasets for plate detection and recognition. However, unlike vehicles, license plates from different countries have different sizes, shapes, colors, fonts, and number patterns. Therefore, there was a need to collect a dataset for Palestinian license plates. With the help of Ramallah Municipality, we were able to obtain a 12-hour long video of traffic passing through Yafa street near Ramallah Municipality Building. This video was taken by a surveillance camera positioned upon a traffic light. It views the two lanes of the street, which means that it has front and rear views of the vehicles. Its resolution is 1920x1080 with frame rate of 10 fps. From this video, we extracted 1737 images for different vehicles with visible license plates. Figure 6-1 shows a sample frame from this video and

Figure 6-2 shows samples from the extracted vehicle images. Additionally, 200 images were acquired from different streets in Ramallah by three different phone cameras; iPhone XR, iPhone 7 and Xiaomi Mi5.

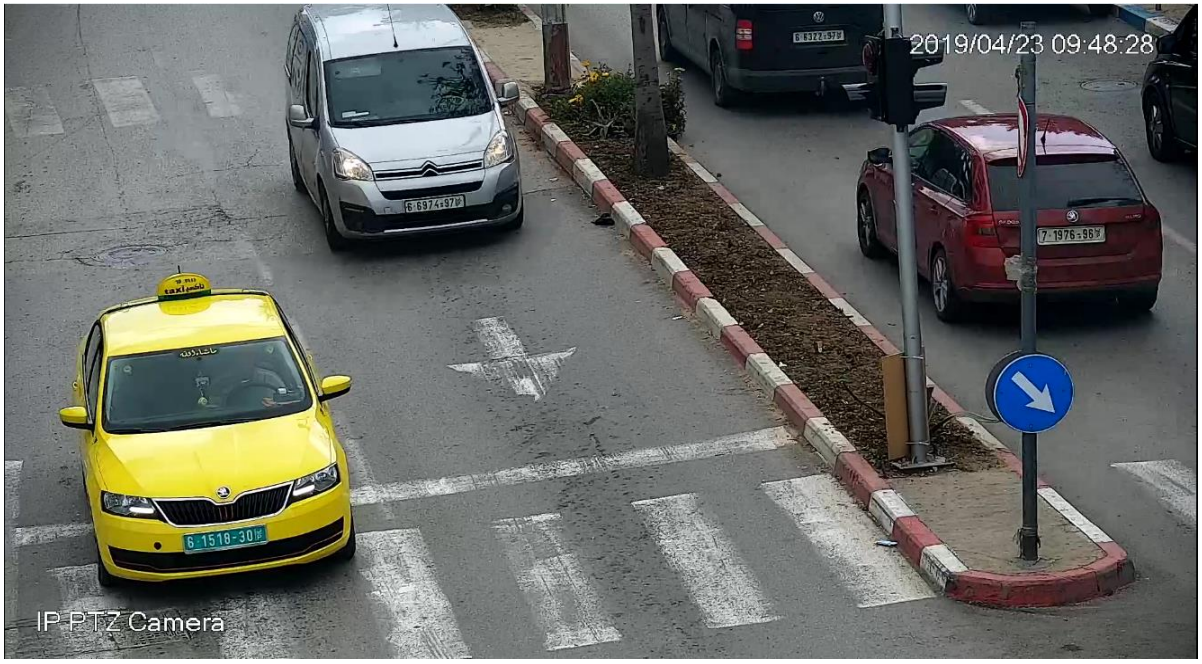

Figure 6-1: Sample Frame from Palestinian Dataset

Figure 6-2: Sample Vehicles from Palestinian Dataset

The overall collected dataset contained license plates from different Palestinian cities including the West Bank and the 1948 boundaries. The plates in West Bank are white and green, and the plates of 1948 boundaries are yellow. Information about the patterns of the Palestinian plates coding was provided by Al-Bireh municipality. The different coding formats for different --including before 1999, after 1999 and after 2017 are shown in Appendix A.

For the annotation of plates, a bounding box around the plate is required for the plate detection, and the text of the plate number is required for the recognition. For training a recognition model in specific, most OCR models require a bounding box around each character in the plate number. However, other models such as Tesseract 4 require a line-based annotation, which means that bounding boxes are needed for each line in a page not for each character. Thus, the bounding box around the plate should be enough in the case of plate recognition.

For this system, the line-based annotation was used. A simple tool was built on python for the goal of annotating the extracted images. Sixteen students studying computer engineering at Birzeit University used this tool to accomplish the annotation process. Figure 6.3 shows different views of the built tool.

Figure 6-3: Views of Annotation Tool

Chapter 7 Models' Evaluations

This chapter aims to compare different implementations of detection, tracking and recognition algorithms. It explains how these algorithms are evaluated and discusses the evaluation results.

7.1 Vehicle Detection

This section introduces some metrics used in measuring the performance of detectors. Then, it explains the algorithm followed in the evaluation process and discusses the results.

7.1.1 Evaluation Metrics

The evaluation of object detection is a non-trivial task. Object detectors are usually evaluated by comparing the detected bounding boxes with the ground truth bounding boxes. Any association between detected boxes and ground truth boxes can be one of the five cases, zero-to-one, one-to-zero, one-to-one, many-to-one, and one-to-many associations [43]. Figure 7-1 illustrates these cases.

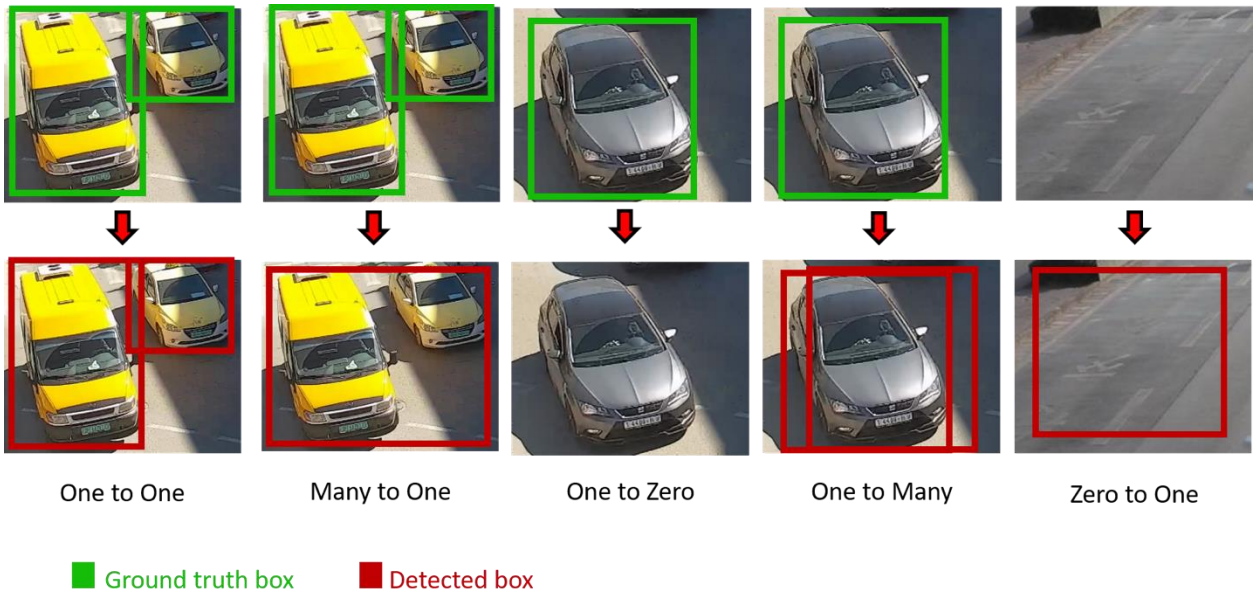

Figure 7-1: The Different Associations between Ground Truth Boxes and Detected Boxes

The correspondence between detected boxes and ground truth boxes is found by the overlap between them. The overlap between two boxes is defined as the area of their intersection over the area of their union as Equation 7-1 shows [44]. If the overlap or the

intersection over union (IOU) between two boxes is above some threshold, then they correspond to each other.

$$IOU\ (Box\ A, Box\ B) = \frac{area\ (Box\ A\ \cap\ Box\ B)}{area\ (Box\ A\ \cup\ Box\ B)} \qquad \textit{Equation 7-1}$$

After finding the correspondences between boxes in an image, three basic metrics are calculated. First, True Positive (TP), which means that the detector correctly predicted the existence of a vehicle in some region, i.e. there is a valid correspondence between a detected and a ground truth box. TP is represented by one-to-one association. Second, False Positive (FP), which means that the detector gave a false alarm indicating that there is a vehicle while there is not. FP implies that a detected box does not correspond to any ground truth box. FP is characterized by one-to-zero case. Third, when the detector fails to predict the existence of a vehicle, this is called False Negative (FN). In this situation a ground truth box does not correspond to any detected box. One-to-zero correspondence illustrates the FN situation.

Some associations are not simply described by one metric. For example, in the one-to-many situation, one of the detected boxes is considered TP while the others are marked as FP. It is bad for the detector to tell that there are more objects in an image than there really are. A more complex situation is many-to-one. In this case it is hard to tell which metric describes the best. It all depends on the IOU, each ground truth box is evaluated separately, if the IOU between the ground truth box and a detected box is larger than the threshold, it is TP. Otherwise, it is FN. If none of the ground truth boxes corresponds to a detected box, then it is also FP.

It should be mentioned that a fourth metric is usually used in evaluating machine learning techniques. This metric is the true negative. It indicates that a classifier correctly classified a negative case. For example, a non-vehicle image classified as non-vehicle. However, this metric is valid only for classification not for detection. It makes no sense in the context of detection, so it is always set to zero.

After calculating these basic metrics, more indicative metrics can be computed. One of these metrics is precision. Precision is the probability that a detected vehicle is really a

vehicle. Another metric is recall. Recall is the ratio of vehicles that the detector was able to detect. Additionally, F-measure is a metric that combines both precision and recall. Equations 7-2, 7-3 and 7-4 show how these metrics are calculated [44].

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \textit{Equation 7-2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \textit{Equation 7-3}$$

$$\text{F}_{\text{Measure}} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \qquad \textit{Equation 7-4}$$

A more comprehensive metric is the average precision AP. AP involves precision, recall, detection confidence, and IOU. Involving IOU gives an indication of the localization accuracy. Computing AP includes several steps [44]:

1. Sort all the detected boxes in all images in the dataset by their confidence increasingly.
2. Classify every box as correct (TP) if it corresponds to any ground truth box (IOU > threshold) and not correct otherwise.
3. For each detected box calculate the cumulative precision and recall. Cumulative precision is the proportion of the correctly detected boxes from the detected boxes so far. Whereas cumulative recall is the proportion of the correctly detected boxes so far from all ground truth boxes.
4. Plot the precision-recall curve.
5. Smooth the precision-recall curve by replacing the precision values in every recall interval by the maximum precision value in that interval as shown in Figure 7-2.
6. Calculate the area under the smoothed curve to finally get the AP.

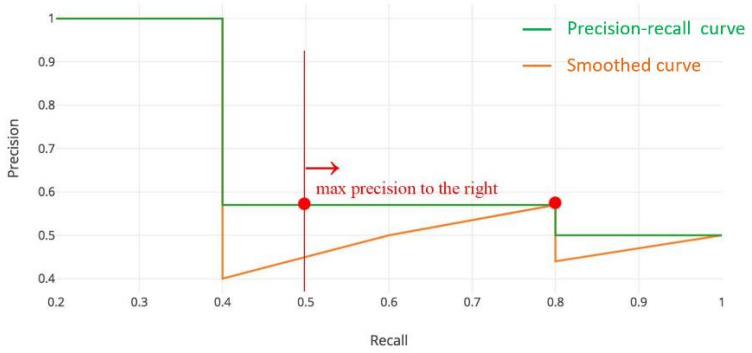

Figure 7-2: Precision-Recall Curve [44]

Besides the performance metrics discussed, since detectors are complex neural networks that are computationally expensive, memory and CPU consumption should be measured.

7.1.2 Evaluation Process

For the purpose of evaluating the detectors, they are run on every image of the dataset. The IOU values between the detected bounded boxes and the ground truth boxes in the image are calculated. A detected box and a ground truth box are assigned to each other whenever the IOU is larger than a specific threshold. True positive is incremented when a detected box is assigned to a non-previously assigned ground truth box. False positive is incremented when a detected box is assigned to a previously assigned ground truth box, and when a detected box is not assigned to any ground truth box. Finally, False negative is the number of ground truth boxes that are not assigned to any detected box. Figure 7-3 illustrates this process in detail. After that, precision, recall, F-measure and AP are calculated as was shown previously in section 7.1.1.

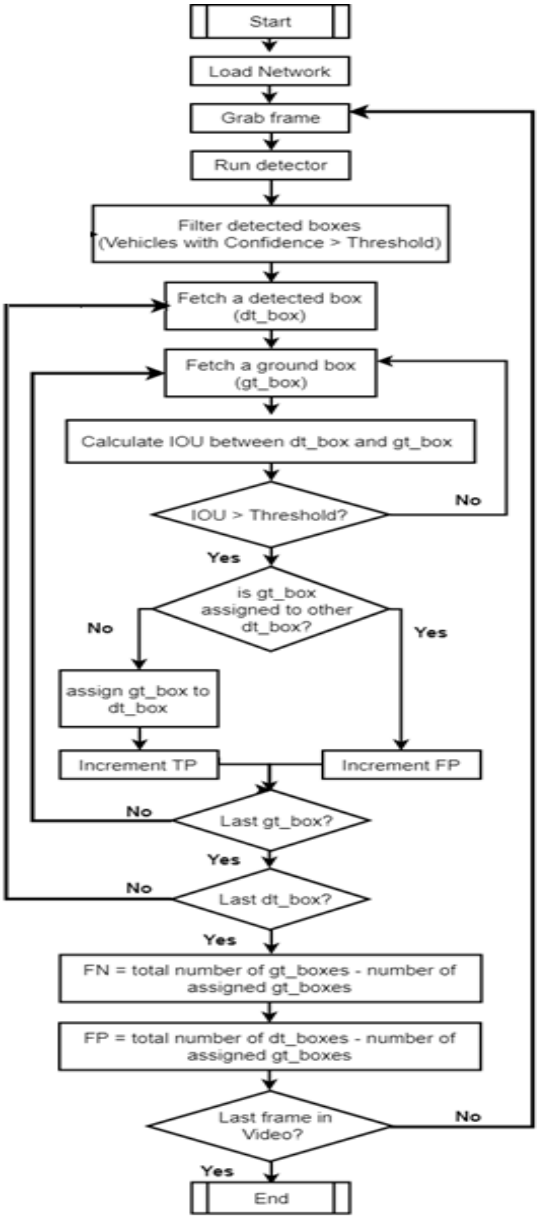

Figure 7-3: Algorithm for Detectors Evaluation

7.1.3 Evaluation Results

Three pretrained detection models from GluonCV were evaluated on UA-DETRAC dataset. GluonCV is a toolkit that provides implementations of deep learning algorithms in computer vision [45]. The evaluated models are YOLO, SSD and RCNN. They are pretrained on the Microsoft COCO (Common Objects in Context) dataset, which consists of 328k images containing 2.5 million labeled instances of 91 object types [46]. The results of this evaluation are shown in Table 7-1.

Table 7-1: Results of Detectors Evaluation

Detector	Recall %	Precession %	F-Measure %	AP %	Time (Sec/image)
YOLO	81.64	91.37	86.23	67.40	2.78
SSD	76.11	82.95	79.38	60.76	1.58
RCNN	90.04	68.88	78.05	70.6%	41.24

From Table 7-1, it can be noticed that YOLO and RCNN give the best performance, whereas SSD gives the best speed. Although RCNN offers an excellent performance, it is very slow and inappropriate for real time applications. It can be concluded that YOLO gives a good trade-off between performance and resources consumption and thus will be used in this system.

7.2 Vehicle Tracking

This section introduces some evaluation metrics used in trackers evaluation. It also illustrates the algorithm followed in this evaluation and analyses its results.

7.2.1 Evaluation Metrics

Besides the metrics introduced for detection evaluation, there are another two important metrics that measure the performance of trackers: Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP). MOTA measures the tracker errors in object configurations including false positives, false negatives and mismatches. Mismatches occur when a tracker swaps the IDs of two objects when they are close to each other in the trajectory. On the other hand, MOTP represents the tracker error in estimating the positions of the objects. MOTP computes the average intersection between the correctly matched pairs (ground truth boxes with detected boxes). MOTA and MOTP can be calculated using Equation 7-5 and Equation 7-6 respectively [47].

$$\mathrm{MOTA} = 1 - \frac{\sum \mathrm{FN} + \mathrm{FP} + \mathrm{Mismatches}}{Ground\ Truth\ Boxes\ Count} \qquad \textit{Equation 7-5}$$

$$\text{MOTP} = \frac{\sum \text{IoU}}{\text{TP}}$$ *Equation 7-6*

7.2.2 Evaluation Process

Trackers evaluation cannot be done on a complete video sequence. That's because as said earlier trackers cannot find new objects but track previously located ones. Thus, trackers evaluation is done on chunks of the video consisting of n frames. That means new trackers are fed with the ground truth boxes at the first frame of each chunk. Then, the basic measures (TP, FP, FN, and mismatches) are calculated independently for that chunk.

The evaluation process of every chunk or piece of a video is done by the following steps [47]:

1. A map of ground truth IDs and boxes (MAP < ID, Tracked_Box >) is initialized in the first frame. A tracker is also initialized for every box in the map.
2. For every frame, the boxes in the map are updated by the trackers.
3. After that, it is checked whether the correspondence between every <ID,Tracked_Box> pair in the map is still valid. By that it is meant that the IOU between the Tracked_Box and the ground truth box with the same ID is greater than some threshold. True positive measure is increased whenever a valid pair is found.
4. For the nonvalid pairs, the system tries to find correspondences with ground truth boxes that have IDs different from their IDs. These ground truth IDs should not be corresponded to any box in the previous step. Whenever a correspondence is found the true positive and the mismatch measures are increased. That's because the tracker found the vehicle but mistakenly changed its ID. Moreover, the old IDs mapped to these tracked boxes are replaced with the ground truth IDs.
5. All the pairs in the map for which no correspondence was found are counted as false positive.
6. All ground truth IDs that are not in the map are counted as false negative.
7. Steps 2-6 are repeated for every frame until the end of the chunk.

After all chunks are evaluated total TP, FP, FN and mismatch is calculated. Then, MOTP and MOTA are computed by the equations shown in the previous section.

7.2.3 Evaluation results

Four trackers from OpenCV were evaluated: KCF, MEDIANFLOW, DLIB and CSRT. They were evaluated on different chunk sizes (skip frames). Figures 7-4 and 7-5 show the evaluation results[1]. According to these results, trackers can be ordered by their MOTP and MOTA values as MEDIANFLOW, DLIB, CSRT then KCF. For the speed, as Table 7-2 shows, KCF and DLIB trackers give the best result. Based on these evaluations, MEDIANFLOW has the best performance but the slowest. DLIB is a good compromise between performance and speed. While KCF has the worst performance and as fast as DLIB and thus excluded.

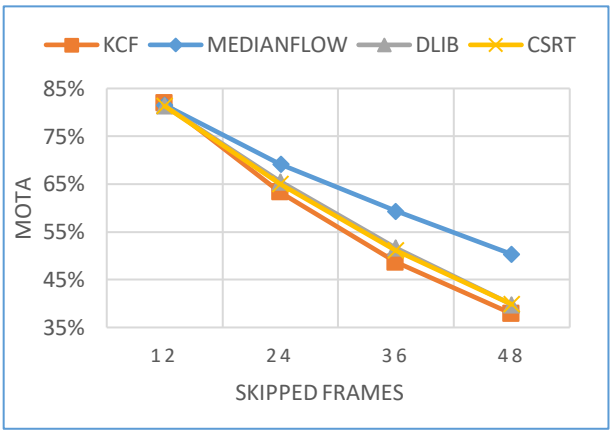

Figure 7-4: The Relation Between MOTA and the Number of the Skipped Frames for Different Trackers

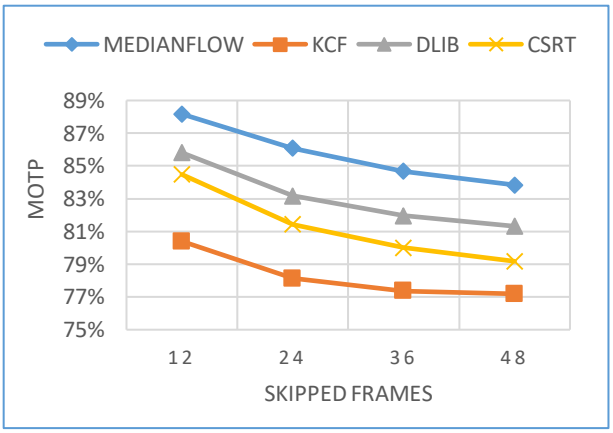

Figure 7-5: The Relation Between MOTP and the Number of the Skipped Frames for Different Trackers

Table 7-2: Running Time Per Image for Different Trackers

Tracker	MEDIANFLOW	DLIB	CSRT	KCF
Time (msec/ image)	226.56	71.88	226.56	70.35

[1] For detailed numbers refer to Table B-1 in Appendix B

7.3 Vehicle Detection and Tracking

In this system, detectors and trackers will work alternatively on the captured frames. Thus, it is important to evaluate them when they work together. The evaluation metrics used for this purpose are the same metrics used for evaluating trackers: MOTA and MOTP. This section will describe the evaluation process used to evaluate a pair of a detector and a tracker. Then, it will show the evaluation results of some pairs of the detectors and the trackers.

7.3.1 Evaluation Process

Evaluating the frames where the tracker works is not different by any way from trackers evaluation in the previous section. For the frames where the detector works, many cases should be considered when mapping the detected boxes to the previously tracked IDs. Figure 7-6. illustrates these cases and how they should be evaluated.

As Figure 7-6 shows that there are eight different paths or cases to consider. In path (1) for example, if the detected box is matched to a tracked box and to a ground truth box, then it should be checked if the IDs of the tracked box and the ground truth box are equal. If they are not equal, then an ID mismatch has happened. However, the true positive measure is also increased since a correspondence with a ground truth box was found.

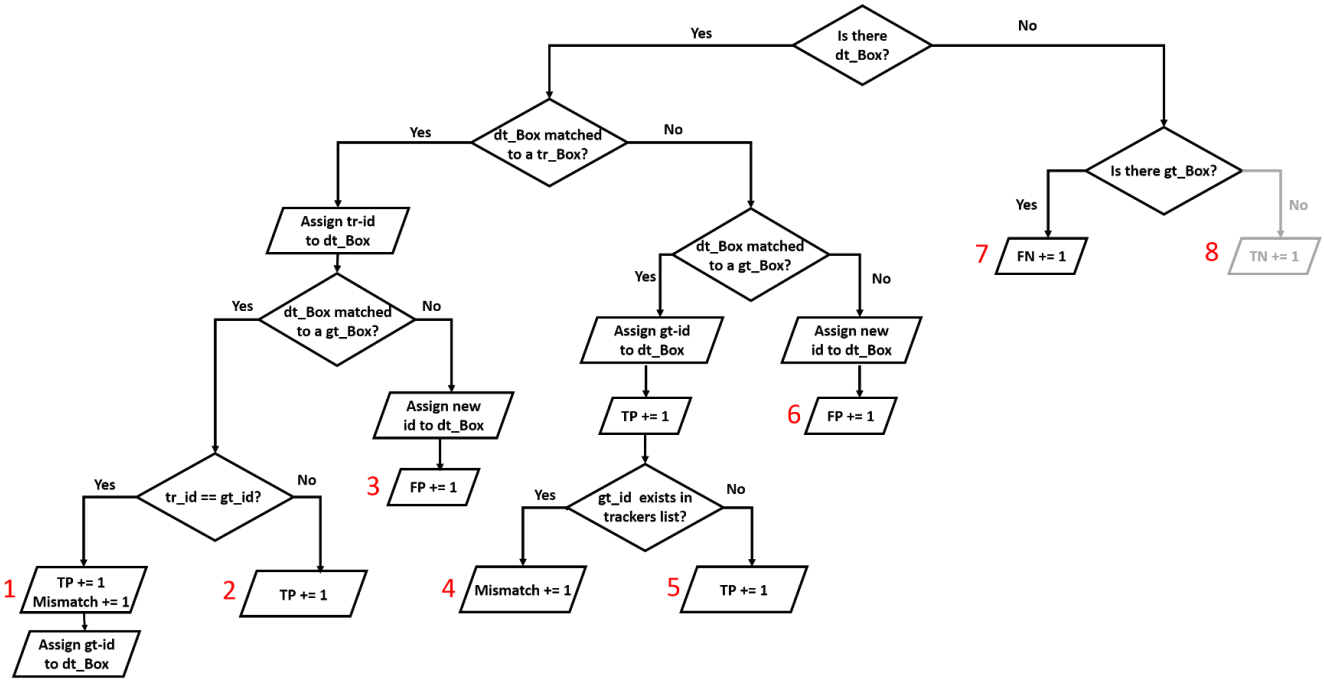

Figure 7-6: The Different Cases That Happen When Evaluating Detectors and Trackers

7.3.2 Evaluation Results

According to the previous evaluations, the best of the evaluated detectors and trackers were chosen for the evaluation in this step. YOLO was chosen for detection because of its convenient performance and speed. For tracking, MEDIANFLOW, DLIB and CSRT were chosen as the best trackers. Figures 7-7 and 7-8 show the evaluation results[2]. It can be noticed that YOLO-DLIB and YOLO -CSRT combinations have the highest MOTA values, which means that they make less mistakes (FP, FN and mismatch) than the other combinations. However, YOLO-MEDIANFLOW have the best MOTP values, which means that they are the most precise in localizing the object. Since MOTP values are very good and close for all combinations, it is reasonable to judge these combinations by their MOTA values were the difference between them is noticeable. Therefore, YOLO-DLIB and YOLO-CSRT are the best combinations to be used in the system.

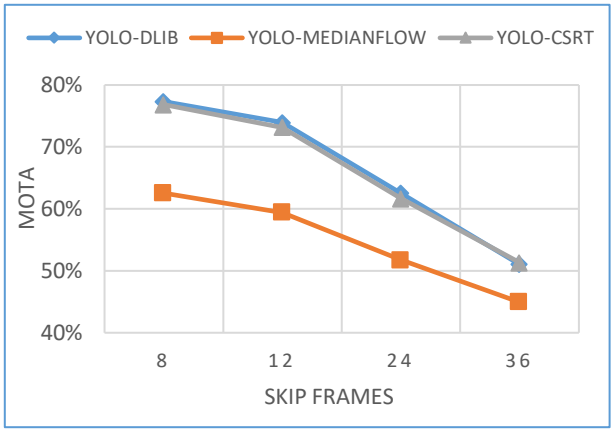

Figure 7-8: The Relation Between MOTA And the Number of the Skipped Frames for Different Detectors and Trackers

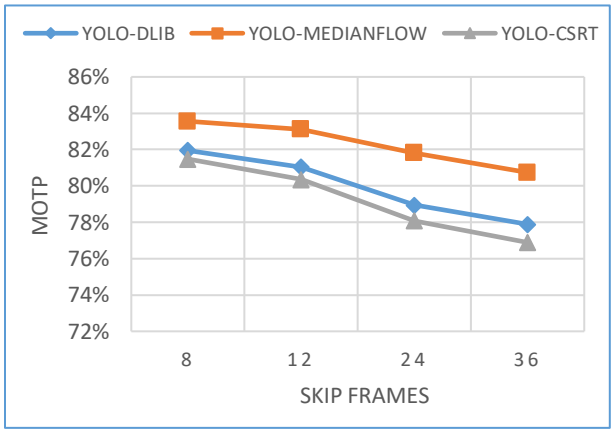

Figure 7-7:The Relation Between MOTP And the Number of the Skipped Frames for Different Detectors and Trackers

7.4 Plate Detection

As well for plate detection, different approaches were applied to reach good results. First, the Cloud APIs, Sighthound and Plate Recognizer, mentioned previously were evaluated on the test data of the Palestinian plates' dataset. It should be mentioned that both APIs are subscription paid and not trained on Palestine's Plates.

[2] For detailed numbers refer to Table B-2 in Appendix B

Second approach was training a model using the train data of Palestinian dataset which consisted of 1580 image using Darknet YOLOv3 network [48]. The training process continued for 20000 patches with average loss or error less than 0.025%. To avoid overfitting problem on the training data, the model was extracted at 5000 patches with loss error about 0.1%. Figure 7-9 shows the loss error through the process of training.

Table 7-3 shows the results of evaluation for the different approaches. It is clear that the performance of the three detectors is pretty good with small differences.

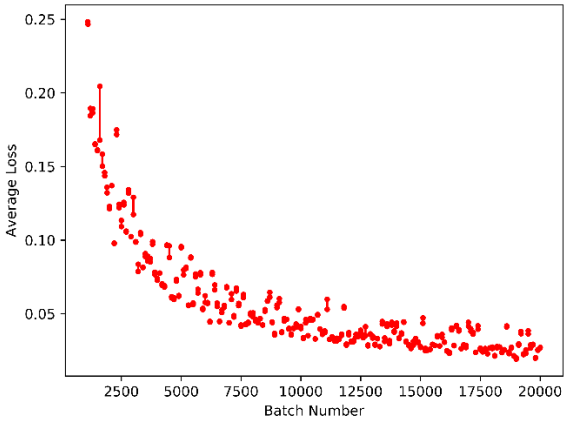

Figure 7-9: Loss Error through YOLO Training Process

Table 7-3: Results of Plate Detection

Detector	Recall %	Precision %	F-Measure %
Sighthound API	98.96	96.96	97.85
Plate Recognizer API	99.69	96.41	98.02
Trained Model	98.76	94.38	96.52

7.5 Plate Recognition

7.5.1 Evaluation Metrics

The metrics considered for assessment of plate recognition methods are both on plate-level and character-level. On plate level, the percentage of plates with all characters of it are detected correctly divided by the total number of plates is calculated. In terms of OCR metrics, this is called strict word metric. For clarity, it will be called plate accuracy. On the other hand, character accuracy metric is based on the Levenshtein distance which represent the minimum number of edit operations (character insertions, deletions, and substitutions) between the ground truth and the recognition result. These metrics are represented by Equations 7-7, 7-8 and 7-9 [49].

$$Plate\ Accuracy = \frac{Number\ of\ Plates\ Detected\ Correctly}{Total\ Number\ of\ Plates} \qquad \textit{Equation 7-7}$$

$$\text{Character Accuracy} = \frac{\text{Total Number of characters} - \text{Total Character Error}}{\text{Total Number of Characters}} \qquad \textit{Equation 7-8}$$

$$\text{Total Character Error} = (\text{Error}_{\text{insert}} + \text{Error}_{\text{delete}} + \text{Error}_{\text{replace}}) \qquad \textit{Equation 7-9}$$

7.5.2 Cloud APIs

The cloud APIs provide both plate detection and then character recognition. The results of their recognition are shown in Table 7-5. Without any post processing, it is obvious that the plate accuracy is low compared to the character accuracy. After analyzing the cause of the error rates, it was noticed that many errors were produced because of adding one or two letters to the end of the plate number. This happens due to the unclear symbol "$\left|\frac{\mathbf{P}}{\text{ف}}\right.$" at the right of the Palestinian plates which is not considered as a part from the plate number. Therefore, the recognized numbers were cropped to match the lengths of the Palestinian plates' patterns. As Table 7-5 shows, the performance of Sighthound was significantly improved whereas the improvement on Plate Recognizer was slight.

According to Appendix A, alphabets are included in Palestinian plates only as the right most character in plates licensed after 2017. However, the APIs are trained on different

patterns of plates for many countries. So, the results of recognition included different alphabet characters in places that are expected to be digits. This happens due to the shape similarity between some digits and alphabets (e.g. O and 0). From this point, a statistical analysis was made to identify each character usually replaces which digit. These statistics were then used to do some replacements. Table 7-4 shows the results of the statistics and the replacements made based on them. As Table 7-5 shows both recognizers were significantly enhanced after the replacement process.

When comparing the two APIs after the cropping and replacing processes, it can be noticed that Sighthound has better performance although both of them have a really good and close performance.

Table 7-4: Statistics for Character Replacements in Cloud-APIs Recognition Results

Alphabet	Digit	Frequency	Digit Replacement
B	6	278	6
	8	15	
	9	3	
C	6	4	6
D	0	3	0
G	0	1	6
	6	180	
I	1	3	1
Q	0	26	0
	3	1	
	8	1	
S	5	5	6
	6	22	
	9	3	
T	1	10	1
	7	1	
U	0	7	0
Z	2	12	2
	7	5	

Table 7-5: Results of Plate recognition of Cloud APIs

API	No Post Processing		After Cropping		After Replacement	
	Plate Accuracy %	Characters Accuracy %	Plate Accuracy %	Characters Accuracy %	Plate Accuracy %	Characters Accuracy %
Sighthound	41.49	88.61	78.95	95.74	88.54	97.58
Plate Recognizer	70.28	94.62	75.85	95.92	92.26	98.57

7.5.3 Tesseract Trained Model

The Open-source OCR engine of Tesseract was used to train a model for character recognition of plates. The training process took around 20000 iterations with Root Mean Square Error (RMS) about 0.02%. First, the model was trained on the input images without any preprocessing but the results were unacceptable as shown in Table 7-6. Tesseract suggests to do some preprocessing steps before passing the input images to the model, which may result in reducing the error rate. Therefore, some preprocessing operations were applied on the train and test dataset of plates, including the following steps:

1. Resizing – for tesseract, the size of the input images is preferred to be unified, so all the images were resized to an average size for plates (130 X 60) px.
2. Grayscale – images were converted from 3 color-channels (RGB) to 1 channel.
3. Denoising – noise in the images was reduced by applying Gaussian and Fast Mean filters with proper parameters [50].
4. Adaptive Binarization – Adaptive Threshold Mean, where the threshold depends on neighborhood area, was used to convert images into black and white [51].
5. De-skewing or Rotation – images were rotated using Hough and Canny mechanisms so that the text is aligned horizontally [52].
6. Cropping black border – the black border that surrounds some plates may be read as additional characters so it was cropped.

7. Padding a white border – it is preferred for tesseract training to have a white border around the text, so a border of width 20px was added to the images.

Figure 7-10 illustrates a sample of applying the steps of preprocessing. The test results of the model trained on the preprocessed dataset is shown in Table 7-6. After applying the post process steps of cropping and replacing on both models, it has almost no effect on the model without preprocessing. While the cropping step on the trained model with preprocessing significantly increased the plate accuracy.

Table 7-6: Results of Plate Recognition of Trained Models

	No Post-Processing		**After Cropping**		**After Replacement**	
	Plate Accuracy %	**Characters Accuracy %**	**Plate Accuracy %**	**Characters Accuracy %**	**Plate Accuracy %**	**Characters Accuracy %**
Trained Model without Preprocessing	16.11	23.50	16.11	23.33	16.11	23.33
Trained Model with Preprocessing	62.19	90.86	73.75	92.71	73.75	92.71

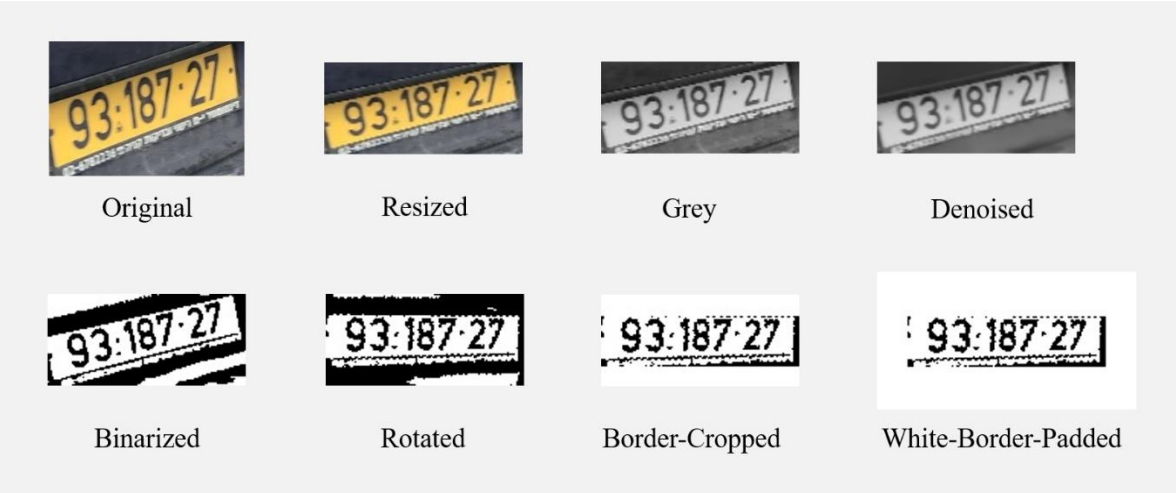

Figure 7-10: Samples of Different Steps of Preprocessing

Chapter 8 Rate-Accuracy Experiments

In this chapter, the relation between streaming bitrate and the performance of some models is investigated. The best of the detectors, trackers and plate recognizers evaluated earlier are experimented under different variations in streaming quality. These variations include encoding videos with different frame-rate and QP values to find a good trade-off between the consumed bandwidth and the achieved accuracy.

8.1 Domain Shifts

Domain shift occurs when the model is trained on a dataset with specific characteristics that are different from the test dataset which matches the actual deployment scenario. Usually, domain shifting degrades the model performance. Thus, domain adaptation appeared as an important field in machine learning. A model is considered to be adaptive when it maintains its performance across various domain shifts [53].

Here, the adaptivity of some models is measured when applying domain shifts in the frame rate and the quantization of the dataset videos. FFmpeg tool is used to generate these shifts. FFmpeg is a multimedia framework that is able to decode, encode, transcode, mux, demux, stream, filter, and play videos with various user defined parameters [54]. Since one of the main objectives of this work is to find a trade-off between the bitrate and the accuracy of some models, the relations between these domain shifts and the bitrate of the UA-DETRAC videos was evaluated. Figure 8-1 and Figure 8-2 show these relations[3]. It is worth mentioning that these relations apply when the videos are encoded by H.264 standard. The trend is not expected to change if other codecs are used.

From Figure 8-2, it can be noticed that increasing QP decreases the bitrate of the video. It is observed that when QP is small, its change significantly affects the bitrate. However, when QP increases to values larger than 40, the bitrate decreases slowly. On the other hand, Figure 8-1 shows that reducing the frame rate of a video reduces its bitrate. Comparing between the two figures, it can be concluded that the effect of the frame rate of a video on its bitrate is less than the QP effect, it is almost linear.

[3] For detailed numbers refer to Table C-1 in Appendix C

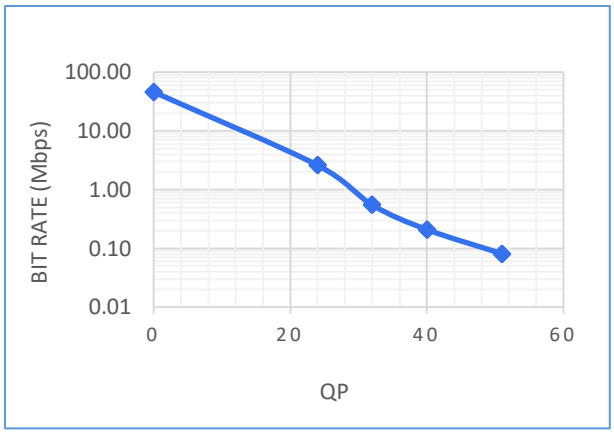

Figure 8-2: The Relation Between the Bit Rate and the Quantization of UA-DETRAC Videos When the Frame Rate is 25 FPS

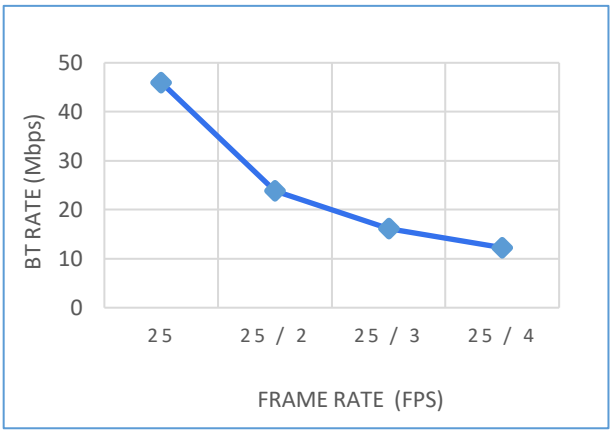

Figure 8-1: The Relation Between the Bit Rate and the Frame Rate of UA-DETRAC Videos When the Quantization Parameter is 0

8.2 Vehicle Detection

The videos of the dataset were encoded with different QP values. Then, YOLO and SSD detectors were evaluated on them. Figure 8-3 and Figure 8-4 show the evaluation results on these QP values[4]. From these figures, it can be observed that both YOLO and SSD detectors show the same behavior as QP value increases. Moreover, it is noticed that when QP is small, its change has a very small effect on the performance although it has a large effect on the bitrate as said earlier; Increasing QP from 0 to 32 reduces the bitrate to 80% of its value with less than 5% drop in the performance metrics. On the other hand, for large QP values the performance decreases rapidly as QP increases although the bitrate decreases slowly.

For the frame rate, it does not make sense to study its effect on detection performance. That's because detectors run on frames by themselves without relating them to any other frames.

[4] For detailed numbers refer to Table C-2 in Appendix C

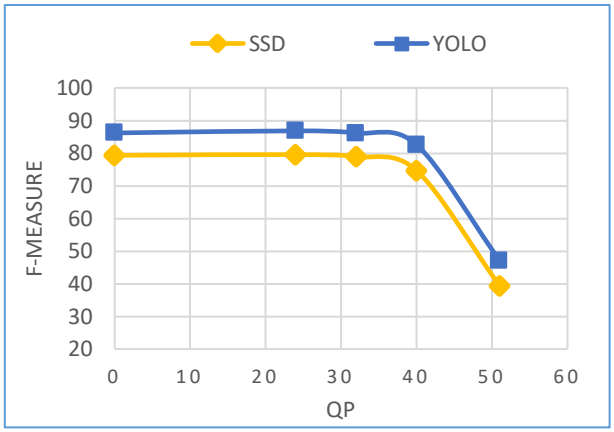

Figure 8-4: The Relation Between the F-Measure of Vehicle Detection and the Quantization of UA-DETRAC Videos When the Frame Rate is 25 FPS

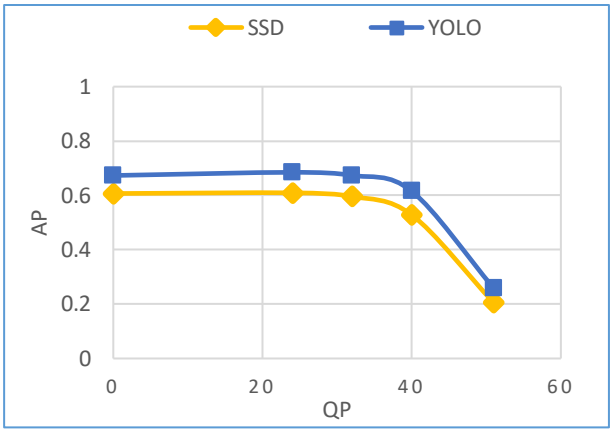

Figure 8-3: The Relation Between the AP of Vehicle Detection and the Quantization of UA-DETRAC Videos When the Frame Rate is 25 FPS

8.3 Vehicle Tracking

CSRT, DLIB and MEDIANFLOW trackers were evaluated on videos encoded with different frame rate and QP values. Figures 8-5 and 8-6 show how MOTA and MOTP metrics vary with QP when the skip frames parameter is 4 and 8 [5]. It can be seen that the performance of the trackers is similar to the detectors; It is almost steady when QP is smaller than 40. However, when QP is larger than 40, different trackers are affected differently. MEDIANFLOW is the most affected. DLIB maintains its MOTP performance but not the MOTA. CSRT seems to be the most robust. Moreover, it is observed that the performance is affected more when the skip frames parameter is 8.

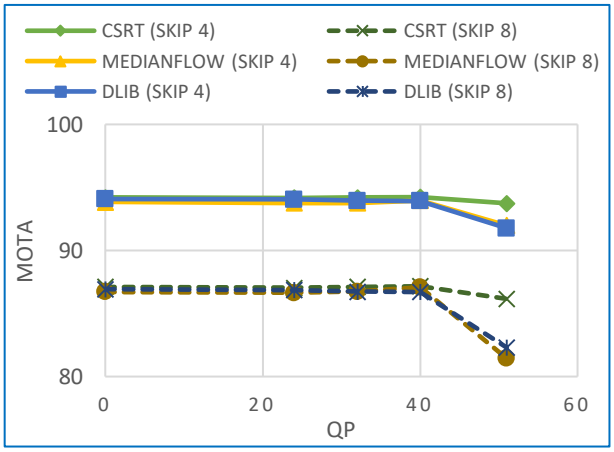

Figure 8-5: The Relation Between the MOTA of Vehicle Tracking and the Quantization of UA-DETRAC Videos When the Frame Rate is 25 FPS

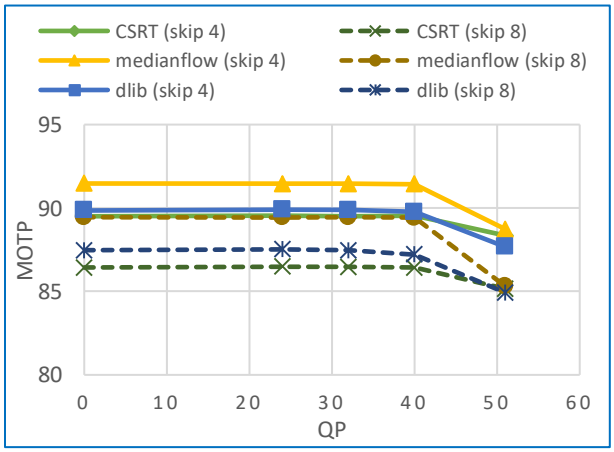

Figure 8-6: The Relation Between the MOTP of Vehicle Tracking and the Quantization of UA-DETRAC Videos When the Frame Rate is 25 FPS

[5] For detailed numbers refer to Table C-3 and Table C-4 Appendix C

For the frame rate, Figure 8-7 and Figure 8-8 show how MOTP and MOTA metrics change when the frame rate changes for two values of the skip frame parameter (4 and 8) [6]. It is observed that trackers performance is more affected by the frame rate than QP. Additionally, the frame rate affects MOTA more than MOTP. For MOTP, the three trackers have the same behavior as the frame rate decreases. On the other hand, MOTA is affected differently for different trackers. DLIB is the worst affected. MEDIANFLOW behavior is linear when skip frames parameter is 4. However, when skip frames parameter is 8 its MOTA decreases in a nonlinear way (yet better than DLIB). CSRT behavior is always linear.

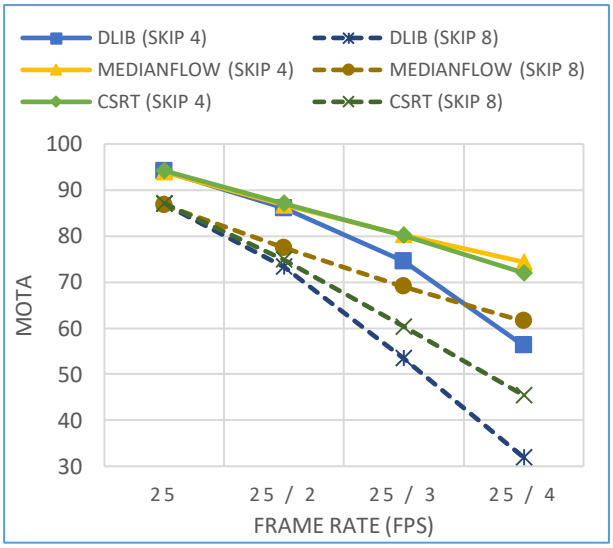

Figure 8-8:The Relation Between the MOTA of Vehicle Tracking and the Frame Rate of UA-DETRAC Videos When the Quantization Parameter is 0

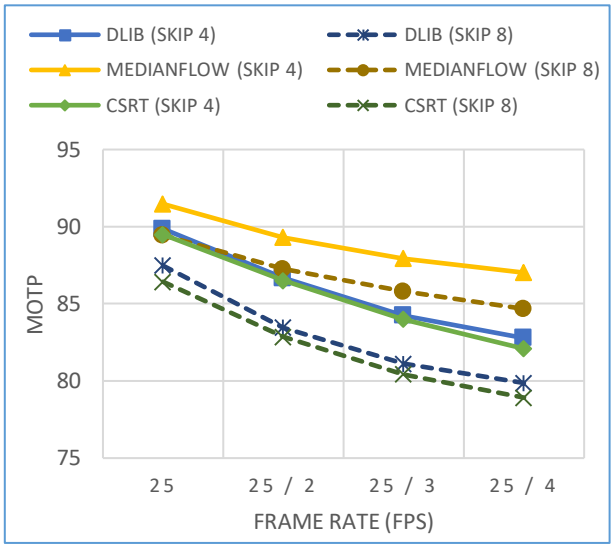

Figure 8-7: The Relation Between the MOTP of Vehicle Tracking and the Frame Rate of UA-DETRAC Videos When the Quantization Parameter is 0

8.4 Vehicle Detection and Tracking

Some combinations of the best detectors and trackers were evaluated with different variations in the frame rate and the QP. These combinations are YOLO-DLIB, YOLO-CSRT and YOLO-MEDIANFLOW. Figure 8-9 and Figure 8-10 show some of these evaluations results[7]. It should be mentioned that since the previous experiments showed that the performance significantly degrades for QP values larger than 40, these values were excluded. Moreover, sine decreasing QP to values smaller than 24 resulted in much larger bitrate but

[6] For detailed numbers refer to Table C-5 and Table C-6 Appendix C
[7] For detailed numbers refer to Table C-7 in Appendix C

unnoticeable improvement in the performance, these values were excluded too. It is obvious from the two figures that MOTA is more affected than MOTP; MOTP is almost steady. Additionally, the effect of the frame rate is larger than QP; QP effect is very small. Moreover, it can be noticed that YOLO-DLIB combination is the worst affected by the frame rate. Whereas YOLO-MEDIANFLOW and YOLO-CSRT have similar behavior. On the other hand, all combinations have the same behavior when QP changes.

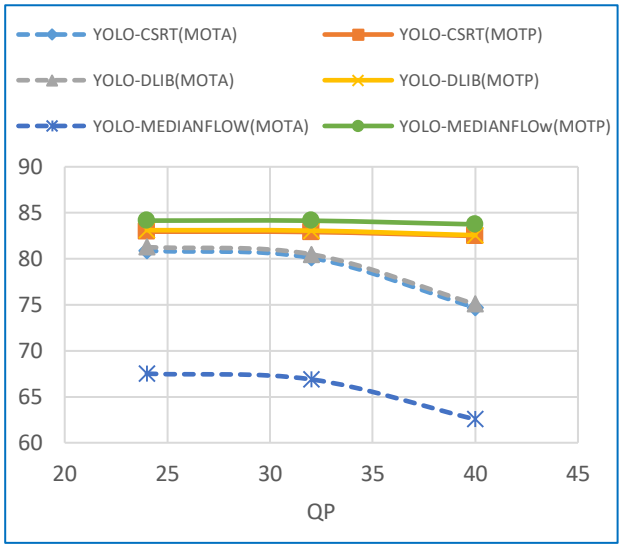

Figure 8-10: The Relation Between the MOTA and MOTP of Vehicle Detection and Tracking versus the Quantization of UA-DETRAC Videos When the Frame Rate is 25

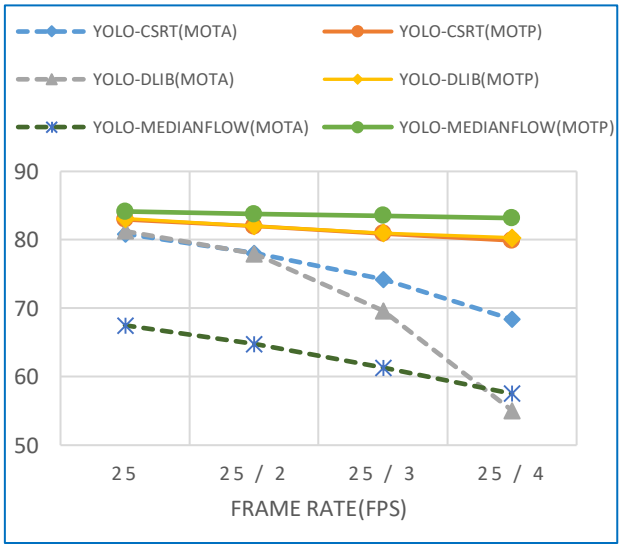

Figure 8-9: The Relation Between the MOTA and MOTP of Vehicle Detection and Tracking versus the Frame Rate of UA-DETRAC Videos When the Quantization Parameter is 24

Finally, the relation between the performance metrics and the bitrate values resulted from different combinations of the QP and the frame rate was investigated. Table 8-1 shows the bitrate values obtained by each experimented pair of the QP and the frame rate. Figures 8-11 and 8-12 show the MOTA and MOTP metrics at these bitrate values[8]. It can be seen that both MOTA and MOTP curves are not monotonic, they oscillate frequently as the bitrate increases. However, it does not make sense to encode a video that consumes higher bandwidth and gives less accuracy when processed. Thus, the points that cause decreasing in the curves were removed and optimal curves were plot (The continuous curves in Figures 8-11 and 8-12). From the optimal curves it can be concluded that encoding a video with bitrates larger than about 0.5 Mbps is meaningless. It does not come out with any improvement. Moreover, if the available

[8] For detailed numbers refer to Table C-7 in Appendix C

bandwidth is limited and 0.5 Mbps is still not affordable, then the bitrate can be reduced to 0.74 Mbps with tolerated drawbacks.

Table 8-1: Bitrate Values for Different Pairs of QP and Frame Rate

QP \ FPS	25	25 / 2	25 / 3	25 / 4
24	2.60	1.58	1.16	0.94
32	0.55	0.36	0.28	0.23
40	0.21	0.14	0.11	0.09

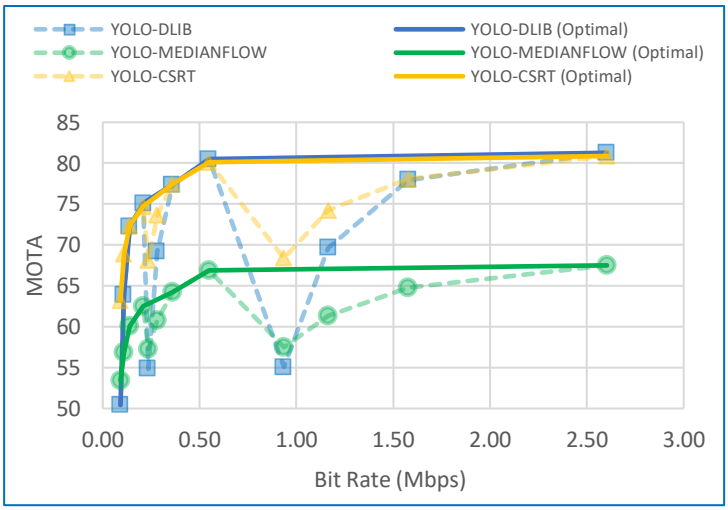

Figure 8-11: The Relation Between MOTA of Vehicle Detection and Tracking Versus the Bit Rate of Different Encodings of UA-DETRAC Videos

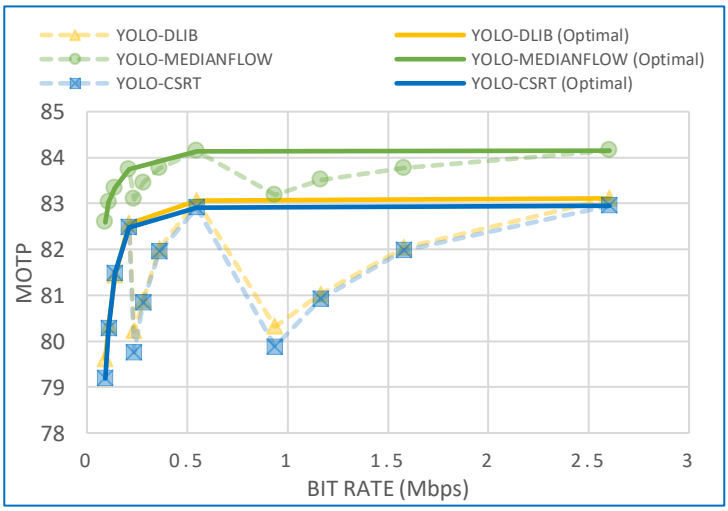

Figure 8-12: The Relation Between MOTP of Vehicle Detection and Tracking Versus the Bit Rate of Different Encodings of UA-DETRAC Videos

8.5 Plate Detection

The collected plate dataset was encoded with different QP values. Then, the three plate detectors: Plate Recognizer, Sighthound and YOLO, were tested on them. Figure 8-13 shows the testing results[9]. From this figure, it can be concluded that the three detectors are very adaptive to the QP changes. Actually, their performance achieves its best when QP equals 40.

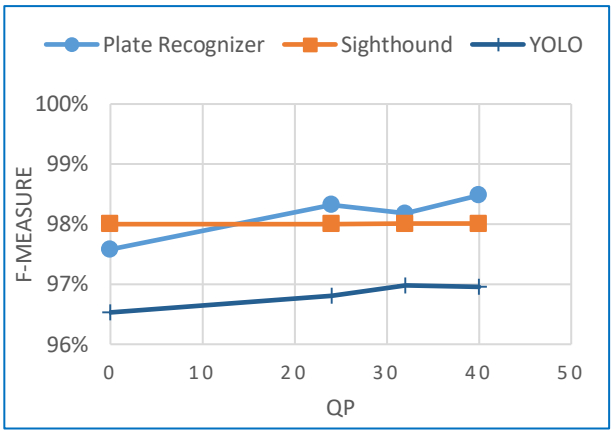

Figure 8-13: The Relation Between the F-Measure of Plate Detection and the Quantization of the Collected Palestinian Dataset When the Frame Rate is 25 FPS

8.6 Plate Recognition

The plate recognition adaptivity to the QP variations was also assessed. The three recognizers: Plate Recognizer, Sighthound and Tesseract, were evaluated with different variations in QP. Figures 8-14 and 8-15 show the plate recognition accuracy and the character recognition accuracy respectively. From these figures it can be said that the evaluated plate recognizers can maintain their performance as QP changes. Actually, the performance of Sighthound surprisingly increases as QP increases from 0 to 24.

[9] For detailed numbers refer to Table C-9 in Appendix C

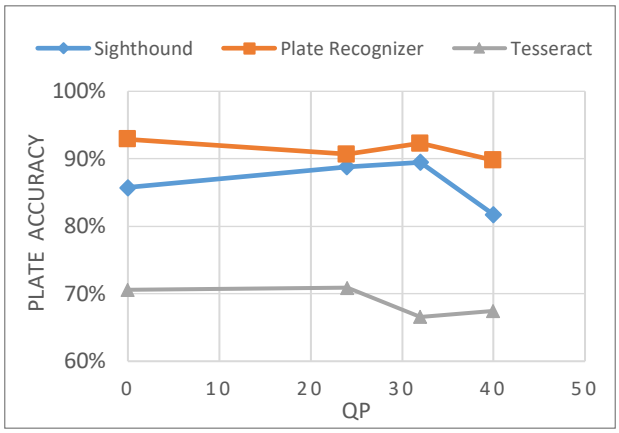

Figure 8-14: The Relation Between the Plate Accuracy of Plate Recognition and the Quantization the Collected Palestinian Dataset When the Frame Rate is 25 FPS

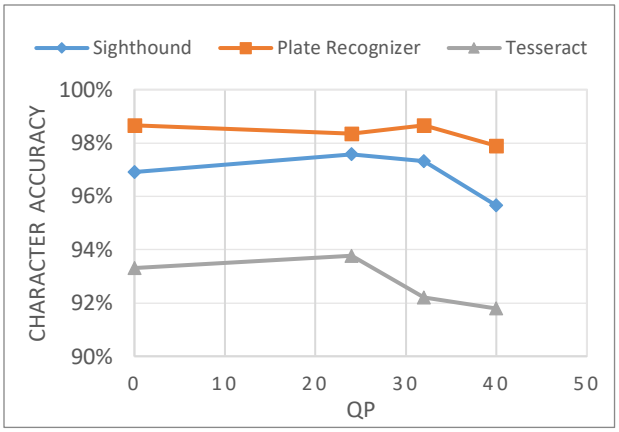

Figure 8-15: The Relation Between the Character Accuracy of Plate Recognition and the Quantization of the Collected Palestinian Dataset When the Frame Rate is 25 FPS

8.7 Savings in Link Cost

From the prior experiments, it can be concluded that a good trade-off between the bitrate of streaming and the performance of the image processing algorithms could be made. Actually, the used bitrate may be reduced to one fifth of its value almost without affecting the performance of these algorithms. However, the real essence of achieving this trade-off is the decrement in cost that results from reducing the bandwidth. The internet service prices offered by Paltel, which is a well-known telecommunication company in Palestine, are shown in Appendix D. According to these prices, supposing that the needed bandwidth was 100 Mbps, reducing it to one fifth will make the 50Mbps package sufficient. This leads to reducing the cost to 65% of its original value.

Chapter 9 Traffic Monitoring Application

In terms of proving the validity of the system proposed in earlier chapters, a surveillance web application for monitoring traffic was designed, developed, deployed and tested. The features of the system were discussed with the municipality of Ramallah city, in order to bring it into a real-life application. The first version of the application focused on generating various statistics and reports about the traffic passing through the streets where the cameras are adjusted. This includes studying congestion times and locations, monitoring types of vehicles (public transportations, private cars and motorcycles), and also monitoring the demographics and the population movement by identifying vehicles plate numbers that can indicate from which region or city are the vehicles owners.

This chapter will discuss the frameworks used in developing the system and deploying it and then the main features provided by the system.

9.1 Frameworks

9.1.1 Spring Boot

Spring Boot is an open source Java-based framework that provides developers with a platform to accelerate the process of developing applications. The main advantages for using Spring boot that it is flexible, auto-configurable and scalable. Moreover, it comes with wide support [55]. Spring Boot was used for developing the backend of the web application including the APIs for the communication with both frontend side and python side and also for managing the database.

9.1.2 Angular

Angular was used to develop the client-side of the application. Angular is a powerful and modern framework that supports multiple platforms including web, mobile and desktop development. It makes the process of developing applications more consistent, structured and easier to manage, debug and deploy [56].

9.1.3 Flask

Flask is one of the best Python web frameworks. It has almost no dependencies on external libraries. It is easy to learn, manage and has no restrictions on the structure of the app or data abstraction layers [57]. In the application, Flask was used for managing the communication between the python code where the system is actually built and the Spring Boot application that manages the database and the APIs.

9.1.4 Docker

Docker is a new tool designed for building, deploying and running the applications using containers. A container is used for isolating and packaging up all the parts of an application including the libraries and dependencies used, and shipping it as one package. This guarantees that the application can run on different environments and machines with no worries about the settings and configurations [58]. Docker was used for the purpose of efficiently packaging the components of the application, deploying it on the server and managing the resources used by it on running time.

9.5.1 MySQL

MySQL is a popular open source database management system. It uses relational model for representing entities where each entity is stored in a separate table. MySQL is reliable, scalable and easy to use and maintain [59]. Figure 9-1 shows the Entity Relational Diagram (ERD) of the system database.

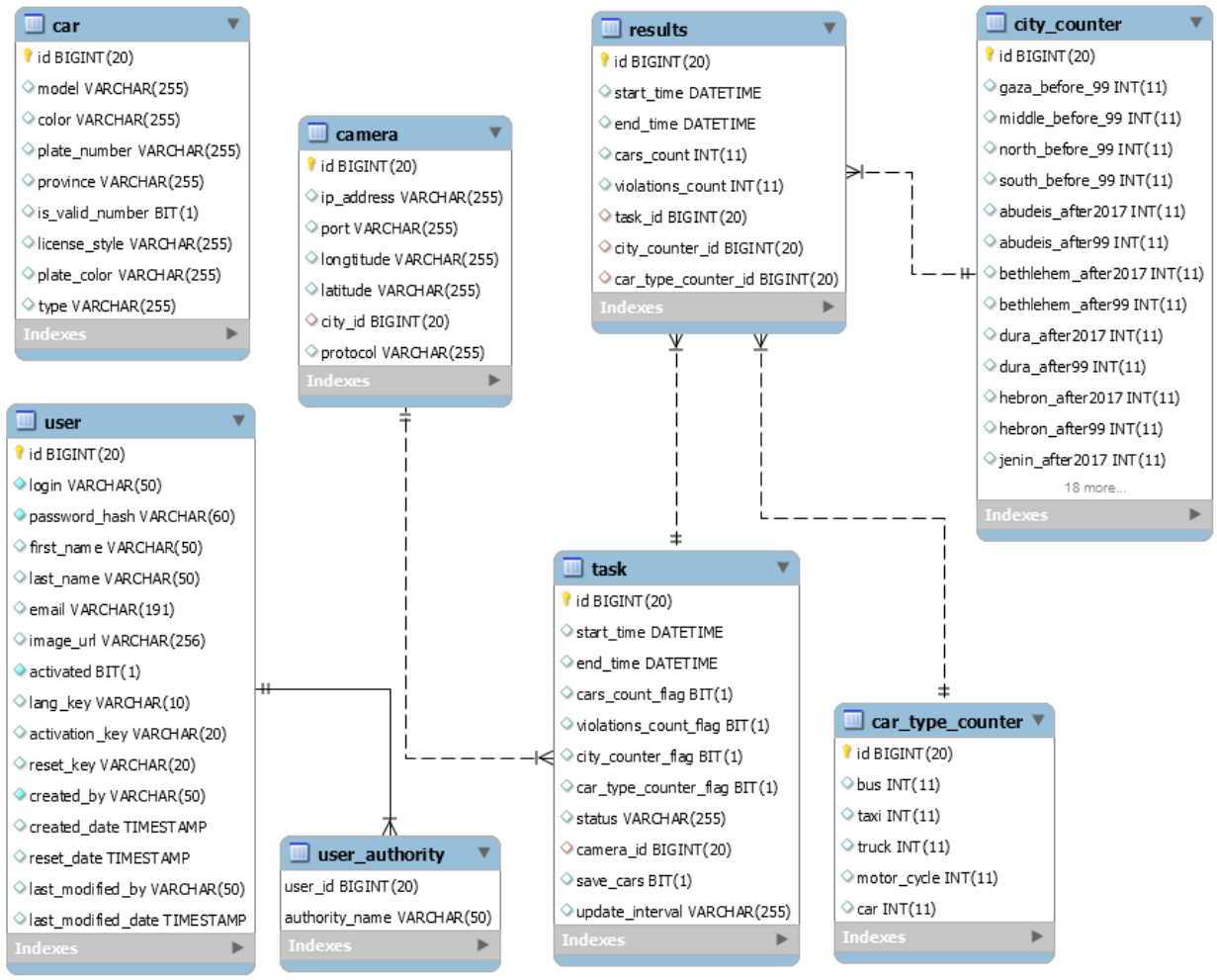

Figure 9-1: Entity Relational Diagram (ERD) of the System Database

Figure 9-2: The Frameworks Used in Developing the Application

9.2 Features and Views

Figures 9-3 to 9-8 show the main views of the application. The user can login to the system with authority either as admin or user to support different privileges in future work. The user

adds the cameras to the system by configuring the settings required to access them. Then, the user can start adding tasks or schedules. A task is a monitoring process for a defined interval of time by a chosen camera. While a task is running, the results are updated continuously per selected update interval to keep tracking the progress of the task. Through the running time, GPU together with multithreading are used to accelerate the processing and nearly achieve a real-time application. The tasks results are presented with different charts that indicate how the number of passing vehicles change through the targeted interval of time, the demographic distribution of vehicles which is determined by their plate numbers patterns, and the distribution of the passing vehicle types (cars, trucks, busses and motorcycles).

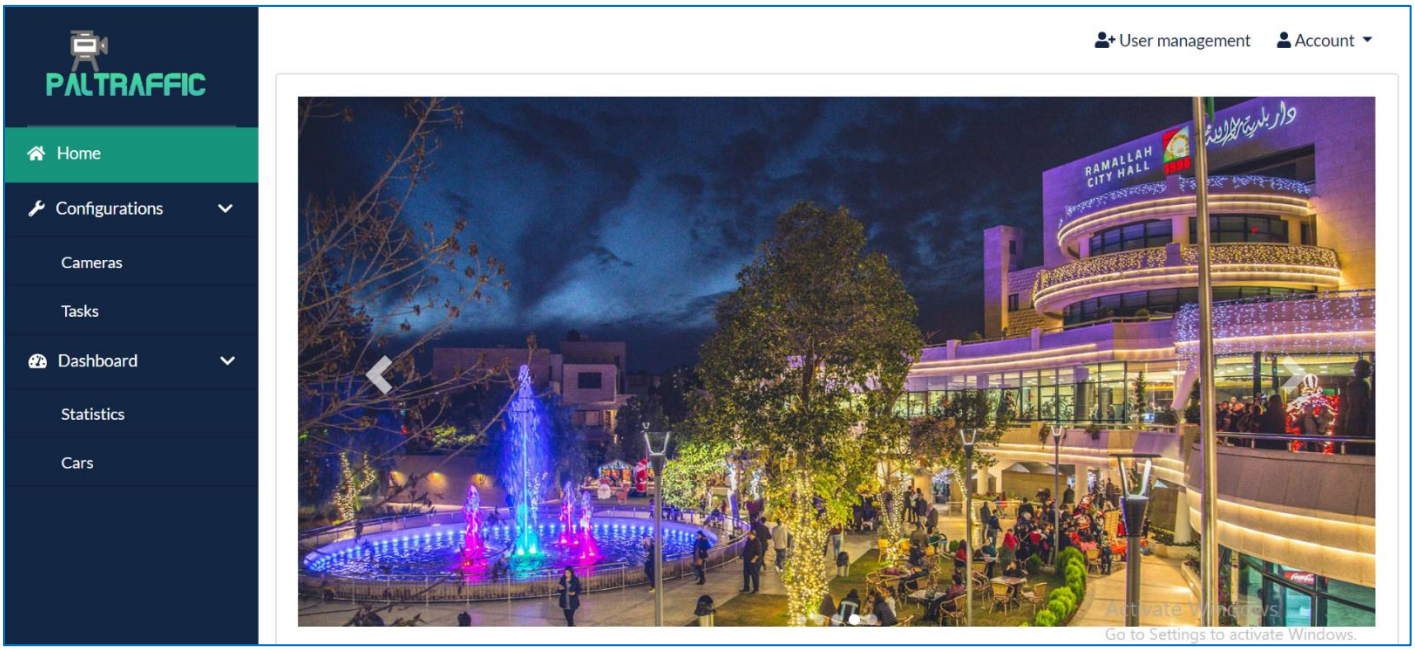

Figure 9-3: Home Page of the Application

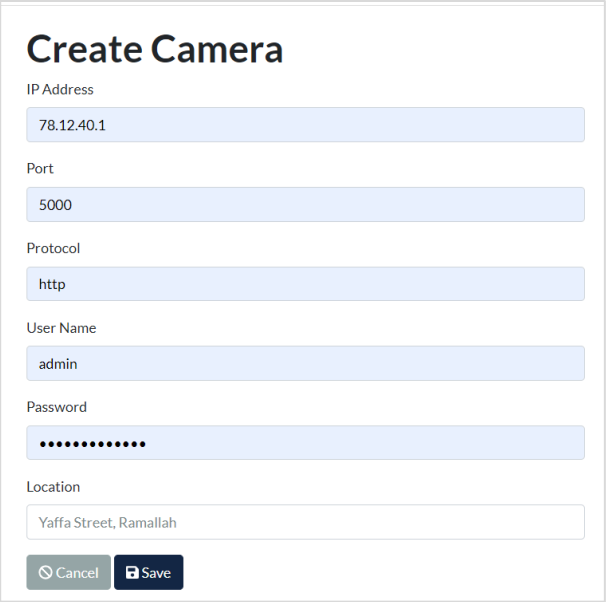

Figure 9-5: The Application Form to Add a Camera

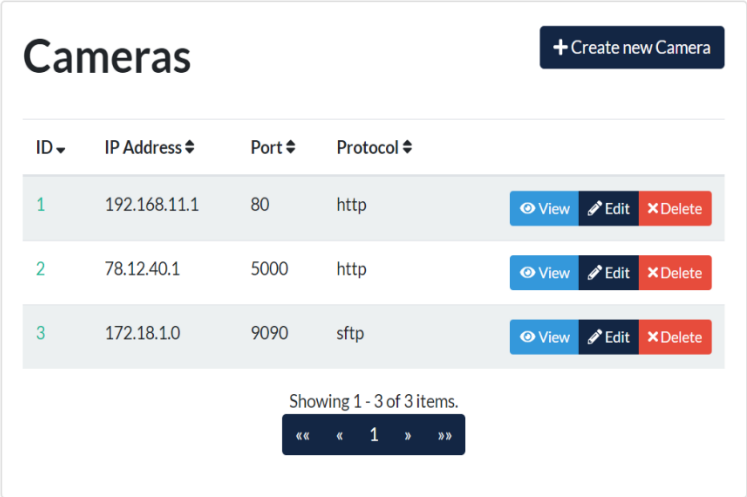

Figure 9-4: The Table of Cameras View

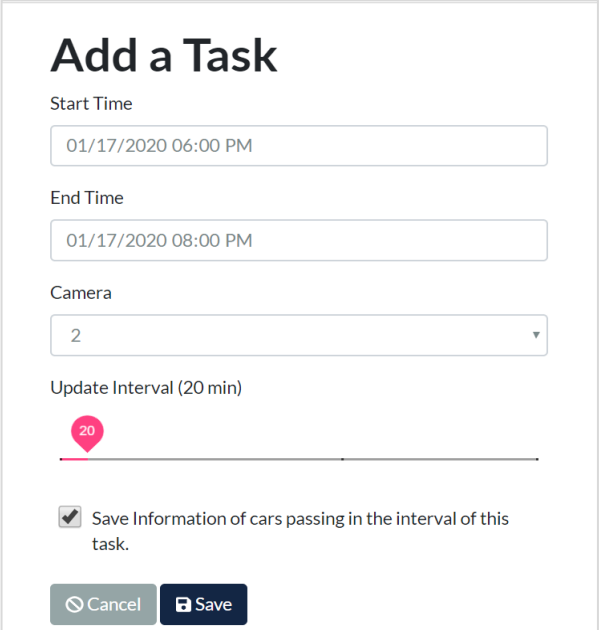

Figure 9-6: The Application Form to Add a Camera

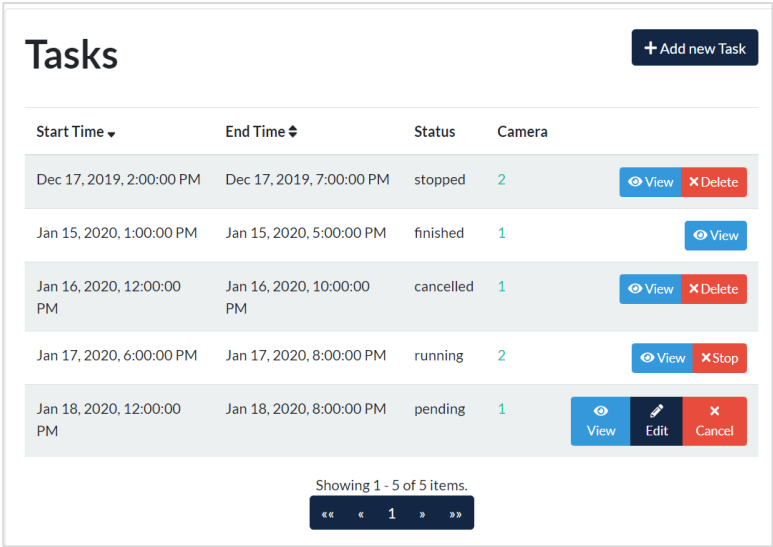

Figure 9-7: The Table of Tasks View

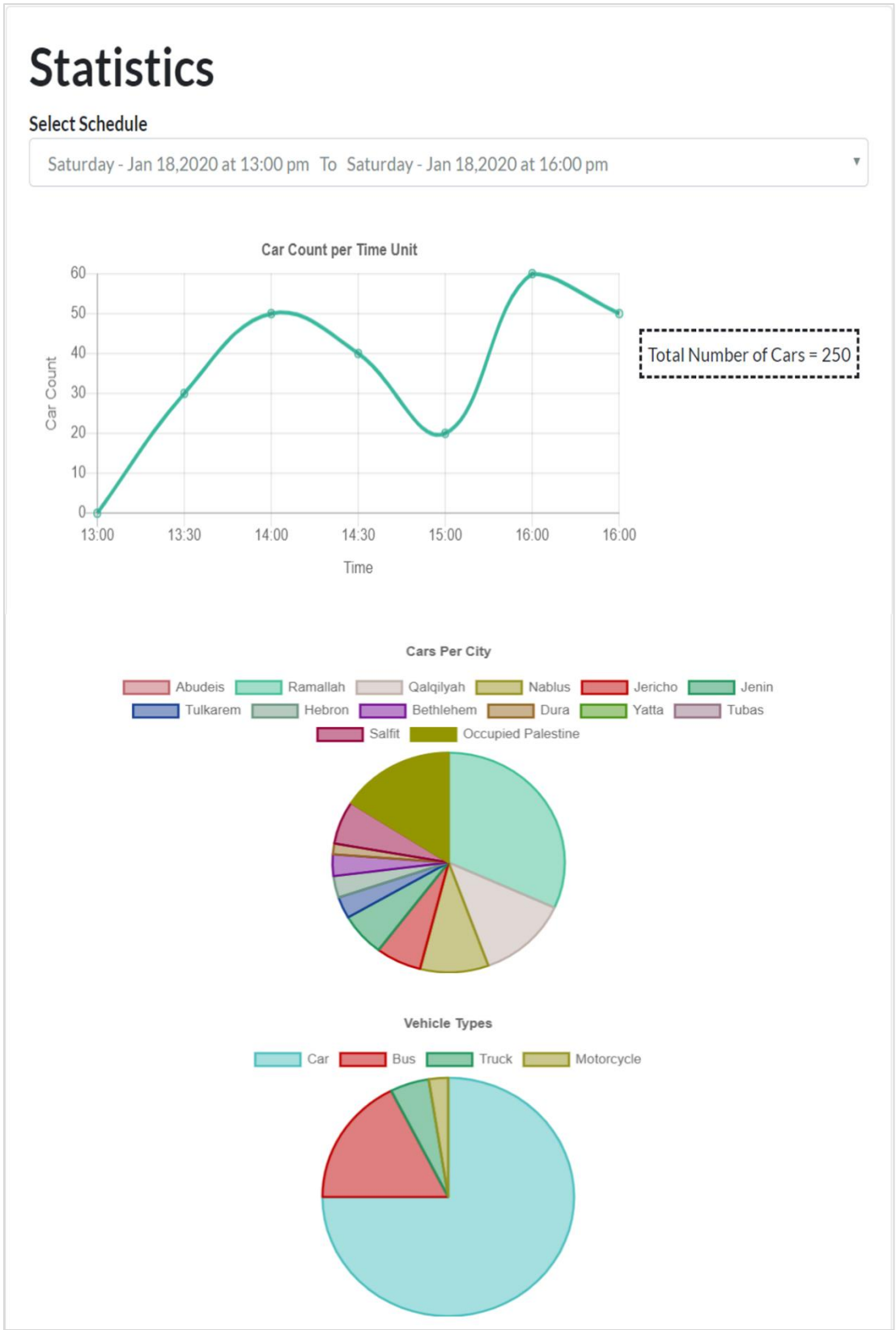

Figure 9-8: The Statistics Page in the Application

Chapter 10 Conclusion, Limitations and Future Work

10.1 Conclusion

In this project, a traffic monitoring system was developed using image processing and deep learning techniques. The constructed system is able to provide useful statistics and reports related to road traffic density and population movement. The main components of the system are vehicle detectors and trackers, and license plate detectors and recognizers. We evaluated the performance of our system using different combinations of the implementations of these components. We have also measured various performance parameters to evaluate these components in a bandwidth limited environment. Based on this study, the *YOLO* detector combined with *CSRT* tracker and *Plate Recognizer* were used in the implementation of our system to achieve the best trade-off between performance and speed. Moreover, we have shown that the bandwidth required to stream CCTV videos can be reduced significantly (to 20%) with marginal effect (< 2%) on the overall performance of the traffic monitoring system. Further savings in bandwidth are attainable with some degradation in system performance.

10.2 Limitations

Through the work on this project, many limitations were encountered. These limitations include:

- The lack of resources needed to run deep learning algorithms on large datasets for both the training and testing processes. That significantly increased the time needed to perform the evaluations and experiments.
- The specific shapes and patterns of the Palestinian license plates forced us to collect and annotate a dataset manually although there are many public annotated datasets for license plates in other countries.
- The pre-trained models for license plate recognition did not do well on the Palestinian license plates which required some improvements through multiple post-processing steps on the output of these models.
- The developed application is a real time application that uses complex deep learning techniques. Thus, besides using trackers, multithreading and GPU (optional configuration) are used to accelerate its speed. This increases the

resources required for running the system especially when it grows and the number of cameras streaming to it increases.

10.3 Future Work

In future, some improvements and enhancements can be added to the system. These improvements include:

- Collecting more data to enhance the performance of the used models.
- Adding new features to the system such as detecting some violations, controlling traffic lights or managing parking lots.
- The developed application is a web application for city administration use. However, the data extracted by processing the images captured from the located cameras can also be employed to build a web or mobile application to be used by the citizens to inform them of roads congestion status. This will be similar to what Google Maps does using GPS. However, Google Maps services for congestion detection are not available in Palestine.
- The rate-accuracy experiments done in this project included shifts in two domains: Video quantization and frame rate. More experiments can be done on other domain shifts such as video compression standard and resolution variations.

References

[1] "حوادث السير في الضفة الغربية الى الان ...بالأرقام," aliqtisadi, 18 9 2017. [Online]. Available: http://www.aliqtisadi.ps/article/49234/. [Accessed 15 7 2019].

[2] "حوادث السير في الضفة الغربية الى الان ...بالأرقام," raya, 2 2 2019. [Online]. Available: https://www.raya.ps/news/1058137.html. [Accessed 15 7 2019].

[3] S. Nathanson, "Act and Rule Utilitarianism," Internet Encyclopedia of Philosophy, [Online]. Available: https://www.iep.utm.edu/util-a-r/. [Accessed 8 1 2020].

[4] "Social Contract Theory," Ethics Unwrapped, [Online]. Available: https://ethicsunwrapped.utexas.edu/glossary/social-contract-theory. [Accessed 8 1 2020].

[5] J. Brownlee, "A Gentle Introduction to Computer Vision," Machine Learning Mastery, 19 3 2019. [Online]. Available: https://machinelearningmastery.com/what-is-computer-vision/. [Accessed 21 4 2019].

[6] *Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019.*

[7] *Walsh, Joseph & O' Mahony, Niall & Campbell, Sean & Carvalho, Anderson & Krpalkova, Lenka & Velasco-Hernandez, Gustavo & Harapanahalli, Suman & Riordan, Daniel. (2019). Deep Learning vs. Traditional Computer Vision. 10.1007/978-3-030-17795-9_10.*

[8] "Object Detection with Deep Learning: The Definitive Guide," Tryolabs, [Online]. Available: https://tryolabs.com/blog/2017/08/30/object-detection-an-overview-in-the-age-of-deep-learning/. [Accessed 23 4 2019].

[9] "What is Deep Learning and Why you need it?," Playment, 26 7 2018. [Online]. Available: https://becominghuman.ai/what-is-deep-learning-and-why-you-need-it-9e2fc0f0e61b. [Accessed 23 4 2019].

[10] M. Hargrave, "Deep Learning," investopedia, 4 30 2019. [Online]. Available: https://www.investopedia.com/terms/d/deep-learning.asp. [Accessed 5 5 2019].

[11] C. Kone, "Introducing Convolutional Neural Networks in Deep Learning," Towards Data Science, [Online]. Available: https://towardsdatascience.com/introducing-convolutional-neural-networks-in-deep-learning-400f9c3ad5e9. [Accessed 27 4 2019].

[12] H. Pokharna, "The best explanation of Convolutional Neural Networks on the Internet," Medium, 28 7 2016. [Online]. Available: https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8. [Accessed 27 4 2019].

[13] S. Goswami, "A deeper look at how Faster-RCNN works," Medium, 11 7 2018. [Online]. Available: https://medium.com/@whatdhack/a-deeper-look-at-how-faster-rcnn-works-84081284e1cd. [Accessed 27 4 2019].

[14] "Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD," CV-Tricks.com, [Online]. Available: https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/. [Accessed 28 4 2019].

[15] J. Hui, "SSD object detection: Single Shot MultiBox Detector for real-time processing," Medium, 14 3 2018. [Online]. Available: https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06. [Accessed 28 4 2019].

[16] *Liu, Wei et al. "SSD: Single Shot MultiBox Detector." Lecture Notes in Computer Science (2016): 21–37. Crossref. Web.*

[17] R. Gandhi, "R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms," Towards Data Science, 9 7 2018. [Online]. Available: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e. [Accessed 29 4 2019].

[18] A. Kathuria, "What's new in YOLO v3?," Towards Data Science, 23 4 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b. [Accessed 29 4 2019].

[19] E. Grevelink, "A Closer Look at Object Detection, Recognition and Tracking," 18 12 2017. [Online]. Available: https://software.intel.com/en-us/articles/a-closer-look-at-object-detection-recognition-and-tracking. [Accessed 23 4 2019].

[20] *Luo, Wenhan & Xing, Junliang & Milan, Anton & Zhang, Xiaoqing & Liu, Wei & Zhao, Xiaowei & Kim, Tae-Kyun. (2017). Multiple Object Tracking: A Literature Review. .*

[21] *Y. Wu, J. Lim and M. Yang, "Object Tracking Benchmark," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp. 1834-1848, 1 Sept. 2015.*

[22] *[7] Murry S., "Real-Time Multiple Object Tracking - A Study on the Importance of Speed", arXiv e-prints, pp. 8-13, Sept 2017.*

[23] S. Mallick, "Object Tracking using OpenCV," Learn OpenCV, 13 2 2017. [Online]. Available: https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/. [Accessed 24 4 2019].

[24] "Dlib Library," Dlib, [Online]. Available: http://dlib.net/. [Accessed 28 4 2019].

[25] C. Woodford, "Optical character recognition (OCR)," Explain That Stuff, 11 12 2018. [Online]. Available: https://www.explainthatstuff.com/how-ocr-works.html. [Accessed 6 1 2020].

[26] V. Chandel, "Deep Learning based Text Recognition (OCR) using Tesseract and OpenCV," Learn OpenCV, 6 6 2018. [Online]. Available: https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/. [Accessed 6 1 2020].

[27] A. Mittal, "Understanding RNN and LSTM," Towards Data Science, 12 10 2019. [Online]. Available: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e. [Accessed 6 1 2020].

[28] *Masood, Syed & Shu, Guang & Dehghan, Afshin & Ortiz, Enrique. (2017). License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks. .*

[29] Plate Recognizer, [Online]. Available: https://platerecognizer.com/blog/. [Accessed 5 1 2020].

[30] *R Rao, Ashwin & Lim, Yeon-sup & Barakat, Chadi & Legout, Arnaud & Towsley, Don & Dabbous, Walid. (2011). Network Characteristics of Video Streaming Traffic. 10.1145/2079296.2079321.*

[31] M. Southard, "The difference between Live Streaming TV and On Demand TV," Nocable, 22 2 2019. [Online]. Available: https://nocable.org/news/the-difference-between-live-streaming-tv-and-on-demand-tv. [Accessed 4 6 2019].

[32] "What is Video Encoding? Codecs and Compression Techniques," IBM, [Online]. Available: https://blog.video.ibm.com/streaming-video-tips/what-is-video-encoding-codecs-compression-techniques/. [Accessed 8 1 2020].

[33] fortinet, "UNDERSTANDING IP SURVEILLANCE CAMERA BANDWIDTH," 11 5 2017. [Online]. Available: https://www.fortinet.com/content/dam/fortinet/assets/white-papers/wp-ip-surveillance-camera.pdf. [Accessed 8 1 2020].

[34] "An Overview of H.264 Advanced Video Coding," vcodex, [Online]. Available: https://www.vcodex.com/an-overview-of-h264-advanced-video-coding/. [Accessed 8 1 2020].

[35] "Inter frame," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Inter_frame#/media. [Accessed 9 1 2020].

[36] "Image Resolution, Size and Compression," microscope, [Online]. Available: https://microscope-microscope.org/microscope-info/image-resolution/. [Accessed 9 1 2020].

[37] *M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," in IEEE Communications Surveys & Tutorials, vol. 17, no. 1, pp. 469-492, Firstquarter 2015.*

[38] *Szegedy, Christian et al. "Deep Neural Networks for Object Detection." NIPS (2013).*

[39] *W. Puarungroj and N. Boonsiriumpun, "Thai License Plate Recognition Based on Deep Learning," Procedia Computer Science, Elsevier, pp. 214-221, 2018.*

[40] *Aryal, Milan, Baine, Nicholas, "Detection, Classification, and Tracking of Objects for Autonomous Vehicles," Proceedings of the 2019 International Technical Meeting of The Institute of Navigation, Reston, Virginia, January 2019, pp. 870-883..*

[41] "Datasets," robotcar-dataset, [Online]. Available: https://robotcar-dataset.robots.ox.ac.uk/datasets/. [Accessed 17 4 2019].

[42] *Wen, Longyin & Du, Dawei & Cai, Zhaowei & Lei, Zhen & Chang, Ming-Ching & Qi, Honggang & Lim, Jongwoo & Yang, Ming-Hsuan & Lyu, Siwei. (2015). UA-DETRAC: A New Benchmark and Protocol for Multi-Object Detection and Tracking.*

[43] *Afzal Godil, Roger Bostelman Will, Shackleford, Tsai Hong, Michael Shneier (July 2014) 'Performance Metrics for Evaluating Object and Human Detection and Tracking Systems'.*

[44] J. Hui, "mAP (mean Average Precision) for Object Detection," medium, 7 3 2018. [Online]. Available: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173. [Accessed 3 6 2019].

[45] "GluonCV," GluonCV, [Online]. Available: https://gluon-cv.mxnet.io/. [Accessed 1 6 2019].

[46] *Lin, Tsung-Yi & Maire, Michael & Belongie, Serge & Hays, James & Perona, Pietro & Ramanan, Deva & Dollár, Piotr & Zitnick, C.. (2014). Microsoft COCO: Common Objects in Context.*

[47] *Bernardin, Keni & Stiefelhagen, Rainer. (2008). Evaluating multiple object tracking performance: The CLEAR MOT metrics. EURASIP Journal on Image and Video Processing. 2008. 10.1155/2008/246309.*

[48] *Redmon, Joseph & Farhadi, Ali. (2018). YOLOv3: An Incremental Improvement.*

[49] *Romain Karpinski, Devashish Lohani, Abdel Belaid. Metrics for Complete Evaluation of OCR Performance. IPCV'18 - The 22nd Int'l Conf on Image Processing, Computer Vision, & Pattern Recognition, Jul 2018, Las Vegas, United States.*

[50] "Gaussian Smoothing," [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. [Accessed 3 1 2020].

[51] "Image Thresholding," OpenCV-Python Tutorials, [Online]. Available: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html. [Accessed 3 1 2020].

[52] "Hough Line Transform," OpenCV, [Online]. Available: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html. [Accessed 3 1 2020].

[53] B. Holländer, "Deep Domain Adaptation In Computer Vision," Towards Data Science, 2 7 2019. [Online]. Available: https://towardsdatascience.com/deep-domain-adaptation-in-computer-vision-8da398d3167f. [Accessed 10 1 2020].

[54] "About FFmpeg," FFmpeg, [Online]. Available: https://www.ffmpeg.org/about.html. [Accessed 10 1 2020].

[55] M. Mulders, "What Is Spring Boot?," Stackify, 16 9 2019. [Online]. Available: https://stackify.com/what-is-spring-boot/. [Accessed 5 1 2020].

[56] I. Bodrov-Krukowski, "Angular Introduction: What It Is, and Why You Should Use It," Sitepoint, 22 3 2018. [Online]. Available: https://www.sitepoint.com/angular-introduction/. [Accessed 4 1 2020].

[57] "Flask," fullstackpython, [Online]. Available: https://www.fullstackpython.com/flask.html. [Accessed 3 1 2020].

[58] "What is Docker?," Microsoft, 31 8 2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/container-docker-introduction/docker-defined. [Accessed 5 1 2020].

[59] "What is MySQL?," MySQL, [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html. [Accessed 5 1 2020].

[60] *Plemakova, Viktoria. "Viktoria Plemakova Vehicle Detection Based on Convolutional Neural Networks." (2018).*

Appendices

Appendix A

Palestinian Plates Before 1999

A - B C D E - F G

Table A-1 Encoding of Palestinian Plates Before 1999

A	**Indicates the region where the vehicle was licensed**
Value	**Region**
4	Nablus, Jenin, Tulkarm, Qalqilya, Salfit
5	Ramallah, Jericho
8	Abu-Dis, Bethlehem, Hebron
1,2,3	Gaza
B	**Indicates the type of the vehicle**
Value	**Type**
1,2,3,4,5	Private
6,7,8	Commercial
9	Public Transportation
C D E	**Serial numbers**
G H	**Vehicle manufacturing year**

Palestinian Plates Between 1999 – 2017

A - B C D E - F G

Table A-2: Encoding of Palestinian Plates 1999-2017

A B C D E	**Indicates the city where the vehicle was licensed**	**F G**	**Indicates the type of the vehicle**
Value	**City**	**Value**	**Type**
60000 - 65999	Ramallah	90 - 98	Private, Commercial, Motorcycle, Tractor
66000 - 67999	Salfit	99	Governmental
68000 - 69999	Jericho	30	Public
70000 - 73999	Nablus	31	The vehicle is duty free
74000 - 76499	Tubas	32	Rental vehicle
76500 - 76999	Jenin	33	The vehicle had a bad accident
77000 - 78999	Tulkarm	34	Bus
79000 - 79999	Qalqilya	40	The vehicle model is before 1999
90000 - 93999	Hebron		
94000 - 96499	Bethlehem		
96500 - 97499	Dura		
97500 – 98999	Jericho		
99000 - 99999	Abu-Dis		

Palestinian Plates After 2017

A - B C D E F

Table A-3: Encoding of Palestinian Plates After 2017

A B C D E	**Serial numbers**
F	**Indicates the city where the vehicle was licensed**
Value	**Region**
A	Jenin
B	Tulkarm
C	Tubas
D	Nablus
E	Qalqilya
F	Salfit
G	Jericho
H	Ramallah
J	Abu-Dis
K	Bethlehem
L	Hebron
M	Dura
N	Yatta

Appendix B

Table B-1: MOTA and MOTP Values That Result When Evaluating Trackers on Different Skip Frames on UA-DETRAC Dataset

	MOTA				MOTP			
Skip Frames	**MEDIANFLOW**	**KCF**	**DLIB**	**CSRT**	**MEDIANFLOW**	**KCF**	**DLIB**	**CSRT**
12	81.54	81.96	81.35	81.32	88.17	80.40	85.81	84.48
24	69.17	63.28	65.51	65.04	86.08	78.13	83.18	81.44
36	59.32	48.62	51.62	51.08	84.67	77.36	81.95	80.00
48	50.31	37.88	39.80	39.82	83.84	77.19	81.32	79.18

Table B-2: MOTA and MOTP Values That Result When Evaluating YOLO with Several Trackers on Different Skip Frames on UA-DETRAC Dataset

	MOTA				MOTP			
Skip Frames	**MEDIANFLOW**	**KCF**	**DLIB**	**CSRT**	**MEDIANFLOW**	**KCF**	**DLIB**	**CSRT**
12	77.33	62.54	76.84	77.33	81.95	83.56	81.48	81.95
24	73.95	59.44	73.15	73.95	81.04	83.12	80.36	81.04
36	62.53	51.74	61.66	62.53	78.95	81.82	78.09	78.95
48	51.03	44.93	51.33	51.03	77.89	80.74	76.89	77.89

Appendix C

Table C-1: The Bitrate Values That Result When Encoding UA-DETRAC Videos with H.246 Codes and Different Values of QP and Frame Rate

Frame Rate = 25 FPS		QP = 0	
QP	**Bitrate (Mbps)**	**Frame rate (FPS)**	**Bitrate (Mbps)**
0	45.98	25	45.98
24	2.60	25 / 2	23.88
32	0.55	25 / 3	16.18
40	0.21	25 / 4	12.25
51	0.08		

Table C-2: Results of Detectors Evaluation with Different Variations in QP at Frame Rate 25 fps on UA-DETRAC Dataset

	F-Measure		AP	
QP	**SSD**	**YOLO**	**SSD**	**YOLO**
0	79.36	86.19	0.61	0.67
24	79.56	86.84	0.61	0.68
32	78.98	86.18	0.60	0.67
40	74.67	82.59	0.53	0.62
51	39.39	47.14	0.20	0.26

Table C-3: MOTA Results of Trackers Evaluation with Different Variations in QP at Frame Rate 25 fps on UA-DETRAC Dataset

	MOTA					
	skip 4			skip 8		
QP	**CSRT**	**DLIB**	**MEDIANFLOW**	**CSRT**	**DLIB**	**MEDIANFLOW**
0	94.19	94.08	93.85	87.12	86.93	86.72
24	94.18	94.06	93.77	87.04	86.86	86.64
32	94.19	93.95	93.78	87.11	86.73	86.75
40	94.23	93.91	93.96	87.16	86.71	87.08
51	93.74	91.77	91.98	86.16	82.30	81.44

Table C-4: MOTP Results of Trackers Evaluation with Different Variations in QP at Frame Rate 25 fps on UA-DETRAC Dataset

	MOTP					
	skip 4			skip 8		
QP	**CSRT**	**DLIB**	**MEDIANFLOW**	**CSRT**	**DLIB**	**MEDIANFLOW**
0	89.49	89.86	91.47	86.43	87.47	89.45
24	89.53	89.90	91.46	86.48	87.52	89.43
32	89.52	89.87	91.46	86.47	87.47	89.45
40	89.54	89.76	91.43	86.43	87.21	89.42
51	88.35	87.67	88.73	85.16	84.93	85.31

Table C-5: MOTA Results of Trackers Evaluation with Different Variations in Frame Rate at QP =0 on UA-DETRAC Dataset

	MOTA					
	skip 4			skip 8		
QP	**CSRT**	**DLIB**	**MEDIANFLOW**	**CSRT**	**DLIB**	**MEDIANFLOW**
0	94.19	94.08	93.85	87.12	86.93	86.72
24	87.08	86.01	86.73	74.83	73.3	77.4
32	80.13	74.49	80.23	60.30	53.41	69.00
40	72.01	56.22	74.36	45.43	31.94	61.5
51	94.19	94.08	93.85	87.12	86.93	86.72

Table C-6: MOTP Results of Trackers Evaluation with Different Variations in Frame Rate at QP =0 on UA-DETRAC Dataset

	MOTP					
	skip 4			skip 8		
QP	**CSRT**	**DLIB**	**MEDIANFLOW**	**CSRT**	**DLIB**	**MEDIANFLOW**
0	89.49	89.86	91.47	86.43	87.47	89.45
24	86.51	86.66	89.3	82.86	83.46	87.24
32	83.99	84.25	87.93	80.43	81.11	85.79
40	82.09	82.78	87.02	78.92	79.87	84.67
51	89.49	89.86	91.47	86.43	87.47	89.45

Table C-7: Results of Evaluating YOLO Detector with Different Trackers when Encoding UA-DETRAC Dataset with Different Values of QP and Frame Rate.

		MOTA			MOTP		
Frame Rate (FPS)	**QP**	**CSRT**	**DLIB**	**MEDIANFLOW**	**CSRT**	**DLIB**	**MEDIANFLOW**
25	24	80.88	81.27	67.51	82.95	83.11	84.15
	32	80.10	80.49	66.89	82.91	83.06	84.14
	40	74.69	75.07	62.54	82.48	82.56	83.74
25 / 2	24	78.07	77.95	64.79	81.98	82.03	83.77
	32	77.44	77.37	64.24	81.95	82.01	83.76
	40	72.34	72.22	60.05	81.48	81.43	83.34
25 / 3	24	74.22	69.66	61.31	80.92	81.02	83.51
	32	73.64	69.20	60.81	80.84	80.89	83.45
	40	68.86	63.89	56.87	80.28	80.35	83.03
25 / 4	24	68.39	55.07	57.52	79.88	80.32	83.18
	32	68.10	54.85	57.27	79.75	80.22	83.09
	40	63.18	50.43	53.43	79.19	79.61	82.59

Table C-8: The Relation Between the F-Measure of Plate Detection and the Quantization of the Collected Palestinian Dataset When the Frame Rate is 25 FPS

	F-Measure		
QP	**Plate Recognizer**	**Sighthound**	**YOLO**
0	97.58%	98.00%	96.53%
24	98.32%	98.00%	96.81%
32	98.18%	98.01%	96.98%
40	98.48%	98.01%	96.96%
51	97.58%	98.00%	96.53%

Table C-9: The Relation Between the Character Accuracy of Plate Recognition and the Quantization of the Collected Palestinian Dataset When the Frame Rate is 25 FPS

	F-Measure					
	Plate Accuracy			Character Accuracy		
QP	**Plate Recognizer**	**Sighthound**	**YOLO**	**Plate Recognizer**	**Sighthound**	**YOLO**
0	92.88%	85.76%	70.59%	98.66%	96.91%	93.32%
24	90.68%	88.82%	70.90%	98.34%	97.57%	93.77%
32	92.26%	89.47%	66.56%	98.66%	97.31%	92.20%
40	89.78%	81.73%	67.49%	97.89%	95.65%	91.79%
51	92.88%	85.76%	70.59%	98.66%	96.91%	93.32%

Appendix D

Table D-1: The Prices of Internet Services Provided by Paltel Company.

Speed	Price (NIS)/ Month
Up to 4 Mbps	44
Up to 8 Mbps	58
Up to 16 Mbps	75
Up to 30 Mbps	103
Up to 50 Mbps	117
Up to 100 Mbps	128